Biologically Plausible Multi-Dimensional Reinforcement Learning in Neural Networks

Jaldert O. Rombouts[1], Arjen van Ooyen[2], Pieter R. Roelfsema[2,3], and Sander M. Bohte[1]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{J.O.Rombouts,S.M.Bohte}@cwi.nl
[2] VU University of Amsterdam, Amsterdam, The Netherlands
arjen.van.ooyen@cncr.vu.nl
[3] The Netherlands Institute for Neuroscience, Amsterdam, The Netherlands
p.roelfsema@nin.knaw.nl

Abstract. How does the brain learn to map multi-dimensional sensory inputs to multi-dimensional motor outputs when it can only observe single rewards for the coordinated outputs of the whole network of neurons that make up the brain? We introduce Multi-AGREL, a novel, biologically plausible multi-layer neural network model for multi-dimensional reinforcement learning. We demonstrate that Multi-AGREL can learn non-linear mappings from inputs to multi-dimensional outputs, given only scalar reward feedback. We further show that in Multi-AGREL, the changes in the connection weights follow the gradient that minimizes global prediction error, and that all information required for synaptic plasticity is locally present.

1 Introduction

Imagine learning to play squash. High dimensional sensory inputs give rise to patterns of neuronal activations. These in turn yield a rich motor output, moving legs and arms to hit the ball with the racket. Which actions were useful? Which were not? While humans are able to learn such complex tasks, it is not clear how the brain solves these high dimensional learning problems. In neuronal terms, the problem is one of credit assignment: how should the efficacies of which synapses in the brain change to make useful actions more probable?

Reinforcement Learning (RL) [11] offers a mathematical framework for learning to select optimal actions in Markov Decision Processes. For each possible state of the world an agent tries to predict the value – $\mathcal{Q}$-values – of all possible actions. It then selects the best one with high probability. However, estimating all action-values quickly becomes intractable in high dimensional spaces, the well-known curse of dimensionality [1].

A natural way to solve high-dimensional learning problems is to decompose them. Imagine a task that requires actions by both hands. Instead of estimating values for joint actions (a simultaneous action by left and right hand), as a naive application of RL suggests, we can estimate the values for atomic actions independently. The optimal joint action can then be produced by selecting locally optimal atomic actions. While a variety of modular reinforcement learning systems based on this intuition exist, e.g. [3, 8, 5], ideas on how such approaches could be instantiated in biologically plausible neural networks are still lacking.

Williams' REINFORCE algorithm for training neural networks [13] can be adapted to train neural networks with a modular structure. However, REINFORCE lacks a mechanism to solve the learning problem in an efficient and biologically plausible way. Roelfsema & van Ooyen developed a biologically plausible learning scheme for training multi-layer neural networks by RL, Attention-Gated Reinforcement Learning (AGREL) [6]. However, AGREL can only learn to solve 1-of-n classification tasks, as it is constrained to select only single actions.

Here, we present Multi-AGREL, a first biologically plausible network model for modular reinforcement learning on multiple concurrent atomic actions. We build on the ideas of AGREL to arrive at a neural network scheme that can learn to select optimal simultaneous actions based on a scalar reinforcement signal. For each output dimension, the model has a separate output layer. Units in these output layers try to learn $\mathcal{Q}$-values [11] for their associated atomic actions. Each output layer subsequently has a separate stochastic Winner-Take-All competition (WTA) to select each atomic action. We show that the model learns to minimize prediction errors by gradient descent on a global prediction error, and that all information required for synaptic plasticity is local.

Plasticity in Multi-AGREL is determined by two factors, as in [6]. The first is a globally available neuromodulatory signal that communicates a global prediction error signal. Such a signal could be implemented by a neuromodulator such as dopamine [10, 7]. The second factor is a set of feedback connections from output layers back to the hidden layer that gates plasticity based on the atomic actions that were selected.

We study classification tasks which require linear and non-linear mappings from inputs to multiple simultaneous outputs. All tasks are single step input-output mappings where rewards are directly delivered, so that we can concentrate on solving the spatial credit assignment problem. We empirically demonstrate that Multi-AGREL can learn to solve multi-dimensional non-linear tasks, and we also show that Multi-AGREL significantly outperforms a non-biologically plausible neural network with similar structure trained with REINFORCE [13].

2 Multi-AGREL

Multi-AGREL is modeled as a standard neural network with multiple Winner-Take-All (WTA) output layers o. In this way multiple atomic actions can be selected at the same time. The network predicts $\mathcal{Q}$-values [11] for all atomic actions. In each output layer, a stochastic WTA competition based on the predicted values selects one atomic action to execute. An example network is shown in figure 1. For each joint action the network executes it receives a scalar reward r. The network learns by gradient descent on the global prediction error.

Input patterns x_i are presented to the input layer with N units. Hidden unit activations y_j are computed by a sigmoidal function of the linearly weighted

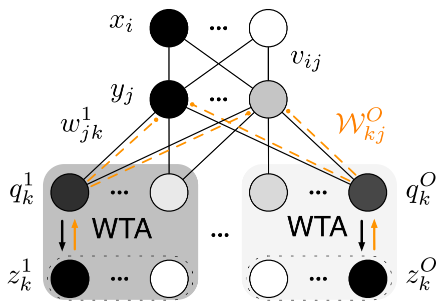

Fig. 1. Multi-AGREL architecture with O separate output layers (shading). Feedback weights shown with dashed lines.

summed input a_j:

$$y_j = \frac{1}{1+\exp(-a_j)} \quad \text{with} \quad a_j = \sum_{i=0}^{N} v_{ij} x_i, \tag{1}$$

where v_{ij} is the synaptic weight between input unit i and hidden unit j and v_{0j} denotes the bias weight. Each output-layer o is fully connected to the hidden layer with M units by connections w^o_{jk}. Each output layer separately computes $\mathcal{Q}$-values q^o_k [11]:

$$q^o_k = \sum_{j=0}^{M} w^o_{jk} y_j, \tag{2}$$

where a bias unit y_0 is included.

With the $\mathcal{Q}$-value estimates, a controller selects one of the atomic actions in each output layer. We implemented a max-Boltzmann [12] controller, which selects the action with the highest estimated $\mathcal{Q}$-value with probability $1-\epsilon$, and otherwise chooses an action with probabilities determined by the Boltzmann distribution:

$$Pr(z^o_k = 1) = \frac{\exp q^o_k}{\sum_{k'} \exp q^o_{k'}}. \tag{3}$$

The winning units K are then set to an activation of 1 and all other units to an activation of 0, $z^o_k = \delta^o_{kK}$ where δ_{kK} is the Kronecker delta function.

After executing an action the network receives a scalar reward r. A prediction error δ is computed as:

$$\delta = r - \sum_{o}^{O} \sum_{k}^{K^o} z^o_k q^o_k, \tag{4}$$

where O is the number of output layers and K^o denotes the number of output units in layer o. We assume that this δ signal is globally available.

Learning. The synaptic updates have two factors, as in AGREL [6]. The first is the global prediction error δ and the second is a Hebbian interaction between feedforward activity and attentional feedback signals. The resulting learning rules are biologically plausible with all information required for the updates available at the synapses [6]. The hidden to output layer synaptic updates are:

$$\Delta w^o_{jk} = \beta y_j z^o_k \delta, \tag{5}$$

where β is the learning rate. The synaptic updates between input and hidden layer are:

$$\Delta v_{ij} = \beta x_i y_j (1 - y_j) \delta \sum_o^O \sum_k^{K^o} \mathcal{W}^o_{kj} z^o_k, \tag{6}$$

where $\mathcal{W}^o_{kj}$ are feedback connections from the output layer to the hidden layer [6]. These feedback connections effectively gate the plasticity of the higher level weights. For convenience, feedback and feedforward weights are assumed to be symmetrical; they can be trained by (5) as in [6].

Multi-AGREL and AGREL differ from Error-Backpropagation (BP) [9] in two important respects. First, attention-gated learning does not require a teaching signal for each output node. Instead it is trained with a global prediction error. Second, it does not require the propagation of specific error signals from output layers to earlier layers [6, 7]. These two elements make attention-gated learning schemes biologically plausible, unlike Error-Backpropagation.

Multi-AGREL minimizes global prediction error. The update local learning rules (5)–(6) update the weights along the gradient on the global prediction error E:

$$E = \frac{1}{2}\left(r - \sum_o \sum_k z^o_k q^o_k\right)^2 = \frac{1}{2}\delta^2. \tag{7}$$

We can derive the updates for weights between the hidden and output layer by the gradient [2]:

$$\frac{\partial E}{\partial w^o_{jk}} = \frac{\partial E}{\partial q^o_k}\frac{\partial q^o_k}{\partial w^o_{jk}} = -\delta z^o_k y_j, \tag{8}$$

where the negative of the gradient matches the direction of the update given in equation (5). The updates for the input to hidden layer weights can be written:

$$\frac{\partial E}{\partial v_{ij}} = \frac{\partial a_j}{\partial v_{ij}}\frac{\partial y_j}{\partial a_j}\sum_o^O \sum_k^{K^o} \frac{\partial E}{\partial q^o_k}\frac{\partial q^o_k}{\partial y_j} = -x_i y_j (1 - y_j)\delta \sum_o^O \sum_k^{K^o} \mathcal{W}^o_{kj} z^o_k, \tag{9}$$

where the rightmost term is the attentional feedback from the output layers via the feedback connections $\mathcal{W}^o_{kj}$. Because of the WTA competition, only one unit per output layer (with $z^o_k = 1$) contributes to the feedback. The negative of the right hand side of (9) matches equation (6).

Together, this derivation shows that by combining attentional feedback signals and a globally available δ signal, Multi-AGREL minimizes prediction error by gradient descent on the $\mathcal{Q}$-value prediction errors, using local updates.

BP-REINFORCE. To compare the Multi-AGREL learning scheme, we implemented the REINFORCE algorithm [13] for a network architecture with multiple WTA output layers. The architecture and activation functions are the same as in Multi-AGREL, except for the output layers, where actions were selected as in equation (3). We applied the REINFORCE updates for output weights [13]:

$$\Delta w^o_{jk} = \beta r(z^o_k - Pr(z^o_k = 1))y_j, \tag{10}$$

The hidden layer weights were updated by Error-Backpropagation [9, 13].

3 Experiments

Tasks To demonstrate the performance of Multi-AGREL we implemented a set of binary tasks with increasing difficulty. The key aspects that we want to illuminate are that Multi-AGREL can deal with tasks that require non-linear transformations and that it scales well to multiple output layers. The tasks were constructed by concatenating different base tasks:

Linear The input consisted of two binary digits, of which one was randomly set to 1. The output was required to be the same as the input pattern. We provided the network with two hidden units for each linear task component.
XOR A version of the non-linear exclusive-OR problem. Two binary inputs need to be mapped to a 'match' signal if both inputs have the same value, and to a 'non-match' signal if they have different values. The output layer had two units, one coding for 'match' and the other for 'non-match'. We provided the network with two hidden units for each XOR task component.
Counting A set of N binary inputs is presented to the network. The network has to count the number of input units with activation 1 and output this number in binary form. We provided an architecture with $\lceil \log_2 N \rceil + 1$ output layers, with two units each. We gave the network $\lceil \log_2 N \rceil + 2$ hidden units.

We evaluated the learning scheme on seven different tasks: Linear-Linear (LL), XOR-Linear (XL), XOR-XOR (XX), 3XOR (3X), 4XOR (4X) and the counting task with 8 (C8) and 32 (C32) inputs. On each trial the input for subtasks was selected independent of the input for the other subtasks. For instance, in the 3X task six random binary inputs were presented to the network, leading to a set of 2^6 possible input patterns. For each output-layer the network received a reward of 1 if it was correct and 0 otherwise. The network only observed the total reward obtained.

Details on training For all non-counting tasks, we trained networks for at most $250,000$ random pattern presentations, or until convergence. Convergence was determined by keeping track of the rewards obtained in the last 200 trials. If the average amount of reward was at least 90% of the total possible reward, the network was said to have reached convergence. For the counting tasks, we set the maximal amount of training trials to $1,000,000$. For all results reported here, we trained 100 networks with random initializations of the synaptic weights. Weights were sampled from a uniform distribution with range $[-0.25, 0.25]$. The

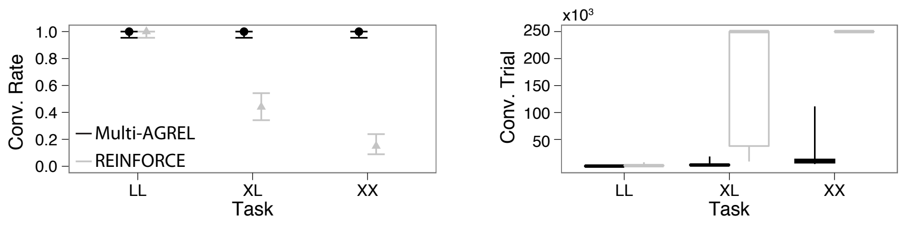

Fig. 2. Performance of Multi-AGREL (black) compared to REINFORCE (grey) on dual tasks, for best parameters found. *Left,* Convergence rates for 100 simulations in each condition. Error bars indicate 95% confidence intervals. *Right,* Box plots show (from bottom to top) minimal convergence trial, Q1 (lower quartile), Q2 (median, *heavy line*), Q3 (upper quartile) and maximal convergence trial. In REINFORCE, learning rates for each layer were individually optimized for best performance.

exploration rate ϵ was set to 0.025. We computed the convergence rate (proportion of 100 networks that reached criterion) and 95% confidence intervals. For convergence times, we report Q1 (lower quartile), Q2 (median) and Q3 (upper quartile) plus minimal and maximal convergence trials. Non-converged networks were assigned the maximal number of pattern presentations.

Performance We compared Multi-AGREL to REINFORCE for the dual tasks (LL, XL and XX), as shown in fig. 2. While performance for the linear task is very comparable, multi-AGREL significantly outperforms REINFORCE for tasks with a non-linear component. Substantial effort was made to find optimal parameters for both algorithms. We evaluated the algorithms with learning rates (β) of $0.01, 0.05, 0.10$, $0.20, 0.30, 0.40$; shown are the results for the best learning rate (highest convergence rate, fastest median convergence time) for both algorithms. For both algorithms 0.4 was the optimal rate for the LL and XL tasks. For the XX task the optimal rates were 0.30 and 0.05 for multi-AGREL and REINFORCE respectively. Due to the meagre performance of REINFORCE on these tasks, we do not show further results for this algorithm.

We further investigated the performance of Multi-AGREL on the harder 3X and 4X tasks (fig. 3a). Here it also showed a robust performance. For the more difficult 4X task, the network worked best with a learning rate of 0.01, but needed more hidden units. We also evaluated Multi-AGREL on the counting tasks (fig. 3b). These were significantly harder to learn than the 4X task, but the algorithm could learn to solve both tasks. Note that the model had to train six disjoint WTA output layers in the C32 task.

Multi-AGREL also performs well in settings with dependent rewards (results not shown). With dependent rewards, the network only receives reward if all atomic actions are simultaneously correct. Performance on the dual tasks is very comparable to that shown in figure 2. However, the network can not escape from the curse of dimensionality [1]. The networks need significantly more trials to learn the 3X and 4X tasks with dependent rewards, but this can not be avoided by any method that learns by trial-and-error. Multi-AGREL is also robust against an unequal distribution of rewards over the modules (results not

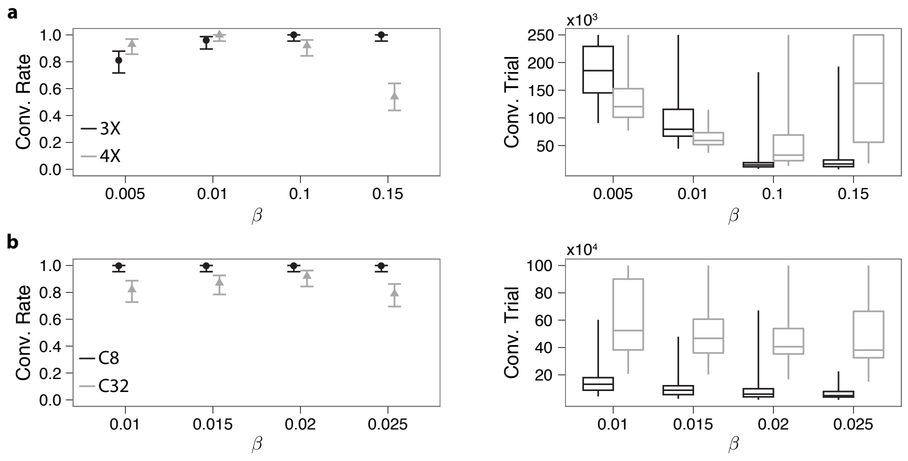

Fig. 3. Scaling performance. Conventions as in fig. 2. *a,* Performance of Multi-AGREL on 3X (black) and 4X (grey) tasks. *b,* Performance of Multi-AGREL on C8 (black) and C32 (grey) tasks. Note that the learning rates (abscissa) and maximal training times (ordinate) differ from those in *a*.

shown). This means that Multi-AGREL also learns effectively in scenarios where different tasks are rewarded differently.

4 Discussion

We have developed a biologically plausible neural network model that can solve a variety of difficult multi-dimensional output problems. Our work builds upon a previous model, AGREL [6], that could learn to solve single-dimensional output problems. Compared to AGREL, the controller in Multi-AGREL separates the action-selection policy from the value-estimation, allowing for the independent selection of atomic actions and the computation of a joints reward estimation. Multi-AGREL solves the spatial credit assignment problem by a combination of feedback signals and a globally released neuromodulatory signal which encodes a global prediction error. The feedback signals encode which atomic actions were selected, and constrain synaptic plasticity to those synapses that were involved in the selected joint action.

Compared to REINFORCE, Multi-AGREL exhibits much better convergence for tasks that contain non-linear components, even when REINFORCE uses the biologically implausible Error Backpropagation algorithm to update the hidden layer weights. A key difference is that REINFORCE updates the policies for all atomic actions after each decision, and not only those that were actually selected. Updating in this way may destroy the correct policy that was stored in the synapses for the non-executed atomic actions.

A distinguishing factor in our model is the idea that multiple unrelated atomic actions can be active at the same time. Most other works assume that there is a single winning action, with all atomic actions mapping to the same output space [8, 5]. It is plausible that both types of solutions are used in the brain. Modules competing for the control of a single effector could use a shared action space model, and modules controlling independent effectors could use a model like Multi-AGREL. Interestingly, recent experimental work has investigated whether humans could learn to simultaneously solve two independently rewarded tasks with two different hands [4]. They show that the results are significantly more consistent with a reinforcement learning model that modularizes the two learning problems than one that learns action values over joint actions. This result gives experimental support for the idea that multiple simultaneous actions are indeed learned by separate modules.

Multi-AGREL is able to learn mappings for co-activated output modules based on a single globally available reward prediction error. This is not a trivial result, as it is not obvious that a global reward prediction error signal combined with local feedback is powerful enough to correctly solve the spatial credit assignment problem. As is mentioned in [4], module specific prediction errors would be best for training separate output modules. However, the dopamine signal found in experiments seems to be unitary throughout the brain [10] (but see the discussion in [4]). Our model and simulations provide evidence that multiple modules can be trained with a unitary reward prediction signal.

Acknowledgments. JOR is supported by NWO grant 612.066.826.

References

1. Bellman, R.E.: Dynamic Programming. Princeton University Press (1957)
2. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, USA (1995)
3. Chang, Y., Ho, T., Kaelbling, L.: All learning is local: Multi-agent learning in global reward games. In: NIPS. vol. 16 (2004)
4. Gershman, S.J., Pesaran, B., Daw, N.D.: Human reinforcement learning subdivides structured action spaces by learning effector-specific values. J. Neurosci. 29(43), 13524–31 (2009)
5. Ring, M., Schaul, T., Schmidhuber, J.: The Two-Dimensional Organization of Behavior. In: IEEE ICDL. pp. 1–8 (2011)
6. Roelfsema, P.R., van Ooyen, A.: Attention-Gated Reinforcement Learning of Internal Representations for Classification. Neural Comput. 2214(17), 2176–2214 (2005)
7. Roelfsema, P.R., van Ooyen, A., Watanabe, T.: Perceptual learning rules based on reinforcers and attention. Trends cogn. sci. 14(2), 64–71 (2010)
8. Rothkopf, C.A., Ballard, D.H.: Credit Assignment in Multiple Goal Embodied Visuomotor Behavior. Front. Psych. 1, 1–13 (2010)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1 (1985)
10. Schultz, W., Dayan, P., Montague, P.R.: A neural substrate of prediction and reward. Science 275(5306), 1593–9 (1997)
11. Sutton, R.S., Barto, A.G.: Reinforcement learning. MIT Press, Cambridge, MA (1998)
12. Wiering, M.: Explorations in Efficient Reinforcement Learning. Ph.D. thesis, University of Amsterdam (1999)
13. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8(3-4), 229–256 (1992)