Q-Learning and Redundancy Reduction in Classifier Systems with Internal State

Antonella Giani, Andrea Sticca, Fabrizio Baiardi, Antonina Starita

Università di Pisa, Dip. di Informatica
Corso Italia 40 56125 Pisa, Italy

Abstract. The Q-Credit Assignment (QCA) is a method, based on Q-learning, for allocating credit to rules in Classifier Systems with internal state. It is more powerful than other proposed methods, because it correctly evaluates shared rules, but it has a large computational cost, due to the Multi-Layer Perceptron (MLP) that stores the evaluation function. We present a method for reducing this cost by reducing redundancy in the input space of the MLP through feature extraction. The experimental results show that the QCA with Redundancy Reduction (QCA-RR) preserves the advantages of the QCA while it significantly reduces both the learning time and the evaluation time after learning.

1 Introduction

Classifier Systems (CSs) [4, 1] are adaptive Reinforcement Learning (RL) systems whose behavior is driven by a set of condition/action rules. This paper addresses credit assignment in CSs that use a message list (ML) as internal state (IS-CSs). While stimulus-response CSs [9] can only solve Markovian Decision Tasks (MDTs), internal messages allow IS-CSs to solve non-Markovian tasks as well. The Q-Credit Assignment (QCA) [2, 3] is a method, based on Q-learning [8], for allocating credit in IS-CSs. It is more powerful than other proposed methods, because it correctly evaluates rules whose application may have different outcomes depending on the context. However, the QCA has a large computational cost, due to the Multi-Layer Perceptron (MLP) [6] that stores the evaluation function. To reduce this cost, we introduce the Q-Credit Assignment with Redundancy Reduction (QCA-RR), that reduces the size of the MLP by reducing redundancy in its input space, through feature extraction.

2 Q-Credit Assignment

The behavior of a CS emerges from the cooperation of several simple rules. Sharing rules among distinct situations allows both compact knowledge representation and generalisation [1, 3]. Commonly used credit assignment methods, such as the Bucket Brigade algorithm [4], allocate a single credit measure to individual rules. This makes them fail in evaluating shared rules whose activation may result in different outcomes depending on the context. The QCA [2, 3] has been devised to overcome this problem. It is based on Q-learning [8], a well known on-line RL algorithm to solve MDTs. Q-learning incrementally estimates the return $Q(x, a)$ of doing an action a in state x, $Q(x, a) = r + \gamma \max_b Q(y, b)$, where r is the immediate reinforcement and γ is the discount factor. The QCA learns Q values of (state,action) pairs (x, a), where x is the contents of the ML and a is a message specified by the action part of a rule that matches x or a part of it[1]. In this way, the QCA estimates the return of activating a rule by taking into account the whole contents of the ML, and it can correctly evaluate rules that have distinct outcomes in different situations. The Q function is approximated through a MLP with one hidden layer and one linear output unit. Given a (state,action) pair, i.e. a ML configuration $\{m_1, \ldots, m_n\}$ and a candidate message m, the corresponding input pattern is the string $m_1, \cdots, m_n, m$. The MLP is trained with the back-propagation algorithm [6] to reduce the Temporal Difference error between two successive evaluations of each pattern.

3 Improving the QCA by reducing redundancy

The main drawback of the QCA is its large computational cost. The evaluation and learning times depend on the number of satisfied rules at each cycle, as well as on the size of the MLP, which grows with the size of the ML and with the length of the messages. A IS-CS assumes that internal messages are as long as detector (input) and effector (output) messages. However, the information to be stored not always needs the same number of bits for each kind of message. Thus, messages may include unuseful or redundant information. A proper feature extraction process can reduce the size of the input patterns and, as a consequence, the size of the MLP, with minimal loss of information.

3.1 Principal component analysis via Hebbian learning

Feature extraction is the process whereby a p-dimensional data space is mapped onto a m-dimensional feature space, $m < p$, so that the data set is represented by a reduced number of features. Given a p-dimensional zero-mean random vector $\mathbf{x}$, let $\mathbf{u}_1, \ldots, \mathbf{u}_p$ be the normalised eigenvectors of the correlation matrix $\mathbf{R} = E[\mathbf{x}\mathbf{x}^T]$, and let $\lambda_1, \ldots, \lambda_p$ be the associated eigenvalues. Principal Component Analysis (PCA) [5] states that, if $\lambda_1, \ldots, \lambda_m$ are the largest m eigenvalues of $\mathbf{R}$, the matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$ maps $\mathbf{x}$ onto a m-dimensional feature vector $\mathbf{y} = \mathbf{x}^T\mathbf{U}$ with a loss of information which is optimal in the mean-square error sense. The $y_i = \mathbf{x}^T\mathbf{u}_i$ are called *principal components*. Sanger [7] shows that a generalised Hebbian algorithm (GHA) can be used to train a feed-forward neural network with p inputs and a single layer of m linear output units, so that the output unit j computes the j-th principal component of the input distribution. This means that the network performs PCA of size m directly on the input patterns, without computing the eigenvectors and the eigenvalues of R.

[1] The QCA uses Q-learning to solve a high-level decision task including both the ML and the original task faced by the CS. The high-level task can be modelled as MDT, even if the original task is non-Markovian, provided that the IS-CS owns the rules to store the appropriate information. For a more complete discussion, see [3]

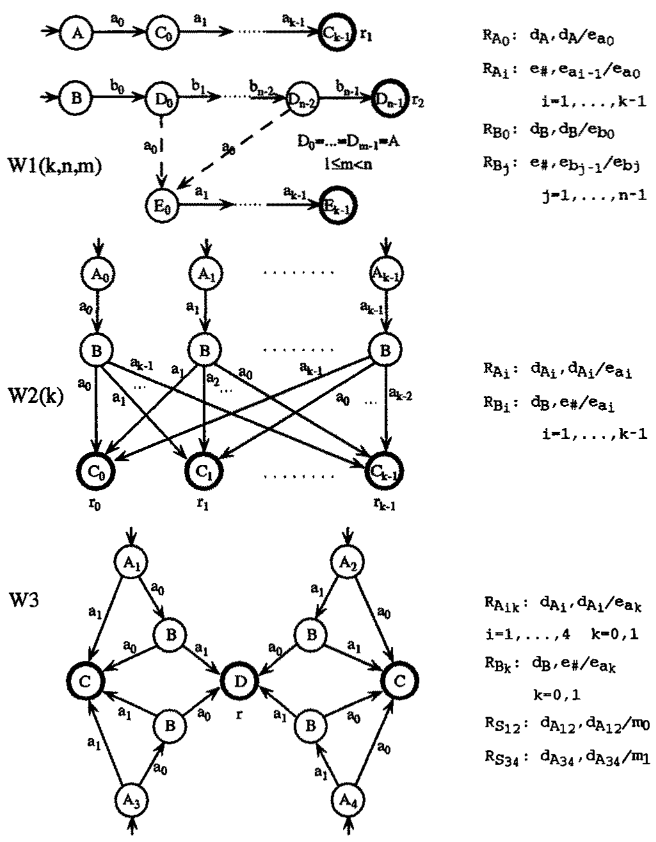
W1(k,n,m)
A
C_0
C_{k-1}
r_1
B
D_0
D_{n-2}
D_{n-1}
r_2
E_0
E_{k-1}
a_0
a_1
a_{k-1}
b_0
b_1
b_{n-2}
b_{n-1}
$D_0 = \ldots = D_{m-1} = A$
$1 \le m < n$
R_{A_0}: $d_A, d_A / e_{a_0}$
R_{A_i}: $e_\#, e_{a_{i-1}} / e_{a_0}$
$i = 1, \ldots, k-1$
R_{B_0}: $d_B, d_B / e_{b_0}$
R_{B_j}: $e_\#, e_{b_{j-1}} / e_{b_j}$
$j = 1, \ldots, n-1$
W2(k)
A_0
A_1
A_{k-1}
B
C_0
C_1
C_{k-1}
r_0
r_1
r_{k-1}
a_{k-2}
R_{A_i}: $d_{A_i}, d_{A_i} / e_{a_i}$
R_{B_i}: $d_B, e_\# / e_{a_i}$
$i = 1, \ldots, k-1$
W3
A_1
A_2
A_3
A_4
B
C
D
r
$R_{A_{ik}}$: $d_{A_i}, d_{A_i} / e_{a_k}$
$i = 1, \ldots, 4 \quad k = 0, 1$
R_{B_k}: $d_B, e_\# / e_{a_k}$
$k = 0, 1$
$R_{S_{12}}$: $d_{A_{12}}, d_{A_{12}} / m_0$
$R_{S_{34}}$: $d_{A_{34}}, d_{A_{34}} / m_1$

Fig. 1. NMFSW worlds and sets of rules used for the experiments. Initial states have a short entering arrow, whereas final states are double circled.

3.2 QCA with redundancy reduction: QCA-RR

The Q-Credit Assignment with Redundancy Reduction (QCA-RR) reduces the number of connections of the MLP by reducing the dimension of the evaluated data space. The p-dimensional (state,action) patterns are pre-processed by an unsupervised neural network with p inputs and m outputs, $m < p$, which computes the m principal components of each pattern, as described in Sect. 3.1. A preliminar exploration phase is needed to determine the training set of the unsupervised network. During the exploration, no learning is performed and the competition among rules is randomly solved. The length of this phase should be tuned accordingly to the difficulty of the task, so that a meaningful sample of the patterns is examined. After exploration, the unsupervised network is trained with the GHA until it converges to a principal component extractor. Then, the MLP can start estimating the Q function on the m-dimensional feature space of the (state,action) space.

4 Experimental Results

The QCA-RR has been tested in Non-Markovian Finite State World (NMFSW) [3], an abstract domain that can model non-Markovian tasks. A world is modelled as an absorbing Markov process, defined by a set of labelled states, a set of actions, and a state-transition probability matrix. Each state may be paired with a reward, detected as reinforcement by the agent. The label of a state is the information that the agent detects about that state. If distinct states have the same label, the world is non-Markovian. The experiments only concern credit assignment. Any execution is run with a fixed set of rules, which includes both rules leading to a reward and wrong ones. The QCA and the QCA-RR are compared in several worlds, with sets of rules that present an increasing amount of rule sharing. The performance measure, P, is the ratio of the reward obtained in a trial to the possible maximum ($0 \leq P \leq 1$), where a trial is a step sequence from an initial state to a final one. P is plotted versus the number of trials and is averaged on the last n trials. The plotted values are averaged on 3 executions, starting with a different seed of the random number generator. Probability values based on the Boltzmann distribution, with decreasing exploration parameter (temperature) T_b, are used to solve both conflicts on effectors and competitions on the ML. Figure 1 shows three different (classes of) worlds, and the corresponding set of rules. When the agent enters a final state, it gets the associated reward, if any, and it is randomly reset to one initial state. d_X is the condition satisfied in any state labelled as 'X', e_a is the effector message that specifies the external action a, and $e_{\#}$ is a general condition that matches any effector message. In $W1(k, n, m)$, the states $D_0, \ldots, D_{m-1}$ ($1 \leq m < n$) are detected as 'A', so that the rule R_{A_0} can be applied in $m + 1$ different situations. When a message e_{b_t} has been posted at the previous step, R_{A_0} leads to zero reward, otherwise it leads to reward r_1 ($r_1/2 > r_2$). In $W2(k)$, each one of the k rules R_{B_i} can be applied in k different situations, with different outcomes: when a state 'B' is detected, if the previous state was 'A_j', then R_{B_i} leads to the state 'C_h' with reward r_h, where $h = (k - j + i) \bmod k$, $0 \leq i, j < k$. In $W3$, when a state 'B' is detected, the most rewarding action depends on the previous state. While a ML of size 2 is sufficient to solve the tasks in $W1(k, n, m)$ and in $W2(k)$, a ML of size 3 is needed in $W3$. Figures. 2(a), 2(b), and 2(c) show the performance of the CS, using QCA and QCA-RR, in the worlds $W1(3, 4, 3)$, $W2(4)$ and $W3$, respectively. For each graph, the value of following parameters is specified: the length l of messages, the ML size, the discount factor γ, and the initial value of the exploration parameter T_b. The learning rates are $\eta = 0.01$ for the MLP and $\beta = 0.001$ for the self-organising network (SON). The QCA-RR, like the QCA, learns correct evaluations (fluctuations in the performance are due to stochastic competitions among messages). In addition, dimensionality reduction of the input space makes the task easier for the MLP, so that it needs a lower number of trials to get a large performance. This is also shown in Table 1, which compares the learning times of the QCA and the QCA-RR, measured in number of trials across the task. Learning time is defined as the time taken by the CS to reach and maintain a high performance level ($P > 0.95$). To highlight the improvement

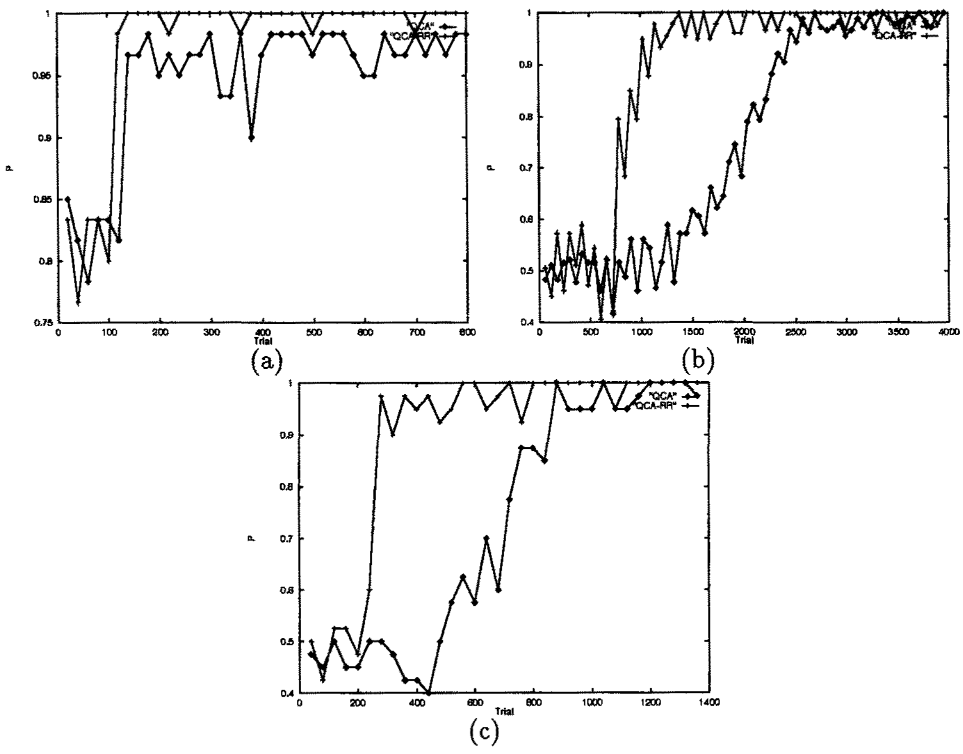

Fig. 2. QCA and QCA-RR performances in $W1(3,4,3)$ (l = 14, ML size=2, $\gamma = 0.6$, $T_b = 100$), in $W2(4)$ (l = 14, ML size=2, $\gamma = 0.7$ in QCA, $\gamma = 0.4$ in QCA-RR, $T_b = 150$), and in $W3$ ($l = 10$, ML size=3, $\gamma = 0.35$, $T_b = 100$).

in execution time, the table also reports the relative CPU time of the QCA-RR, measured with respect to the QCA, taken as 1. Relative CPU time is given both for learning time, which includes the time taken by the SON to converge, and for evaluation time. To measure the evaluation time, we stopped learning after the CS had reached a high stable performance. Then we measured the CPU time the CS takes to perform a fixed number of trials across the task. The improvement in evaluation time shows that the pre-processing overhead of the QCA-RR is fully balanced by the improvement obtained by reducing the size of the MLP. Table 1 also shows the data compression rate, i.e. the dimensionality reduction rate performed by the SON, and the number of weights of the neural networks for each analysed task.

5 Conclusions

We propose a solution to reduce the large computational cost of the QCA through feature extraction. The experimental results in tasks of increasing com-

Table 1. Comparison of QCA and QCA-RR in three worlds.

	$W1(3,4,3)$		$W2(4)$		$W3$	
	QCA	QCA-RR	QCA	QCA-RR	QCA	QCA-RR
Learning time (trials)	160	120	960	450	2640	840
Learning time (CPU)	1	.667	1	.607	1	.307
Evaluation time (CPU)	1	.500	1	.750	1	.555
# weights SON	–	756	–	880	–	840
# weights MLP	2025	381	1849	553	2025	463
# weights tot	2025	1137	1849	1433	2025	1303
Data compression	57%		52%		45%	

plexity show that the QCA-RR, like the QCA, correctly evaluates rules whose activation may have different outcomes depending on the context. In addition, feature extraction dramatically reduces the number of weights of the MLP, thereby reducing both learning time and evaluation time after learning. In particular, the pre-processing overhead of the QCA-RR is fully balanced by the reduction of the size of the MLP. Feature extraction also reduces the difficulty of the task to be learned by the MLP, so that a CS using the QCA-RR reaches a high stable performance in a lower number of trials with respect to a CS using the QCA.

References

1. L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. In J. G. Carbonell, editor, *Machine learning: paradigms and methods*. MIT Press, 1990.
2. A. Giani, F. Baiardi, and A. Starita. Q-learning in evolutionary rule based systems. In *Proceedings of the 3rd Parallel Problem Solving from Nature/ International Conference on Evolutionary Computing*, LNCS 866. Springer-Verlag, 1994.
3. A. Giani, F. Baiardi, and A. Starita. Using Q-learning in classifier systems with internal state and rule sharing. Submitted for pubblication, 1997.
4. J. H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithm applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial inteligence approach*, volume 2. Morgan Kaufmann, 1986.
5. M. Loéve. *Probability Theory*. Van Nostrand, New York, 1963.
6. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representation by error propagation. In D. E. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, volume 1. MIT Press, 1986.
7. T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 12:459–473, 1989.
8. C. J. C. H. Watkins. *Learning with delayed rewards*. PhD thesis, University of Cambridge, England, 1989.
9. S. W. Wilson. ZCS: A zeroth order classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.