published in: ESANN'96, Proc. of the 1996 Conference, Bruges, pp. 25-30

Practicing Q-Learning

Jörg Bruske, Ingo Ahrns, Gerald Sommer

Christian Albrechts University of Kiel - Computer Science Institute
Preusserstr. 1-9 - 24118 Kiel - Germany
e-mail: jbr@informatik.uni-kiel.de

Abstract. Q-Learning has gained increasing attention as a promising real time learning scheme from delayed reinforcement. Being compact, model free and theoretically optimal it is commonly preferred to AHC-Learning and its derivatives. However, it has long been noticed that theoretical optimality has to be sacrificed in order to meet the constraints of most applications. In this article we report of experiments with modified Q-Learning algorithms together with their key ingredients for practical success in reinforcement learning. These include *optimistic initialization*, the principle of *piecewise constancy of policy* and the use of *activity traces*. Finally, we extend these algorithms for *growing RBF networks* with additional on-line learning vector quantization (adaptive perceptualization) and obtain very encouraging results as well.

Our test bed is pole balancing with additional noise on the sensory input.

1. Q-Learning

Q-Learning [18] as well as AHC-Learning [16] by learning optimal actions try to minimize the expected cumulative discounted reward associated with each state in a Markovian environment. The function Π mapping a state to the action to be executed in this state is called the policy. More formally, we have a finite set of states $S = \{1, \ldots, n\}$, a finite set of actions $U = \{1, \ldots, m\}$, stationary (unknown) probabilities $p_{ij}(u)$, denoting the probability to get to state j if applying action u in state i, and a reinforcement signal $r(t)$ denoting the immediate reinforcement received for applying action u in state i. The objective is to find an optimal (stationary) policy Π^{opt} minimizing for all states $s \in S$ the evaluation function

$$V_{\Pi}(s) = E_{\Pi}\left[\sum_{t=0}^{\infty} \gamma^t r(t) \,\middle|\, s_0 = s\right], \tag{1}$$

where γ is the so called discount factor. Such a policy Π^{opt} always exists and -knowing all the $p_{ij}(u)$ - can be found by Dynamic Programming techniques [3].
Q-Learning [19] is guaranteed to find the optimal policy without knowledge of the transition probabilities. Here, one starts with an arbitrarily initialized *table* of state-action values $Q(s, u)$ for all $(s, u) \in (S \times U)$. The update equation is

$$\Delta Q(s_{t-1}, u_{t-1}) = \alpha_t err(t) \text{ with} \tag{2}$$

$$err(t) = r(t) + \gamma max\{Q(s_t, a) \,|\, a \in U\} - Q(s_{t-1}, u_{t-1}) \tag{3}$$

and convergence of this learning rule to the optimal state-action values $Q^{opt}(s, u)$ is guaranteed if a) α_t is appropriately decreased to zero with t and b) all admissable state action pairs are backed up infinitely often. The optimal evaluation function is then implied by

$$V^{opt}(s) = max\{Q^{opt}(s, u) \,|\, u \in U\} \tag{4}$$

and the optimal policy by

$$\Pi^{opt}(s) = argmax\{Q^{opt}(s,u) \mid u \in U\} . \quad (5)$$

Both prerequisites for convergence to the optimal policy are, however, unrealistic in most real world domains. A reactive agent, for instance, cannot afford to try each action in each state infinitely often (a)[1], especially if safety demands have to be met (exploration/ exploitation dilemma), and (b) "freezing" the learning constant is highly undesirable as well, if the agent should retain plasticity to cope with a changing environment (stability/ plasticity dilemma). Other problems of Q-Learning are that the representation of the Q-function has to be tabular and that the update rule (2) is inherently sequential. This sequentiality results from the *max*-operator in (3): Changes to the Q-function at the current point of time cannot be used to back up Q-values of former visited states because the policy may have changed meanwhile. On the other hand, for AHC-Learning the use of "traces" through time fully exploits parallelism and has been shown to significantly speed up learning [2].

2. Modified Q-Learning

In order to deal with these shortcomings, we now introduce some modifications to the original learning scheme and demonstrate their effect in pole balancing. For the details of our simulation the reader is referred to section 4.

First of all we shift priority from stability to plasticity and therefore use a *constant learning rate* throughout all the following experiments. This is quite common in Q-Learning, see e.g. [10], [7], [1].

The problem of backing up the costs of all state action pairs infinitely often while nevertheless trying to behave optimally according to the current Q-values is usually addressed by making the policy a stochastic function of the Q-values (e.g. Boltzmann distribution, see [3])[2]. Our experiments with the pole balancer, however, suggested that given a constant learning rate an *optimistic initalization*[3] (see also [8]) of the Q-values together with a *greedy strategy* yield best results (and were thus employed in the following). Hence our modified Q-Learners are deterministic.

The performance of such a (tabular) Q-Learner is depicted in Fig 1. (*Box*-curve).

Two more ingredients were needed to yield the impressive success and learning rates of our second algorithm, *BoxAt*, depicted as the second curve in Fig 1. The first one we call the *principle of piecewise constancy of policy* which - often hidden between the lines- turns out to be a read thread through most successful delayed reinforcement learning applications (e.g. [2], [12] and [11]). Its essence is that policies must not change too frequently so that the learner gets a better chance to evaluate the current policy.

---

1. Without prior knowledge of the transition probabilities $p_{ij}(u)$, however, it is hard to imagine any learner which is guaranteed to find an optimal policy while violating a).

2. For AHC-Learning, interesting exploration and learning strategies have been proposed in e.g. [12], [9]. Here, the system frequently switches between deterministic and stochastic behavior and learning is stopped if a quasi optimal solution has been found.

3. We call an initialization optimistic iff $Q^{init}(s,u) \geq V^{opt}(s)$ for all $s \in S, u \in U$.

We implement this principle by accumulating errors until failure. On failure, the Q(s,u) values are then updated with their accumulated errors. The learning equation thus obtained is

$$\Delta Q^{tot}(s,u) = \begin{cases} \frac{\alpha}{T}\sum_{t=1}^{T} err_t(s,u) & : \text{failure} \\ 0 & : \text{otherwise} \end{cases} \quad (6)$$

where T is the number of time steps from the last until the current failure and

$$err_t(s,u) = \begin{cases} err(t) & : s = s_{t-1}, u = u_{t-1} \\ 0 & : \text{otherwise} \end{cases} \quad (7)$$

is the contribution of the current error to the accumulated error $\Delta Q^{tot}(s,u)$.
The second equally important ingredient is the introduction of activity traces, similar to the eligibility traces of [2]. Here the main idea is to define an error signal not only for the last but also for all previously encountered states and applied actions. For Q-Learning such ideas have been put forward in e.g. [13] with the $Q(\lambda)$ family and [15]. However, error tracing in the latter two is questionable within the Q-Learning framework since policies may change and leave the traces invalid. On the other hand, combining error traces with piecewise constancy of policy yields valid error traces and, moreover, allows to store a trace for each state only (as opposed to one trace for each state action pair). Our tracing mechanism replaces (7) by

$$err_t(s,u) = \gamma^n err(t) \quad \text{with } n = min\{m | s_{t-m} = s\} \quad (8)$$

which can be effectively calculated as

$$err_t(s,u) = a_s(t)\, err(t)\,, \quad (9)$$

with each state's *activity trace* calculated as $a_s(t) = \begin{cases} 1 & : s = s_{t-1} \\ \gamma a_s(t-1) & : \text{otherwise} \end{cases}$ and $a_s(0) = 0$ on a reset.

3. Q-Learning with Dynamic Cell Structures

While *Box* and *BoxAt* presupposed an a priori quantization of the input space we will now attempt to Q-learn with a growing Radial Basis Function network which has to solve the additional problem of adaptive vector quantization of the input space on-line. RBF networks, [14], promise to be particular well suited for Q-Learning. The well known problem of overestimation when using a function approximator for Q-Learning, [17], is largely eliminated by using piecewise constant policies. Our approach is similar to [1], using RBF units of constant width and employing on-line gradient descent for adjusting the output layer and the centres, but includes the *principle of piecewise constancy* and *activity traces* which we have reformulated for normalized DCS networks. While basically an RBF network, a DCS network concurrently learns and utilizes the topology of the input manifold (see [4] for detail). On presentation of a stimulus *s* only the "best matching RBF unit" (bmu) and its topological neighbors, *Nh(bmu)*, are acti-

vated. In the following, let $Nh^+(bmu)$ denote $\{bmu\} \cup Nh(bmu)$. The output of the DCS network than calculates as

$$Q_{DCS}(s,u) = \sum_{j \in Nh^+(bmu)} \frac{q_{ju} rbf(\|s-\mu_j\|)}{\sum_{l \in Nh^+(bmu)} rbf(\|s-\mu_l\|)} \tag{10}$$

The output can be interpreted as a weighted sum of the q_{ju} values attached to each "state neuron". Gradient descent on the squared prediction error $err^2(t)$ yields the learning rule

$$\Delta q_{ju} = -\alpha_q \frac{\partial}{\partial q_{ju}} err^2(t) \text{ and } \Delta \mu_{ij} = -\alpha_\mu \frac{\partial}{\partial \mu_{ij}} err^2(t) \tag{11}$$

and accumulation of gradients for realizing the principle of constant policy:

$$\Delta w^{tot} = \begin{cases} \frac{-\alpha_w}{T} \sum_{t=1}^{T} \frac{\partial}{\partial w} err^2(t) & : \text{failure} \\ 0 & : \text{otherwise} \end{cases} . \tag{12}$$

As in the case of table-based Q-Learning with constant policy the current error *err(t)* can be used to update the output weights q_{ju} for all previously encountered state-action pairs s_{t-n}, u_{t-n}. If unit i has been activated in state s_{t-n} and the controller has decided to take action k_{t-n}, we can derive the partial derivative of $err^2(t)$ with respect to $q_{i,k(t-n)}$ as

$$\frac{\partial}{\partial q_{i,k(t-n)}} err^2(t) = -err(t) \gamma^{n-1} y_i(t-n) . \tag{13}$$

Thus by defining an activity trace for each output weight as

$$a_{ik}(t) = \begin{cases} y_i(t-1) & : i \in Nh^+(bmu(s_{t-1})), k = u_{t-1} \\ \gamma a_{ik}(t-1) & : \text{otherwise} \end{cases} \tag{14}$$

with $a_{ik}(0) = 0$ on a reset, we can replace the right side of (13) by $-a_{ik}(t)\, err(t)$. The balancing performance for this DCS based approach is depicted in Fig 3., *DcsAt* and *Dcs* denoting the DCS based algorithms with and without activity trace. Bearing in mind that these growing networks with a maximum of 25 units had to solve the (extremely non-trivial) adaptive perceptualization problem for the 4d pole balancer in addition to coping with noise these results are quite impressive. Again, activity traces helped to increase performance. The reader should be aware of the fact that no learning constants had to be "frozen". Gradient descent for center adaptation was found to offer a good compromise w.r.t. the stability/plasticity dilemma. Note that networks with a Kohonen type center adaptation rule would have collapsed without freezing if not augmented with some kind of modulation of the learning constant as in eg. [5].

4. Experiments

The kind of pole balancing problem we dealt with is the same problem as attacked in e.g. [1], [2] and [9] but with additional noise of 5% on the sensory input. The controller is a four input (4d) single output MISO system, the output being +10/-10N (Bang-bang

control), and the learner receives a penalty only if either the pole exceeds a certain angle or the card leaves a certain range on the x-axis. On failure, the pole/ card system is reset to the state (0,0,0,0) and a new trial begins. Fig 1. and Fig 3. report the number of balancing time steps versus the number of trials averaged over 20 runs. Similar to [2] our runs consisted of 500000 time steps and if a successful trial had to be interrupted because of exceeded time the remaining trials were assigned the previous number of balancing steps. In order to avoid deceivingly high balancing results runs with more than 10000 balancing steps were only counted with 10000 balancing steps when averaging. Fig 2. and Fig 4. show results of some single runs for *BoxAt* and *DcsAt*. More details, parameter sets and additional experiments will be published elsewhere [6].

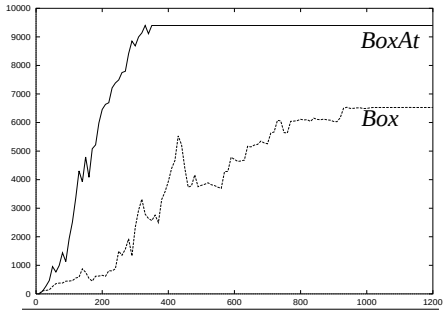

Fig 1. Table Based Q-Learners, averaged

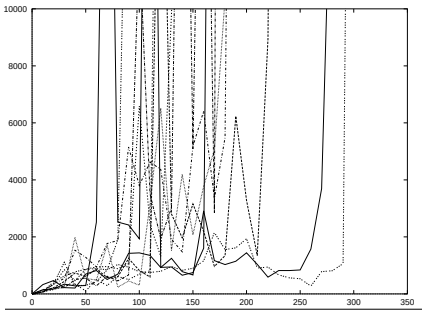

Fig 2. 10 single *BoxAt* runs

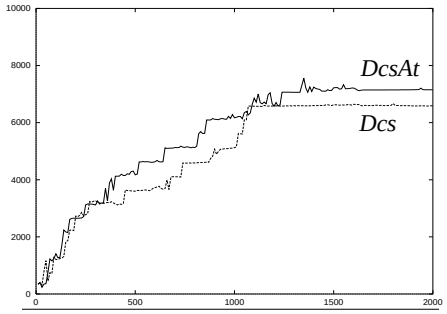

Fig 3. DCS Based Q-Learners, averaged

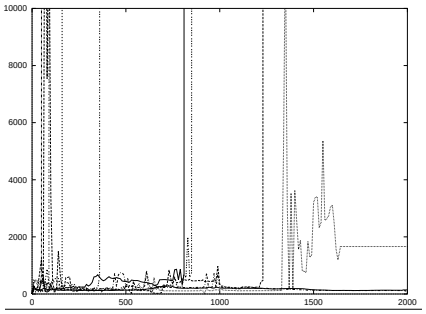

Fig 4. 10 single *DcsAt* runs

5. Summary

Our experiments give further evidence that *piecewise constancy of policy* and *activity traces* are key ingredients for practical success in Q-learning. In combination with a *constant learning rate*, *optimistic initialization* and a *greedy strategy* - our answers to the stability/ plasticity and exploration/ exploitation dilemma - we were able to derive a deterministic algorithm (*BoxAt*) which in spite of its simplicity exhibits impressive balancing performance on an (a priori) quantized input space with additional noise.

Equally important, we have demonstrated that a growing DCS (RBF) network with as few as 25 units (compared to the quantization into 162 boxes) and using the same principles can be very successful as well while solving the additional task of adaptive perceptualization on-line. The algorithms (*BoxAt* and *DcsAt*) compare well to Barto's famous, yet more sophisticated, probabilistic AHC based pole balancer and to Anderson's probabilistic RBF Q-learners, the latter using networks of 30 units and achieving balance after 3300 trials on average.

References

1. C.W. Anderson: Q-Learning with Hidden Units Restarting, NIPS 2, Morgan Kaufman, San Mateo, pp.81-87, 1992.
2. A. Barto and R. Sutton and C. Anderson: Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems, IEEE Trans. on Systems, Man and Cybernetics, Vol. 5, pp.834-846, 1983.
3. Andrew G. Barto and Steven J. Bradtke: Learning to Act using Real-Time Dynamic Programming, Artificial Intelligence, pp. 81-138, 1995.
4. J. Bruske and G. Sommer: Dynamic Cell Structure learns Perfectly Topology Preserving Map, Neural Computation, Vol. 7, No. 4, pp. 845-865, 1995.
5. I. Ahrns, J. Bruske and G. Sommer: On-line Learning with Dynamic Cell Stuctures, ICANN'95, pp. 141-146.
6. J. Bruske and G. Sommer: Practicing Q-Learning, Technical Report, Inst. f. Inf. u. Prakt. Math CAU Kiel, to appear.
7. Bruce L. Digney: Emergent Intelligence in a Distributed Adaptive Control System, Ph.D. thesis, Univ of Saskatchewan, 1994.
8. M. Heger and K. Berns: Risikoloses Reinforcement-Lernen, KI 4/92, pp.26-32.
9. D. Kontoravdis: Efficient Reinforcement Learning Strategies for the Pole Balancing Problem, ICANN 94, pp.659-662.
10. Long-Ji Lin: Reinforcement Learning for Robots Using Neural Networks, Ph.D. Thesis, Carnegie Mellon Univ., 1993.
11. D. Michie and R.A. Chambers: BOXES: An experiment in adaptive control, Machine Intelligence 2, eds. E. Dale and D. Michie, pp.137-152., 1968.
12. J. R. Millan and C. Torras: A Reinforcement Connectionist Approach to Robot Path Finding in Non-Maze-Like Environments, Machine Learning, Nr. 8, pp.363-395,1992.
13. J. Peng, J. and R. J. Williams: Incremental multi-step Q-learning, Proc. of ML94, pp.226-232.
14. T. Poggio and F. Girosi: Networks for Approximation and Learning, Proc. of the IEEE, Vol. 78, Nr. 9, pp.1481-1497,1990.
15. G. Rummery and M. Niranja: On-line Q-Learning using Connectionist Systems, Tech. Rep. CUED/F-INFENG/TR 166, Cambridge Univ. Engeneering Dept., 1994
16. Richard S. Sutton: Temporal Credit Assignment in Reinforcement Learning, Ph.D. Thesis, Univ. of Mass., Amherst, 1984.
17. S.B. Thrun and A. Schwartz: Issues in Using Function Approximation for Reinforcement Learning, Proc. of the Fourth Connectionsit Models Summer School, 1993.
18. C. Watkins: Learning from delayed rewards, Ph.D. Thesis, Univ. of Cambridge, England, 1989.
19. C. Watkins: Q-Learning, Machine Learning, 8, pp.279-292, 1992.