Fitted Q-iteration in continuous action-space MDPs

Authored by:

R?mi Munos
Csaba Szepesv?ri
Andr?s Antos

Abstract

We consider continuous state, continuous action batch reinforcement learning where the goal is to learn a good policy from a sufficiently rich trajectory generated by another policy. We study a variant of fitted Q-iteration, where the greedy action selection is replaced by searching for a policy in a restricted set of candidate policies by maximizing the average action values. We provide a rigorous theoretical analysis of this algorithm, proving what we believe is the first finite-time bounds for value-function based algorithms for continuous state- and action-space problems.

1 Paper Body

We consider continuous state, continuous action batch reinforcement learning where the goal is to learn a good policy from a sufficiently rich trajectory generated by some policy. We study a variant of fitted Q-iteration, where the greedy action selection is replaced by searching for a policy in a restricted set of candidate policies by maximizing the average action values. We provide a rigorous analysis of this algorithm, proving what we believe is the first finite-time bound for value-function based algorithms for continuous state and action problems.

1

Preliminaries

We will build on the results from [1, 2, 3] and for this reason we use the same notation as these papers. The unattributed results cited in this section can be found in the book [4]. A discounted MDP is defined by a quintuple (X , A, P, S, ?), where X is the (possible infinite) state space, A is the set of actions, P : X ? A ? M (X) is the transition probability kernel with P (?—x, a) defining the next-state distribution upon taking action a from state x, S(?—x, a) gives the corresponding distribution of immediate rewards, and ? ? (0, 1) is the discount factor. Here X is a measurable space and M (X) denotes the set of all probability measures over X . The Lebesguemeasure shall be denoted

by ?. We start with the following mild assumption on the MDP: Assumption A1 (MDP Regularity) X is a compact subset of the dX -dimensional Euclidean space, ? max A is a compact subset of [?A? , A?]dA . The random immediate rewards are bounded by R R and that the expected immediate reward function, r(x, a) = rS(dr—x, a), is uniformly bounded by Rmax : krk? ? Rmax . A policy determines the next action given the past observations. Here we shall deal with stationary (Markovian) policies which choose an action in a stochastic way based on the last observation only. The value of a policy ? when it is started from a state x is defined as the expected discounted Ptotal ? reward that is encountered while the policy is executed: V ? (x) = E? [t=0 ? t Rt —X0 = x]. Here Rt ? S(?—Xt , At) is the reward received at time step t, the state, Xt , evolves according to Xt+1 ? ? Also with: Computer and Automation Research Inst. of the Hungarian Academy of Sciences Kende u. 13-17, Budapest 1111, Hungary.

1

P (?—Xt , At), where At is sampled from the distribution determined by ?. We use Q? : X ? A ? R P? to denote the action-value function of policy ?: Q? (x, a) = E? [t=0 ? t Rt —X0 = x, A0 = a]. The goal is to find a policy that attains the best possible values, V ? (x) = sup? V ? (x), at all states ? x ? X . Here V ? is called the optimal value function and a policy ? ? that satisfies V ? (x) = ? ? ? V (x) for all x ? X is called optimal. The optimal action-value function Q (x, a) is Q (x, a) = sup? Q? (x, a). We say that a (deterministic stationary) policy ? is greedy w.r.t. an action-value function Q ? B(X ? A), and we write ? = ? ? (?; Q), if, for all x ? X , ?(x) ? argmaxa?A Q(x, a). Under mild technical assumptions, such a greedy policy always exists. Any greedy policy w.r.t. Q? ? is optimal. For ? : X ? A we ? A), by R define its evaluation operator, T : B(X ?? A) ?? B(X ? (T Q)(x, a) = r(x, a) + ? X Q(y, ?(y)) P (dy—x, a). It is known that Q = T Q? . Further, if weR let the Bellman operator, T : B(X ? A) ? B(X ? A), defined by (T Q)(x, a) = r(x, a) + ? X supb?A Q(y, b) P (dy—x, a) then Q? = T Q? . It is known that V ? and Q? are bounded by Rmax /(1 ? ?), just like Q? and V ? . For ? : X ? A, the operator E ? : B(X ? A) ? B(X) is defined by (E ? Q)(x) = Q(x, ?(x)), while E : B(X ? A) ? B(X) is defined by (EQ)(x) = supa?A Q(x, a). Throughout the paper F ? {f : X ? A ? R} will denote a subset of real-valued functions over the state-action space X ? A and ? ? AX will R be a set of policies. For ? ? M (X) and f : X ? R p measurable, we let (for p ? 1) kf kp,? = X —f (x)—p ?(dx). We simply write kf k? for kf k2,? . R R 2 Further, we extend k?k? to F by kf k? = A X —f —2 (x, a) d?(x) d?A (a), where ?RA is the uniform distribution over A. We shall use the shorthand notation ?f to denote the integral f (x)?(dx). We denote the space of bounded measurable functions with domain X by B(X). Further, the space of measurable functions bounded by 0 ¡ K ¡ ? shall be denoted by B(X ; K). We let k?k? denote the supremum norm.

2

Fitted Q-iteration with approximate policy maximization

We assume that we are given a finite trajectory, {(Xt , At , Rt)}1?t?N , generated by some stochastic stationary policy ?b , called the behavior policy:

At ? ?b (?—Xt), Xt+1 ? P (?—Xt , At), Rt ? def S(?—Xt , At), where ?b (?—x) is a density with ?0 = inf (x,a)?X ?A ?b (a—x) ¿ 0. The generic recipe for fitted Q-iteration (FQI) [5] is Qk+1 = Regress(Dk (Qk)),

(1)

where Regress is an appropriate regression procedure and Dk (Qk) is a dataset defining a regression problem in the form of a list of data-point pairs: ?h ? i Dk (Qk) = (Xt , At), Rt + ? max Qk (Xt+1 , b) .1 b?A

1?t?N

Fitted Q-iteration can be viewed as approximate value iteration applied to action-value functions. To see this note that value iteration would assign the value (T Qk)(x, a) = r(x, a) + R ? maxb?A Qk (y, b) P (dy—x, a) to Qk+1 (x, a) [6]. Now, remember that the regression function for the jointly distributed random variables (Z, Y) is defined by the conditional expectation of Y given Z: m(Z) = E [Y —Z]. Since for any fixed function Q, E [Rt + ? maxb?A Q(Xt+1 , b)—Xt , At] = (T Q)(Xt , At), the regression function corresponding to the data Dk (Q) is indeed T Q and hence if FQI solved the regression problem defined by Qk exactly, it would simulate value iteration exactly. However, this argument itself does not directly lead to a rigorous analysis of FQI: Since Qk is obtained based on the data, it is itself a random function. Hence, after the first iteration, the ?target? function in FQI becomes random. Furthermore, this function depends on the same data that is used to define the regression problem. Will FQI still work despite these issues? To illustrate the potential difficulties consider a dataset where X1 , . . . , XN is a sequence of independent random variables, which are all distributed uniformly at random in [0, 1]. Further, let M be a random integer greater than N which is independent of the dataset (Xt)N t=1 . Let U be another random variable, uniformly distributed in [0, 1]. Now define the regression problem by Yt = fM,U (Xt), where fM,U (x) = sgn(sin(2M 2?(x + U))). Then it is not hard to see that no matter how big N is, no procedure can 1

Since the designer controls Qk , we may assume that it is continuous, hence the maximum exists.

2

estimate the regression function fM,U with a small error (in expectation, or with high probability), even if the procedure could exploit the knowledge of the specific form of fM,U . On the other hand, if we restricted M to a finite range then the estimation problem could be solved successfully. The example shows that if the complexity of the random functions defining the regression problem is uncontrolled then successful estimation might be impossible. Amongst the many regression methods in this paper we have chosen to work with least-squares methods. In this case Equation (1) takes the form ? ? ??2 N X 1 Q(Xt , At) ? Rt + ? max Qk (Xt+1 , b) Qk+1 = argmin . (2) b?A Q?F t=1 ?b (At —Xt) We call this method the least-squares fitted Q-iteration (LSFQI) method. Here we introduced the weighting 1/?b (At —Xt) since we do not want to give more weight to those actions that are preferred by the behavior policy. Besides this weighting, the only parameter of the method is the function set F. This function set should be chosen carefully, to keep a

balance between the representation power and the number of samples. As a specific example for F consider neural networks with some fixed architecture. In this case the function set is generated by assigning weights in all possible ways to the neural net. Then the above minimization becomes the problem of tuning the weights. Another example is to use linearly parameterized function approximation methods with appropriately selected basis functions. In this case the weight tuning problem would be less demanding. Yet another possibility is to let F be an appropriate restriction of a Reproducing Kernel Hilbert Space (e.g., in a ball). In this case the training procedure becomes similar to LS-SVM training [7]. As indicated above, the analysis of this algorithm is complicated by the fact that the new dataset is defined in terms of the previous iterate, which is already a function of the dataset. Another complication is that the samples in a trajectory are in general correlated and that the bias introduced by the imperfections of the approximation architecture may yield to an explosion of the error of the procedure, as documented in a number of cases in, e.g., [8]. Nevertheless, at least for finite action sets, the tools developed in [1, 3, 2] look suitable to show that under appropriate conditions these problems can be overcome if the function set is chosen in a judicious way. However, the results of these works would become essentially useless in the case of an infinite number of actions since these previous bounds grow to infinity with the number of actions. Actually, we believe that this is not an artifact of the proof techniques of these works, as suggested by the counterexample that involved random targets. The following result elaborates this point further: Proposition 2.1. Let F ? B(X ? A). Then even if the pseudo-dimension of F is finite, the fatshattering function of ? ? ? Fmax = VQ : VQ (?) = max Q(?, a), Q ? F a?A

2

can be infinite over (0, 1/2).

Without going into further details, let us just note that the finiteness of the fat-shattering function is a sufficient and necessary condition for learnability and the finiteness of the fat-shattering function is implied by the finiteness of the pseudo-dimension [9].The above proposition thus shows that without imposing further special conditions on F, the learning problem may become infeasible. One possibility is of course to discretize the action space, e.g., by using a uniform grid. However, if the action space has a really high dimensionality, this approach becomes unfeasible (even enumerating 2dA points could be impossible when dA is large). Therefore we prefer alternate solutions. Another possibility is to make the functions in F, e.g., uniformly Lipschitz in their state coordinates. ? Then the same property will hold for functions in Fmax and hence by a classical result we can bound the capacity of this set (cf. pp. 353?357 of [10]). One potential problem with this approach is that this way it might be difficult to get a fine control of the capacity of the resulting set. 2 The proof of this and the other results are given in the appendix, available in the extended version of this paper, downloadable from http://hal.inria.fr/inria-00185311/en/.

3

In the approach explored here we modify the fitted Q-iteration algorithm by introducing a policy set ? and a search over this set for an approximately

greedy policy in a sense that will be made precise in a minute. Our algorithm thus has four parameters: F, ?, K, Q0 . Here F is as before, ? is a user-chosen set of policies (mappings from X to A), K is the number of iterations and Q0 is an initial value function (a typical choice is Q0 ? 0). The algorithm computes a sequence of iterates (Qk , ? ?k), k = 0, . . . , K, defined by the following equations: ? ?0 Qk+1 ? ?k+1

=

argmax ???

=

argmin

N X

argmax ???

Q0 (Xt , ?(Xt)),

t=1

Q?F

=

N X

t=1 N X

? ? ??2 1 Q(Xt , At) ? Rt + ?Qk (Xt+1 , ? ?k (Xt+1)) , ?b (At —Xt)

(3)

Qk+1 (Xt , ?(Xt)).

(4)

t=1

Thus, (3) is similar to (2), while (4) defines the policy search problem. The policy search will generally be solved by a gradient procedure or some other appropriate method. The cost of this step will be primarily determined by how well-behaving the iterates Qk+1 are in their action arguments. For example, if they were quadratic and if ? was linear then the problem would be a quadratic optimization problem. However, except for special cases3 the action value functions will be more complicated, in which case this step can be expensive. Still, this cost could be similar to that of searching for the maximizing actions for each t = 1, . . . , N if the approximately maximizing actions are similar across similar states. This algorithm, which we could also call a fitted actor-critic algorithm, will be shown to overcome the above mentioned complexity control problem provided that the complexity of ? is controlled appropriately. Indeed, in this case the set of possible regression problems is determined by the set F?? = { V : V (?) = Q(?, ?(?)), Q ? F, ? ? ? } , and the proof will rely on controlling the complexity of F?? by selecting F and ? appropriately.

3 3.1

The main theoretical result Outline of the analysis

In order to gain some insight into the behavior of the algorithm, we provide a brief summary of its error analysis. The main result will be presented subsequently. For f ,Q ? F and a policy ?, we define the tth TD-error as follows: dt (f ; Q, ?) = Rt + ?Q(Xt+1 , ?(Xt+1)) ? f (Xt , At). Further, we define the empirical loss function by N X d2t (f ; Q, ?) ? N (f ; Q, ?) = 1 , L N t=1 ?(A)?b (At —Xt)

where the normalization with ?(A) is introduced for mathematical convenience. Then (3) can be ? N (f ; Qk , ? written compactly as Qk+1 = argminf ?F L ?k). ? N (f ; Q, ?) is an The algorithm can then be motivated by the observation that for any f ,Q, and ?, L unbiased estimate of def 2 L(f ; Q, ?) = kf ? T ? Qk? + L? (Q, ?), (5) where the first term is the error we are interested in and the second term captures the variance of the random samples: Z L? (Q, ?) = E [Var [R1 + ?Q(X2 , ?(X2))—X1 , A1 = a]] d?A (a). A 3 Linear quadratic regulation is such a nice case. It is interesting to note that in this special case the obvious choices for F and ? yield zero error in the limit, as can be proven based on the main result of this paper.

4

h i ? N (f ; Q, ?) = L(f ; Q, ?). This result is stated formally by E L Since the variance term in (5) is independent of f , argminf ?F L(f ; Q, ?) = 2 ? argminf ?F kf ? T Qk? . Thus, if ? ?k were greedy w.r.t. Qk then argminf ?F L(f ; Qk , ? ?k) = 2 argminf ?F kf ? T Qk k? . Hence we can still think of the procedure as approximate value iteration over the space of action-value functions, projecting T Qk using empirical risk minimization on the space F w.r.t. k?k? distances in an approximate manner. Since ? ?k is only approximately greedy, we will have to deal with both the error coming from the approximate projection and the error coming from the choice of ? ?k . To make this clear, we write the iteration in the form Qk+1 = T ??k Qk + ?0k = T Qk + ?0k + (T ??k Qk ? T Qk) = T Qk + ?k , def

where ?0k is the error committed while computing T ??k Qk , ?00k = T ??k Qk ? T Qk is the error committed because the greedy policy is computed approximately and ?k = ?0k + ?00k is the total error of step k. Hence, in order to show that the procedure is well behaved, one needs to show that both errors are controlled and that when the errors are propagated through these equations, the resulting error stays controlled, too. Since we are ultimately interested in the performance of the policy obtained, we will also need to show that small action-value approximation errors yield small performance losses. For these we need a number of assumptions that concern either the training data, the MDP, or the function sets used for learning. 3.2

Assumptions

3.2.1

Assumptions on the training data

We shall assume that the data is rich, is in a steady state, and is fast-mixing, where, informally, mixing means that future depends weakly on the past. Assumption A2 (Sample Path Properties) Assume that {(Xt , At , Rt)}t=1,...,N is the sample path of ?b , a stochastic stationary policy. Further, assume that {Xt } is strictly stationary (Xt ? ? ? M (X)) and exponentially ?-mixing with the actual rate given by the parameters (?, b, ?).4 We further assume that the sampling policy ?b satisfies ?0 = inf (x,a)?X ?A ?b (a—x) ¿ 0. The ?-mixing property will be used to establish tail inequalities for certain empirical processes.5 Note that the mixing coefficients do not need to be known. In the case when no mixing condition is satisfied, learning might be impossible. To see this just consider the case when X1 = X2 = . . . = XN . Thus, in this

case the learner has many copies of the same random variable and successful generalization is thus impossible. We believe that the assumption that the process is in a steady state is not essential for our result, as when the process reaches its steady state quickly then (at the price of a more involved proof) the result would still hold. 3.2.2

Assumptions on the MDP

In order to prevent the uncontrolled growth of the errors as they are propagated through the updates, we shall need some assumptions on the MDP. A convenient assumption is the following one [11]: Assumption A3 (Uniformly stochastic transitions) For all x ? X and a ? A, assume that P (?—x, a) is absolutely continuous w.r.t. ? and the derivative of P w.r.t. ? is bounded ? ? Radon-Nikodym def ? dP (?—x,a) ? uniformly with bound C? : C? = supx?X ,a?A ? d? ? ¡ +?. ?

Note that by the definition of measure differentiation, Assumption A3 means that P (?—x, a) ? C? ?(?). This assumption essentially requires the transitions to be noisy. We will also prove (weaker) results under the following, weaker assumption: 4

For the definition of ?-mixing, see e.g. [2]. We say ?empirical process? and ?empirical measure?, but note that in this work these are based on dependent (mixing) samples. 5

5

Assumption A4 (Discounted-average concentrability of future-state distributions) Given ?, ?, m ? 1 and an arbitrary sequence of stationary policies {?m }m?1 , assume that the futuredef state distribution ?P ?1 P ?2 . . .?P ?m is absolutely continuous w.r.t. ?. Assume that c(m) = ? ?1 ?2 ?m ? P def ? sup?1 ,...,?m ? d(?P Pd? ...P) ? satisfies m?1 m? m?1 c(m) ¡ +?. We shall call C?,? = ? ? ? P P max (1 ? ?)2 m?1 m? m?1 c(m), (1 ? ?) m?1 ? m c(m) the discounted-average concentrability coefficient of the future-state distributions. The number c(m) measures how much ? can get amplified in m steps as compared to the reference distribution ?. Hence, in general we expect c(m) to grow with m. In fact, the condition that C?,? is finite is a growth rate condition on c(m). Thanks to discounting, C?,? is finite for a reasonably large class of systems (see the discussion in [11]). A related assumption is needed in the error analysis of the approximate greedy step of the algorithm: Assumption A5 (The random policy ?makes no peak-states?) Consider the distribution ? = (? ? ?A)P which is the distribution of a state that results from sampling an initial state according to ? and then executing an action which is selected uniformly at random.6 Then ?? = kd?/d?k? ¡ +?. Note that under Assumption A3 we have ?? ? C? . This (very mild) assumption means that after one step, starting from ? and executing this random policy, the probability of the next state being in a set is upper bounded by ?? -times the probability of the starting state being in the same set. def

Besides, we assume that A has the following regularity property: Let Py(a, h, ?) = ? ? (a0 , v) ? RdA +1 : ka ? a0 k1 ? ?, 0 ? v/h ? 1 ? ka ? a0 k1 /? denote the pyramid with hight ? def ? h and base given by the ‘1 -ball B(a, ?) = a0 ? RdA : ka ? a0 k1 ? ? centered at a. Assumption A6 (Regularity of

the action space) We assume that there exists ? ¿ 0, such that for all a ? A, for all ? ¿ 0, ? ? ?(Py(a, 1, ?) ? (A ? R)) ?(A) ? min ?, . ?(Py(a, 1, ?)) ?(B(a, ?)) For example, if A is an ‘1 -ball itself, then this assumption will be satisfied with ? = 2?dA . Without assuming any smoothness of the MDP, learning in infinite MDPs looks hard (see, e.g., [12, 13]). Here we employ the following extra condition: Assumption A7 (Lipschitzness of the MDP in the actions) Assume that the transition probabilities and rewards are Lipschitz w.r.t. their action variable, i.e., there exists LP , Lr ¿ 0 such that for all (x, a, a0) ? X ? A ? A and measurable set B of X , —P (B—x, a) ? P (B—x, a0)— ? LP ka ? a0 k1 , —r(x, a) ? r(x, a0)— ? Lr ka ? a0 k1 .

Note that previously Lipschitzness w.r.t. the state variables was used, e.g., in [11] to construct consistent planning algorithms. 3.2.3

Assumptions on the function sets used by the algorithm

These assumptions are less demanding since they are under the control of the user of the algorithm. However, the choice of these function sets will greatly influence the performance of the algorithm, as we shall see it from the bounds. The first assumption concerns the class F: Assumption A8 (Lipschitzness of candidate action-value functions) Assume F ? B(X ? A) and that any elements of F is uniformly Lipschitz in its action-argument in the sense that —Q(x, a) ? Q(x, a0)— ? LA ka ? a0 k1 holds for any x ? X , a,a0 ? A, and Q ? F . 6

Remember that ?A denotes the uniform distribution over the action set A.

6

We shall also need to control the capacity of our function sets. We assume that the reader is familiar with the concept of VC-dimension.7 Here we use the pseudo-dimension of function sets that builds upon the concept of VC-dimension: Definition 3.1 (Pseudo-dimension). The pseudo-dimension VF + of F is defined as the VCdimension of the subgraphs of functions in F (hence it is also called the VC-subgraph dimension of F). Since A is multidimensional, we define V?+ to be the sum of the pseudo-dimensions of the coordinate projection spaces, ?k of ?: V

?+

=

dA X

V? + ,

k=1

k

?k = { ?k : X ? R : ? = (?1 , . . . , ?k , . . . , ?dA) ? ? } .

Now we are ready to state our assumptions on our function sets: Assumption A9 (Capacity of the function and policy sets) Assume that F ? B(X ? A; Qmax) for Qmax ¿ 0 and VF + ¡ +?. Also, A ? [?A? , A?]dA and V?+ ¡ +?. Besides their capacity, one shall also control the approximation power of the function sets involved. Let us first consider the policy set ?. Introduce e? (F, ?) = sup inf ?(EQ ? E ? Q). Q?F ???

Note that inf ??? ?(EQ ? E ? Q) measures the quality of approximating ?EQ by ?E ? Q. Hence, e? (F, ?) measures the worst-case approximation error of ?EQ as Q is changed within F. This can be made small by choosing ? large.

Another related quantity is the one-step Bellman-error of F w.r.t. ?. This is defined as follows: For a fixed policy ?, the one-step Bellman-error of F w.r.t. T ? is defined as E1 (F; ?) = sup inf kQ0 ? T ? Qk? . 0 Q?F Q ?F

Taking again a pessimistic approach, the one-step Bellman-error of F is defined as E1 (F, ?) = sup E1 (F; ?). ???

Typically by increasing F, E1 (F, ?) can be made smaller (this is discussed at some length in [3]). However, it also holds for both ? and F that making them bigger will increase their capacity (pseudo-dimensions) which leads to an increase of the estimation errors. Hence, F and ? must be selected to balance the approximation and estimation errors, just like in supervised learning. 3.3

The main result

Theorem 3.2. Let ?K be a greedy policy w.r.t. QK , i.e. ?K (x) ? argmaxa?A QK (x, a). Then under Assumptions A1, A2, and A5?A9, for all ? ¿ 0 we have with probability at least 1 ? ?: given Assumption A3 (respectively A4), kV ? ? V ?K k? (resp. kV ? ? V ?K k1,?), is bounded by ? ?? ? d 1+1 ?+1 ? ? A ? ? 4? (log N + log(K/?)) K ? + ? , C ?E1 (F, ?) + e? (F, ?) + 1/4 ? ? N ? ? A where C depends on dA , VF + , (V?+)dk=1 , ?, ?, b, ?, C? (resp. C?,?), ?? , LA , LP ,Lr , ?, ?(A), ?0 , k

?+1

? max , and A? . In particular, C scales with V 4?(dA +1) , where V = 2VF + + V?+ Qmax , Rmax , R plays the role of the ?combined effective? dimension of F and ?. 7 Readers not familiar with VC-dimension are suggested to consult a book, such as the one by Anthony and Bartlett [14].

7

4

Discussion

We have presented what we believe is the first finite-time bounds for continuous-state and actionspace RL that uses value functions. Further, this is the first analysis of fitted Q-iteration, an algorithm that has proved to be useful in a number of cases, even when used with non-averagers for which no previous theoretical analysis existed (e.g., [15, 16]). In fact, our main motivation was to show that there is a systematic way of making these algorithms work and to point at possible problem sources the same time. We discussed why it can be difficult to make these algorithms work in practice. We suggested that either the set of action-value candidates has to be carefully controlled (e.g., assuming uniform Lipschitzness w.r.t. the state variables), or a policy search step is needed, just like in actorcritic algorithms. The bound in this paper is similar in many respects to a previous bound of a Bellman-residual minimization algorithm [2]. It looks that the techniques developed here can be used to obtain results for that algorithm when it is applied to continuous action spaces. Finally, although we have not explored them here, consistency results for FQI can be obtained from our results using standard methods, like the methods of sieves. We believe that the methods developed here will eventually lead to algorithms where the function approximation methods are chosen based on the data (similar to adaptive regression methods) so as to optimize performance, which in our opinion is one of the biggest open questions in RL. Currently we are exploring this pos-

sibility. Acknowledgments Andr?as Antos would like to acknowledge support for this project from the Hungarian Academy of Sciences (Bolyai Fellowship). Csaba Szepesv?ari greatly acknowledges the support received from the Alberta Ingenuity Fund, NSERC, the Computer and Automation Research Institute of the Hungarian Academy of Sciences.

2 References

[1] A. Antos, Cs. Szepesv?ari, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In COLT-19, pages 574?588, 2006. [2] A. Antos, Cs. Szepesv?ari, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning, 2007. (accepted). [3] A. Antos, Cs. Szepesv?ari, and R. Munos. Value-iteration based fitted policy iteration: learning with a single trajectory. In IEEE ADPRL, pages 330?337, 2007. [4] D. P. Bertsekas and S.E. Shreve. Stochastic Optimal Control (The Discrete Time Case). Academic Press, New York, 1978. [5] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. Journal of Machine Learning Research, 6:503?556, 2005. [6] R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. Bradford Book. MIT Press, 1998. [7] N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines (and other kernel-based learning methods). Cambridge University Press, 2000. [8] J.A. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In NIPS-7, pages 369?376, 1995. [9] P.L. Bartlett, P.M. Long, and R.C. Williamson. Fat-shattering and the learnability of real-valued functions. Journal of Computer and System Sciences, 52:434?452, 1996. [10] A.N. Kolmogorov and V.M. Tihomirov. ?-entropy and ?-capacity of sets in functional space. American Mathematical Society Translations, 17(2):277?364, 1961. [11] R. Munos and Cs. Szepesv?ari. Finite time bounds for sampling based fitted value iteration. Technical report, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Kende u. 13-17, Budapest 1111, Hungary, 2006. [12] A.Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, pages 406?415, 2000. [13] P.L. Bartlett and A. Tewari. Sample complexity of policy search with known dynamics. In NIPS-19. MIT Press, 2007. [14] M. Anthony and P. L. Bartlett. Neural Network Learning: Theoretical Foundations. Cambridge University Press, 1999. [15] M. Riedmiller. Neural fitted Q iteration ? first experiences with a data efficient neural reinforcement learning method. In 16th European Conference on Machine Learning, pages 317?328, 2005. [16] S. Kalyanakrishnan and P. Stone. Batch reinforcement learning in a complex domain. In AAMAS-07, 2007.

8