Efficient $\mathcal{Q}$-Learning by Division of Labor

Michael Herrmann
RIKEN, Lab. f. Information Representation
2-1 Hirosawa, Wakoshi 351-01, Japan
michael@sponge.riken.go.jp

Ralf Der
Universität Leipzig, Inst. f. Informatik
Postf. 920, D-04009 Leipzig, Germany
der@informatik.uni-leipzig.d400.de

Abstract: *Q-learning as well as other learning paradigms depend strongly on the representation of the underlying state space. As a special case of the hidden state problem we investigate the effect of a self-organizing discretization of the state space in a simple control problem. We apply the neural gas algorithm with adaptation of learning rate and neighborhood range to a simulated cart-pole problem. The learning parameters are determined by the ambiguity of successful actions inside each cell.*

1 Introduction

$\mathcal{Q}$-learning [1] is an algorithm for dynamic programming that represents an efficient implementation of the reinforcement learning paradigm. It is based on an estimation of future reinforcement conditioned on the previously acquired policy of assigning actions to the states of a controlled system. One may distinguish two ways of implementing $\mathcal{Q}$-learning, either by means of a state-action look-up table or by approximating the $\mathcal{Q}$-function, e.g. using multi-layered neural nets [2]. For continuous state spaces the look-up table refers to a quantization of the states of the system given by either prior knowledge or an adaptive algorithm. Our paper is concerned with the latter approach and tries to reveal and to resolve some of its pitfalls. These are caused by the interference of two types of learning: The adaptation of the partitioner and the reinforcement learning of the controller. The situation is schematically represented in Fig. 1. The determination of quantized states, which are internal states in the full control problem, represents an instance of the *hidden state problem* (cf. [3]). The reinforcement learning in turn relies on the hidden states that comprise cells of the state space partition.

For discrete actions an ideal partition of the (continuous) state space consists of domains with each having a unique optimal action for all states belonging to that domain. Hence, an optimal partition is defined by means of the policy function assigning states to actions. If the partitioning is done by means of a learning vector quantizer, cells are defined by reference vectors in the input space together with a nearest neighbor assignment of input states to reference vectors. The learned distribution of the reference vectors, however, is determined usually by statistical properties of the inputs to the vector quantizer [4], the density of which corresponds to the frequency by which the states of the system are encountered. In particular, one obtains a high resolution (fine grained partitions) around the stable states of the controlled system. In general, the partition obtained in this way will seldom be optimal in the above sense, cf. Fig. 2. The present paper is devoted to introducing a convenient algorithm which is able of on-line self-learning the optimal partitions.

Here, the learning rule for the vector quantizer is based on the neural gas algorithm [5] which allows for cooperativity between the neurons (categories) so that it shares the noise filtering properties of Kohonen's self-organizing feature maps without relying on a prespecified topology. This algorithm appeared to be well suited for representing data from high dimensional input spaces. The reinforcement learning for the policy function of the controller is implemented by the standard $\mathcal{Q}$-learning algorithm introduced by Watkins [1]. The two learning rules are introduced in the next section, the modification leading to optimized partitioning being given in section 3. The fourth section is devoted to a numerical study of the controller, where for reasons of comparability we have chosen the pole-and-cart problem with the standard set of parameters (cf. e.g. [6]). Several further improvements of partitioner-controller systems, in particular avoiding the problem of destabilizing feed-back loops will be discussed briefly in the final section.

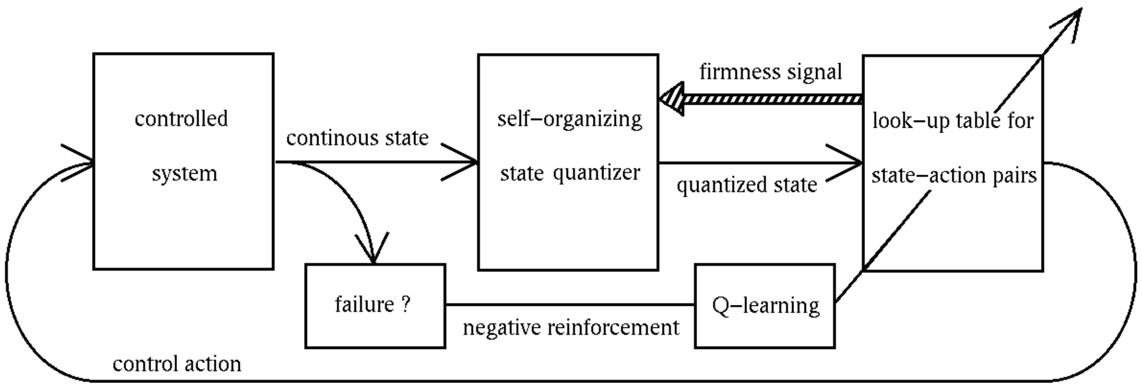

Figure 1: Scheme of the learning problem combining controller and partitioner learning

2 Controller and partitioner learning

$\mathcal{Q}$-learning: We consider a general control task of the following type. Suppose, a system is characterized by the state vector $x \in \mathbf{X}$. Using an appropriate discretization of the generally continuous $\mathbf{X}$ the states can be represented by a finite number of categories $j(t)$. x is to be controlled towards and stabilized inside a target region using a sequence of control actions $\{a(t)\}$, $t = 0, 1, \ldots$. The set of possible actions is assumed to be finite. The $\mathcal{Q}$-learning algorithms determines optimal action sequences $\{a(t)\}$ in correspondence to the categories $\{j(t)\}$ based on reinforcement signals $r(t)$. Optimal actions $a(t)$ are chosen according to

$$a(t) = \text{ argmax } \mathcal{Q}(j(t), a(t)) \tag{1}$$

which together wit the mapping implicitly defines the policy function $a = \pi(x)$. $\mathcal{Q}(j(t), a(t))$ which measures the value of a state-action pair is adjusted according to

$$\Delta\mathcal{Q}\left(j(t), a(t)\right) = \epsilon\left(r(t+1) + \gamma V(j(t+1)) - \mathcal{Q}(j(t), a(t))\right) \tag{2}$$

where

$$V(j(t+1)) = \max_a \mathcal{Q}(j(t+1), a) \tag{3}$$

The reinforcement signal $r(t)$ assumes, e.g. an elevated value if the state has reached the target. It is possible to reward states that are close to the target or to use negative reinforcement when the system has reached a 'failure' state. The example in section 4 relies solely on negative reinforcement.

Adaptive partitioning: Both the control algorithm and the representation of the system's state should be adaptive if (i) the environment changes in time, (ii) no sufficient prior knowledge is available, or (iii) the distribution of state depends intrinsically on the behavior of the system itself. It appears, hence, to be natural to learn the partitions for the $\mathcal{Q}$-algorithm. Unsupervised learning algorithms known from neural networks are well suited for this purpose, where the units (neurons) of the net are associated with the categories. In order to avoid drawbacks of a prescribed topology of reference units we favored the neural gas algorithm [5] rather than a feature map [7]. Whereas the latter is definitely faster in learning the number of units per dimension has to be specified in advance A particularly useful property consists in the determination of the neighborhood by the order of the distances from a given input datum such that correlations among neural units can be controlled specifically in a dimension-independent manner. The basic update rule [5] for category vectors w_j representing a prototype state of category j reads

$$\Delta w_j = -\epsilon h\left(k_j(x, \mathbf{w})\right)(w_j - x). \tag{4}$$

Topological relations between an input datum and the set of reference vectors $\mathbf{w}$ is introduced by the function h depending on the rank k of the distance between w_j and x among all such distances.

$$h_\lambda\left(k_j(x, \mathbf{w})\right) = \exp(-k_j(x, \mathbf{w})/\lambda_j). \tag{5}$$

In particular, for $\lambda \to 0$ only the unit closest to x will be influenced by the update, whereas for larger λ other neurons, too, are attracted towards x. In our formulation λ may vary across units. The resulting asymmetry of interaction allows to transport neurons to regions which are only poorly represented.

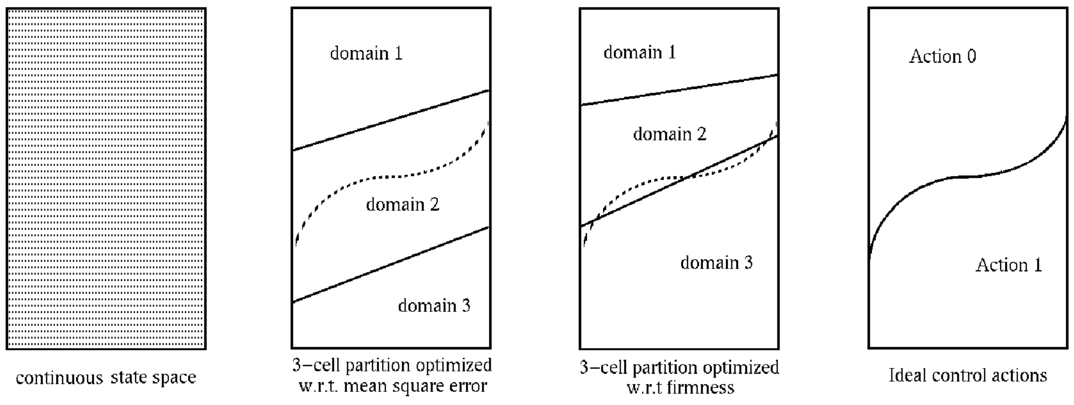

Figure 2: Illustrative example of a three-cell partition. First, optimized with respect to reconstruction error being common in self-organizing maps or, next, optimized with respect to disambiguity of the partition.

3 Partitioning adapted to $\mathcal{Q}$-learning

The maximal value of the $\mathcal{Q}$-function at a given unit decides which action is to be executed at the next time step. In the ideal partitioning defined in the introduction, there is only one optimal action in each domain so that in the converged state the $\mathcal{Q}$-function is stationary under the learning rule 2. Hence, the fluctuations of $\mathcal{Q}$ in each partition of the state space may serve as an indicator for the reliability of this partition. When monitoring $\mathcal{Q}$-values it becomes obvious which domains having already decided for an action. Otherwise, the unit 'requests for assistance' to its neighbors by means of an increased λ-value (cf. Eq. 5). In the case of two possible actions denoted by $a(t) = \pm 1$ we define the *firmness* $f \in \{0, 1\}$ of a unit i as

$$f_i = \| \langle a(t) \rangle_{t, j(t)=i} \| \tag{6}$$

The average $\langle \ldots \rangle$ is a moving average over those time steps where unit i was activated and acting by the greedy strategy (cf. below). Other definitions of the firmness are possible. In particular, in the case of more than two actions instead of (6) entropy measures based on the probability of success for each action are suggestive. The choice (6, 7) on the other hand is not only distinguished by its simplicity, but has also the advantage of allowing for small deviations from maximal firmness which are unavoidable when using a nearest neighbor classifier to approximate the smooth boundaries in the control problem. This is due to a vanishing derivative of λ with respect to f at $f = 1$.

A neuron cannot be *firm* for the 'wrong' action as this is defined in terms of the $\mathcal{Q}$-function. On the other hand, the maximum of the $\mathcal{Q}$-function at unit j will approach the appropriate value as soon as the domain of j is such that a unique correct action exists. For the problem of rarely visited units that may not have achieved a firm state cf. Discussion on controlling the resolution properties of a partitioner. Now, λ can be defined using the length of the mean action vector, in the present case simply

$$\lambda_i \propto (1 - f_i)^2 \tag{7}$$

such that firm units will not disturb their neighbors whereas nonfirm units will effectively attract firm units such that their domain shrinks and reliable control becomes possible.

4 The cart-pole example

The principles described above have been exemplified by the standard task of balancing a pole. (cf. [6]. Details of the problem are given in the Appendix. (For consistency we will call here the set of values of the four degrees of freedom of the system a state space rather than a phase space). Upon receiving a four dimensional input vector the partitioner returns the best-matching unit as the category to which the input vector belongs. The category is submitted to the $\mathcal{Q}$-learning controller together with a reinforcement signal $r(t)$ as the only information the controller receives for advice. $r = -1$ if a failure occurred and $r = 0$ otherwise. The number of categories (units) was 162 (compare [6]), two different actions have been allowed. We want to remark that by exploiting symmetries of the model a solution could be found with much less neurons.

The $\mathcal{Q}$-learning of control actions contains a simple form of explorational behavior by choosing actions according to (1) with probability c and randomly otherwise. c is linearly increased reaching unity at the end of the learning phase of length T_{learn}. During the subsequent test period ($T_{test} = 0.5 T_{learn}$) only the greedy policy, i.e. $c = 1$, is used, but changes in the $\mathcal{Q}$-function are still allowed. The controller is required to achieve control within a maximal time range $T = T_{learn} + T_{test}$ A trial of T time steps counts as a successful one if during at least 10000 time steps in series no failure occurred.

Results are presented in the figures. Fig. 3 demonstrates the speed up of the learning time by a factor of more than 2 due to an appropriate assignment of domains. Because of the success-independent exploration strategy used in all simulations also higher learning times ($T \leq 500000$) did not yield success in each trial. The maximal success rates were 0.68 and 0.8 for the fixed partition and the adaptive partition controller, resp. The *firmness* depicted in Fig. 4 reflects the mean value with respect to active units. The failure of a trial may, however, be caused by the existence undecided units which are rarely visited and, hence, do not have much influence on the average of the firmness.

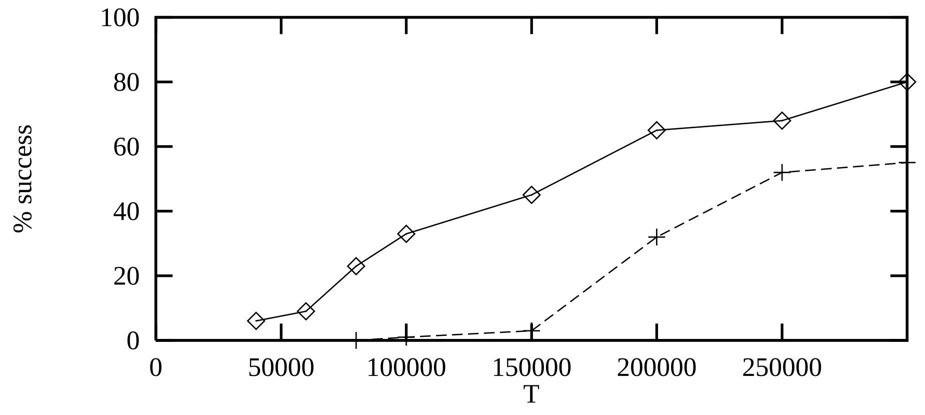

Figure 3: Number of successes in 100 trails at different values of the total learning time for the adaptive partitioner (full line) and a fixed box partition (dashed line).

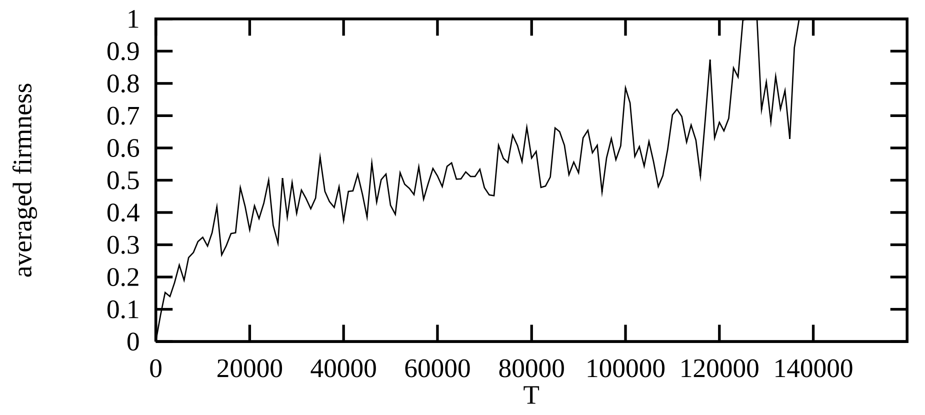

Figure 4: Time course of the firmness averaged over all units for a typical learning trial.

5 Discussion

The proposed learning algorithm for optimized partitioning of the state space of a controlled system rests on tuning the neighborhood width of a neural gas partitioner according to the diversity of the actions proposed by the Q-learner. Also the learning rate of the adaptive partioner has been modified such that it deceases when the partition improves. The computer simulations of the generic cart-pole problem clearly demonstrate that this new algorithm encreases both accuracy and speed of learning of the controller.

The resulting partition deviates from the box partition by a better adaptation to the dynamics of the problem: If position and momentum have opposite signs actually no control action is necessary. Thus, either of the two available actions is critical. This is taken into account by the adaptive partition by a finer resolution in the diagonal region of the state space.

Finally, it should be mentioned that the feedback of output informations to the learning dynamics of the partitioner may well lead to instabilities in early learning. However, these can be avoided by compensating the shift of the reference vectors by conveniently reevaluating the values of the Q-function affected. An appropriate algorithm has shown in preliminary computer simulations to not only avoid the instabilities but also to further increase the speed of Q-learning. Detailed results will for lack of space be presented elsewhere.

Acknowledgements: Part of this work has been done in the LADY project sponsored by the

German BMFT under grant 01 IN 106B/3. MH has received support form the HCM-EC program under grant CHRX-CT93-0097.

References

[1] C. J. C. H. Watkins (1992) $\mathcal{Q}$-learning. *Machine Learning* **8**, p. 279-292.

[2] P. J. Werbos (1990) A menu of designs for reinforcement learning over time. In: W. T. Miller, R. S. Sutton, P. J. Werbos (eds.) *Neural Networks for Control*, MIT Press, Mass. p. 67-95.

[3] S. Das, M. C. Mozer (1994) A unified gradient-descent/clustering architecture for finite state machine induction. In: J. D. Cowan, G. Tesauro, J. Alspector (eds.) *Adv. in Neur. Inform. Proc. Syst. 6*, p. 19-26.

[4] H. Ritter, K. Schulten (1986) On the stationary state of Kohonen's self–organizing sensory mapping. Biol. Cybern. **54**, 99-106.

[5] Th. M. Martinetz, K. Schulten (1991) A 'neural gas' network learns topologies. In: *Proc. ICANN91, Helsinki*, Elsevier Amsterdam.

[6] M. Pendrith (1994) On reinforcement learning of control actions in noisy and non-markovian domains. U. New South Wales, *Techn. rep.* UNSW-CSE-TR-9410.

[7] T. Kohonen (1984) Self-organization and associative memory. Springer, Berlin, Heidelberg, New York.

[8] H.-U. Bauer, R. Der, M. Herrmann (1995) Controlling the magnification factor of self-organizing feature maps. Submitted to *Neur. Comp.*

Appendix: Details of the cart-pole problem

The task (compare [6]) consists in balancing a pole (inverted pendulum) hinged to a cart by applying a binary force $F = \pm 10N$ to the cart. The motion of the cart is confined to a fixed length track such that the task is successfully solved if both the pole remains upright and the cart avoids the ends of the track. The pole is described by the deviation from its vertical position θ and the angular momentum $\dot{\theta}$. The cart position and velocity is denoted by x and $\dot{x}$, resp. The vector $\mathbf{v} = (x, \dot{x}, \theta, \dot{\theta})$ serves as input to the partitioner. The four dimensional system is governed by the following equations:

$$\ddot{\theta} = \frac{mg\sin\theta - \cos\theta(F + m_p l \dot{\theta}^2 \sin\theta)}{(4/3)ml - m_p l \cos^2\theta}, \quad \ddot{x} = \frac{1}{m}\left(F + m_p l(\dot{\theta}\sin\theta - \ddot{\theta}\cos\theta)\right)$$

Parameter values are: length of track $-2.4m < x < 2.4m$, maximal deviation $-0.21 < \theta < 0.21$, mass of cart plus pole $m = 1.1kg$, mass of pole $m_p = 0.1kg$, gravitational acceleration $g = 9.81m/s^2$, effective length of pole $l = 0.5m$. The equations of motion were integrated using the Euler method with time step $\Delta t = 0.02s$.