Goal-Directed Feature Learning

Cornelius Weber and Jochen Triesch

***Abstract*—Only a subset of available sensory information is useful for decision making. Classical models of the brain's sensory system, such as generative models, consider all elements of the sensory stimuli. However, only the action-relevant components of stimuli need to reach the motor control and decision making structures in the brain. To learn these action-relevant stimuli, the part of the sensory system that feeds into a motor control circuit needs some kind of relevance feedback. We propose a simple network model consisting of a feature learning (sensory) layer that feeds into a reinforcement learning (action) layer. Feedback is established by the reinforcement learner's temporal difference (delta) term modulating an otherwise Hebbian-like learning rule of the feature learner. Under this influence, the feature learning network only learns the relevant features of the stimuli, i.e. those features on which goal-directed actions are to be based. With the input preprocessed in this manner, the reinforcement learner performs well in delayed reward tasks. The learning rule approximates an energy function's gradient descent. The model presents a link between reinforcement learning and unsupervised learning and may help to explain how the basal ganglia receive selective cortical input.**

I. Introduction

THROUGHOUT life, we learn to perform actions, to recognize objects and scenes, and, importantly, to ignore irrelevant information. As we walk, unconsciously, we combine features from vestibular, tactile and visual systems that inform us about our posture to keep the right balance. Our balance system ignores irrelevant visual features such as colors. But in addition to features from the vestibular system, it also takes into consideration relevant visual features such as the horizon that provides information on the own orientation in space. Action selection is generally based on selected sensory inputs, as opposed to all available inputs. For this to happen, sensory representations need to adjust to tasks.

Evidence for long-term changes of sensori-motor neural representations has been obtained during habit learning in the rat striatum [1]. The striatum receives direct cortical input and is part of the basal ganglia. Doya [2] proposed that unsupervised learning happens in the cortex and reinforcement learning in the basal ganglia. Accordingly, the cortex preprocesses data to yield a representation that is suitable for reinforcement learning by the basal ganglia [3].

However, it remains open how the striatum reads the relevant inputs from the cortex, i.e. which features are read from the cortical activation pattern that are relevant for selecting actions and obtaining rewards.

Cornelius Weber and Jochen Triesch are with the Frankfurt Institute for Advanced Studies (FIAS), Johann Wolfgang Goethe University, Frankfurt am Main, Germany (email: {c.weber, j.triesch}@fias.uni-frankfurt.de).

This work was supported by the German Federal Ministry of Education and Research within the "Bernstein Focus: Neurotechnology" through research grant 01GQ0840, by EU projects "PLICON" and "IM-CLeVeR", and by the Hertie Foundation..

Sensory representations can be influenced by task demands. In the classical view of the sensory cortex, neurons such as edge detectors in visual cortex V1, adapt merely to the statistics of the incoming data. However, this limited view is challenged since reward, such as a few drops of water, can influence neuronal responses in rat V1 [4]. Even in an experiment without such a reward, if and only if certain stimuli are important for the decision about an action, the corresponding neurons' tuning curves in monkey V1 increase their slope [5]. Such learning effects, despite being observed in the cortex, suggest an influence of reinforcement learning.

So, how do the basal ganglia receive their cortical input? Models that contain cortico-striatal mappings [6], [7], [8], [9] address mainly the functional integration of these structures, rather than an integrated account of learning. Biological evidence suggests a δ-modulated learning rule for cortico-striatal synapses: synapses between the cortex and the striatum are potentiated when the substantia nigra is stimulated [10]. The substantia nigra supplies the striatum with dopamine that is hypothesised to mediate a reward prediction error (below, the δ-signal) in reward seeking tasks.

A. Reinforcement Learning (RL) Models

In the reinforcement learning literature, it is often assumed that incoming data is in the form of a suitable state space representation. For example, hierarchical reinforcement learning models such as the options framework (see [11] for a review) re-organize such a state/action space into tree structures to allow for more efficient decisions.

Another class of models tackles the problem of having more possible states than can be implemented by a lookup table. For example, TD-Gammon is a 3-layer network in which a relatively small number of input units encodes a large space by a distributed code [12], [13]. The weights are trained via a backpropagation algorithm such that the output encodes the state's value. The learning rule is non-local, and the network has shortcomings for biological interpretation, since the representations of the middle layer(s) are not easily interpretable.

Other algorithms that organize the input space make specific assumptions about representations in that space. For example, the Parti-game algorithm [14] and Reinforcement Learning of Visual Classes [15] divide a continuous space and assume a topology, so that when a state is being split, useful new states are created that are similar to the divided state. Instance-based methods such as the U-Tree algorithm [16] record all raw experiences, which may not be computationally permissive and may not generalize well to realistic situations.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

Feature learning is often treated as unsupervised learning. Comparatively little work has addressed how neuronal networks can learn optimized features for a reinforcement learner.

B. Embedding RL Models

Other approaches to a more integrative view of brain function originate from recurrent network models or models of unsupervised learning. Such models are furnished with reward-dependent signals that mimic the effect of dopamine to include reinforcement learning. For example, when spike-timing dependent plasticity (STDP) is modulated by a reward-dependent signal, a recurrent network can solve distal reward problems [17], [18], [19], [20].

RL ideas are also incorporated into other network types. A feedforward model of a cortico-striatal mapping extends unsupervised learning and explains various response properties of striatal neurons that receive visual input [21]. In that model, a dopamine signal modulates neural activations, and thereby learning, when saccades are being made to a rewarded position. Similarly, the attention-gated reinforcement learning model (AGREL) [22] can learn features in the context of 1-of-n classification tasks. A δ signal modulates learning directly in the case of correct trials. As a model of vergence eye movements, AGREL learns disparity-tuned visual cells [23]. However, these feedforward models do not deal with delayed rewards.

C. Model Idea

Our starting point extends the assignment of a modulatory signal within a feedforward network. Instead of assigning it only when a reward is given, we take advantage of a RL algorithm and assign a δ signal to modulate learning globally and throughout extended phases of learning.

In reinforcement learning (for an introduction, see [24]), a reward is given only at the end of a successful trial, i.e. when an agent that started at a random state reaches a goal state. In the simplest models, during learning, i.e. over many repeated trials, a value function builds up (encoded in the weights) that increases toward states that are more proximal to the goal. A scalar δ encodes the prediction error between estimates of neighboring state values. The δ is used to modulate learning of the action weights that encode both, value function and action strategy. This addresses the temporal credit assignment problem and allows RL to deal with delayed rewards.

We use the same δ to modulate an otherwise Hebbian learning of a winner-take-all feature layer that feeds into the action layer, building on a suggestion in [25]. The δ value is large only when an action leads to a state that is better than predicted, and can become negative during random actions. Hence, this learning signal tends to be positive for surprisingly reward predicting features, which characterizes the important learning phases, and is small in average for less relevant features.

The winner-take-all characteristics of the feature layer (cf. self-organizing maps, SOM) lends itself to preprocess raw data for a reinforcement learner, because its output is characterized by a single active unit. Standard RL networks assume that on their input layer, a single active unit describes the state of the agent. In existing models that involve such preprocessing of the sensory data [26], [27], [28], [29], [30], the SOM units represent the entire input space, influenced only by the density of the input data. Unlike these models, our proposed δ-modulated winner-take-all network considers goal-relevance of sensory input dimensions, and learns to neglect irrelevant parts of the input.

II. A Heuristic Local Learning Rule

The model consists of an input layer holding a sensory vector I. A hidden feature layer learns features of the input and encodes a state vector s, of which only one unit is active at a time. An output action layer with activation vector a represents the currently selected action by one activated unit. The network is feedforward with full connectivity between adjacent layers, as shown in Fig. 1.

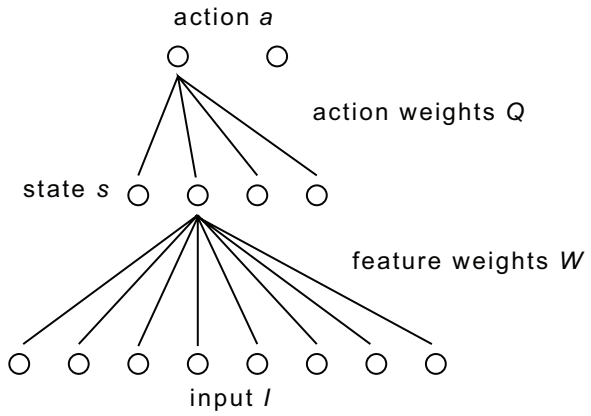

Fig. 1. Feed-forward model architecture drawn schematically. From each layer to the next, only the connections to one recipient unit are shown.

The activation and learning algorithm is given in Table I. Listed are the processing steps for one trial, i.e. one sequence of actions that lasts until a reward $r = 1$ is obtained in the goal state, while elsewise $r = 0$. At the beginning of each trial, the agent is placed at a new random position.

The algorithm inherits the top-level structure of the SARSA algorithm [24]. Hence, Table I resembles a description of SARSA [3]; only steps (1) and (6) are extended to involve the feature weights.

The procedure is as follows. In step (0), the agent performs an action and reads the new sensor values I'. From these values, in (1) the (internal) state representation s' is obtained via the feature weights W and a winner-take-all operation. In steps (2) and (3), an action is chosen stochastically using the action weights Q. Step (4) computes the value v' of the new state based on the next possible actions and the correspondingly used weights Q. Step (5) computes the prediction error δ between the time-discounted current value (discount factor $\gamma = 0.9$) and the earlier value v, considering also the reward r, if delivered ($r = 0$, else).

Initialization	
(i0) Set agent to random position. Get sensory input I	
(i1) Winning feature unit	$m = \text{argmax}_j \sum_n W_{jn} I_n$
Compute old state	$s_j = 1$, if $j = m$, else 0
(i2) Action unit input	$h_i = \sum_l Q_{il} s_l = Q_{im}$
(i3) Softmax action selection	$\texttt{Prob}(a_i = 1) = \frac{e^{2h_i}}{\sum_k e^{2h_k}}$
(i4) Value	$v = \sum_{k,l} Q_{kl} a_k s_l$
Repeat until trial ends successfully, i.e. until $r = 1$	
(0) Act. Observe new sensory input I' and reward r	
(1) Winning feature unit	$m = \text{argmax}_j \sum_n W_{jn} I'_n$
Compute new state	$s'_j = 1$, if $j = m$, else 0
(2) Action unit input	$h_i = \sum_l Q_{il} s'_l = Q_{im}$
(3) Softmax action selection	$\texttt{Prob}(a'_i = 1) = \frac{e^{2h_i}}{\sum_k e^{2h_k}}$
(4) Value	$v' = \sum_{k,l} Q_{kl} a'_k s'_l$
(5) Prediction error	$\delta = r + \gamma v' - v$
(6a) Actor weight update	$\Delta Q_{ij} \propto \delta\, a_i s_j$
(6b) Feature weight update	$\Delta W_{jn} \propto \delta\, s_j\, I_n$
If $r = 1$, update also	$\Delta W_{jn} \propto \delta\, s'_j\, I'_n$
(6c) Normalize and rectify	$\vec{W}_j = [\vec{W}_j / \|\vec{W}_j\|]^+$
(7) New becomes old	$I, s, a, v \leftarrow I', s', a', v'$

TABLE I
SARSA ALGORITHM EXTENDED WITH HEBB-LIKE FEATURE LEARNING

In step (6), the action layer weights Q are trained, essentially by a δ-modulated Hebbian rule with state s and action a as pre- and post-synaptic values, as in normal SARSA. The feature layer weights are also updated by a δ-modulated Hebbian rule, with pre-synaptic activations I and post-synaptic activations s.

As a variation to the Hebb-like rule, we also tried replacing the weight updates in step (6b) of Table I by the following "Kohonen" update:

$$\begin{aligned} \Delta W_{jn} &\propto \delta\, s_j (I_n - W_{jn}) \\ \Delta W_{jn} &\propto \delta\, s'_j (I'_n - W_{jn}) \qquad \text{(if rewarded)} \end{aligned}$$

in which the pre-synaptic activation term is replaced by the difference between the input I and the current weight vector W_j of the winning unit. This is as in traditional Kohonen-like SOMs [31] without neighbourhood interaction. Novel is the δ term.

The weights W are normalized to length 1 for each feature unit, which ensures that a unit that wins for one data point will not also win for all others. Since all data values are non-negative, the weights W are rectified to be positive.

The feature layer performs competitive learning, similar to K-means clustering (K being the number of feature units), or to a SOM in which the neighborhood interaction is set to zero. The δ term makes sure that the feature layer learns only when there is learning progress, that is, when currently relevant data are encountered. Since unimportant components of the data are not correlated with the learning progress, on average, they will not contribute to learning.

In addition to the steps mentioned, we impose a weight decay on the action weights Q as

$$\Delta Q_{ij} \longleftarrow \Delta Q_{ij} - \eta Q_{ij}^3$$

with a small constant $\eta = 0.00003$. This term does not influence initial learning, but prevents large weights from emerging after very long training durations. These would limit the network's potential to unlearn and to re-adapt to new tasks in our re-learning experiments. Other schemes like noise on the parameters [32], here the Q, may also be possible.

III. A Gradient Descent Learning Rule

For a better understanding, we derive the gradient of an energy function, which will lead to a non-local learning rule. We will later see that the learning rule in step (6b) of Table I may be regarded as a somewhat crude approximation to such a gradient descent, by simply omitting the non-local terms.

We recall that the action weights Q estimate values v of actions a in states s. These values approximate a value function that increases toward the rewarded state, the goal of the agent's actions. In our network, the states are functions of the inputs I via the feature weights W. Let $\Theta = (Q, W)$ denote the network parameters. Taking the stance of [24] (Chapter 8), the $v = v(\Theta)$ values shall be updated to minimize a mean squared error

$$E = \frac{1}{2} \sum_{s,a} P^\Pi(s,a) (V^\Pi(s,a) - v(s,a))^2$$

where $V^\Pi(s,a)$ is the "true" value given the current action policy Π (which, too, depends on the network parameters Θ). The difference term that replaces the old estimate v using a better estimate v' obtained from the next time step is

$$V^\Pi - v = r + \gamma v' - v = \delta$$

which is step (5) in Table I. The probability distribution $P^\Pi(s,a)$ is the *on-policy* distribution of state-action pairs that the agent follows. In on-line learning, this distribution is implicit in the selection of data, selected according to step (3) in Table I and retrieved via steps (0) through (2). The on-line update of network parameters becomes

$$\Delta\Theta \propto -\frac{\partial E}{\partial \Theta} = (V^\Pi - v) \frac{\partial}{\partial \Theta} v$$

Using $v = \sum_{k,l} Q_{kl} a_k s_l$, as given in step (4) in Table I, we obtain the action weight update

$$\Delta Q_{ij} \propto -\frac{\partial E}{\partial Q_{ij}} = \delta\, a_i\, s_j$$

which is step (6a) of Table I. A derivative to the feature weights is not possible using a winner-take-all mechanism (step (1) in Table I), which is not a differentiable function. A close approximation is a softmax function, which becomes winner-take-all-like with a sufficiently large parameter β (we will use $\beta = 100$ in our simulations):

$$s_j = g(h_j) = \frac{e^{\beta h_j}}{\sum_k e^{\beta h_k}}, \quad \text{where } h_j = \sum_n W_{jn} I_n$$

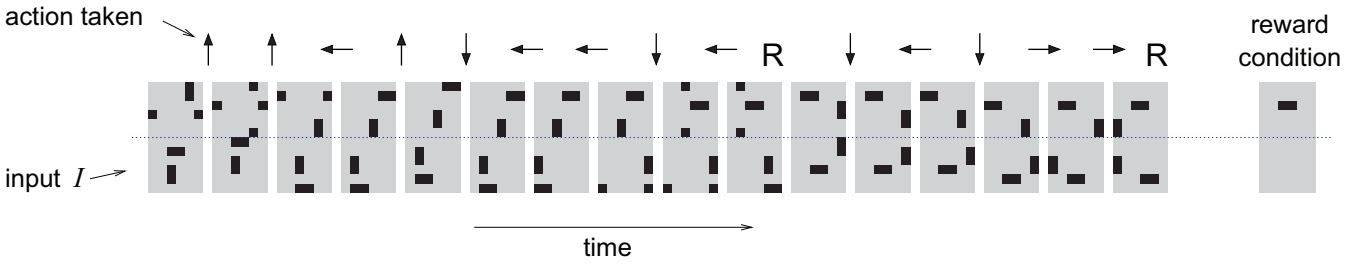

Fig. 2. Data. The sensory input I at any time instant consists of four features, two horizontal and two vertical bars, one in each half of the input area (separated by the dotted line). Note periodic boundary conditions. An action moves the features into one of four directions from one time step to the next, as indicated by the arrows (a feature may fail to move with a certain probability). A reward R is given when the "rewarded feature" (here a horizontal bar in the upper half) appears at a specific location (shown as "reward condition"). The network must learn the relevant features (here, the upper horizontal bars) and an action strategy. After reward is given, all features are set to a new random position.

Considering that W_{jn} influences not only s_j of feature unit j but the activations s_k of all feature layer units, we have

$$\begin{aligned}\Delta W_{jn} &\propto -\frac{\partial E}{\partial W_{jn}} = -\sum_k \frac{\partial E}{\partial s_k}\frac{\partial s_k}{\partial h_j}\frac{\partial h_j}{\partial W_{jn}} \\ &= \delta \sum_k Q_{ik}\frac{\partial s_k}{\partial h_j} I_n\end{aligned}$$

where we have assumed that action unit i was the activated one. Using the following identities for the softmax function [33]

$$\frac{\partial s_j}{\partial h_j} = s_j(1-s_j) \quad \text{and} \quad \frac{\partial s_j}{\partial h_{k,k\neq j}} = -s_k s_j$$

we obtain

$$\begin{aligned}\Delta W_{jn} &\propto \delta Q_{ij} s_j(1-s_j) I_n \; - \; \delta \sum_{k,k\neq j} Q_{ik} s_k \, s_j I_n \\ &= \delta Q_{ij} s_j I_n \; - \; \delta \sum_k Q_{ik} s_k \, s_j I_n\end{aligned}$$

The first term is similar to the feature weight update in step (6b) of Table I, except for the factor Q_{ij} that arises through backpropagation and that denotes how strong state neuron j contributes to the output.

The second term represents a competitive decay term that has a larger suppressive effect if strong activations s_k in the feature layer are paired to large weights Q_{ik}. If exactly one feature unit is active, $s_j = 1$, and for all others $s_{k,k\neq j} = 0$, i.e. if a clear winner is found, then the first and the second term cancel, so learning has converged.

Note that this learning rule is not local, first, because of the weight Q_{ij} to an action unit and, second, because of the second term summing over activations on the feature layer.

In our experiments, we will apply weight normalization and rectification to this "Softmax" rule as described in Section II. We will also test this without these weight constraints ("Softmax no constr.").

IV. Scenario

We devised artificial data that would require a model to learn select feature detectors for learning an action strategy. At each point in time, four features are shown to the model, each being a short bar consisting of just two pixels. Two of these features are in the upper half of the input, two in the lower half; in each half, one bar is vertically oriented and the other horizontally. Fig. 2 shows example data.

When an action is made, all features make a small transformation: they move up, right, down or left, depending on one of four actions chosen. There is a 20% probability for each feature to fail doing this transformation, assuring that over time the positions of the features with respect to each other are independent. Both halves of the input have periodic boundary conditions, i.e. a feature that leaves on one side enters the opposite side; the two pixels of a bar may then be on opposite sides of each half of the input area.

Of the four bars present at each instant, three are irrelevant. The remaining one belongs to the relevant feature class, because its position determines reward: a reward is given whenever the relevant bar is at one specific position, irrespective of the position of the other three bars.

To learn an action strategy, it is not sufficient to learn the relevant bar only at the position where it is when rewarded. Instead, all bars of this class (for example, all horizontal bars in the upper half of the input area) are relevant. This bar needs to be watched whereever its current location; the action needs to be chosen such as to move the bar of this class to the specific position (reward condition). Hence, the appropriate action leading to reward depends on the currently present bar of one class that can be any bar of this class.

These data test for two capabilities of the model: *(i)* Subspace discovery, meaning that there are relevant features only in one half of the input area. The other half shall be disregarded, despite similar input distributions. *(ii)* Feature detection, meaning that of two different feature classes in the same area (horizontal and vertical bars), only the features belonging to one class shall be learnt. This is more difficult, because relevant and irrelevant features overlap.

In addition, we will test the model for re-learning ability. A previously learnt feature will become irrelevant after a given time, and instead, a new feature becomes relevant.

V. Results

The input layer of the network is of size 12×6, consisting of two sub-areas of size 6×6 each. Hence, given the four stimuli at varying positions, there are 36^4 possible input

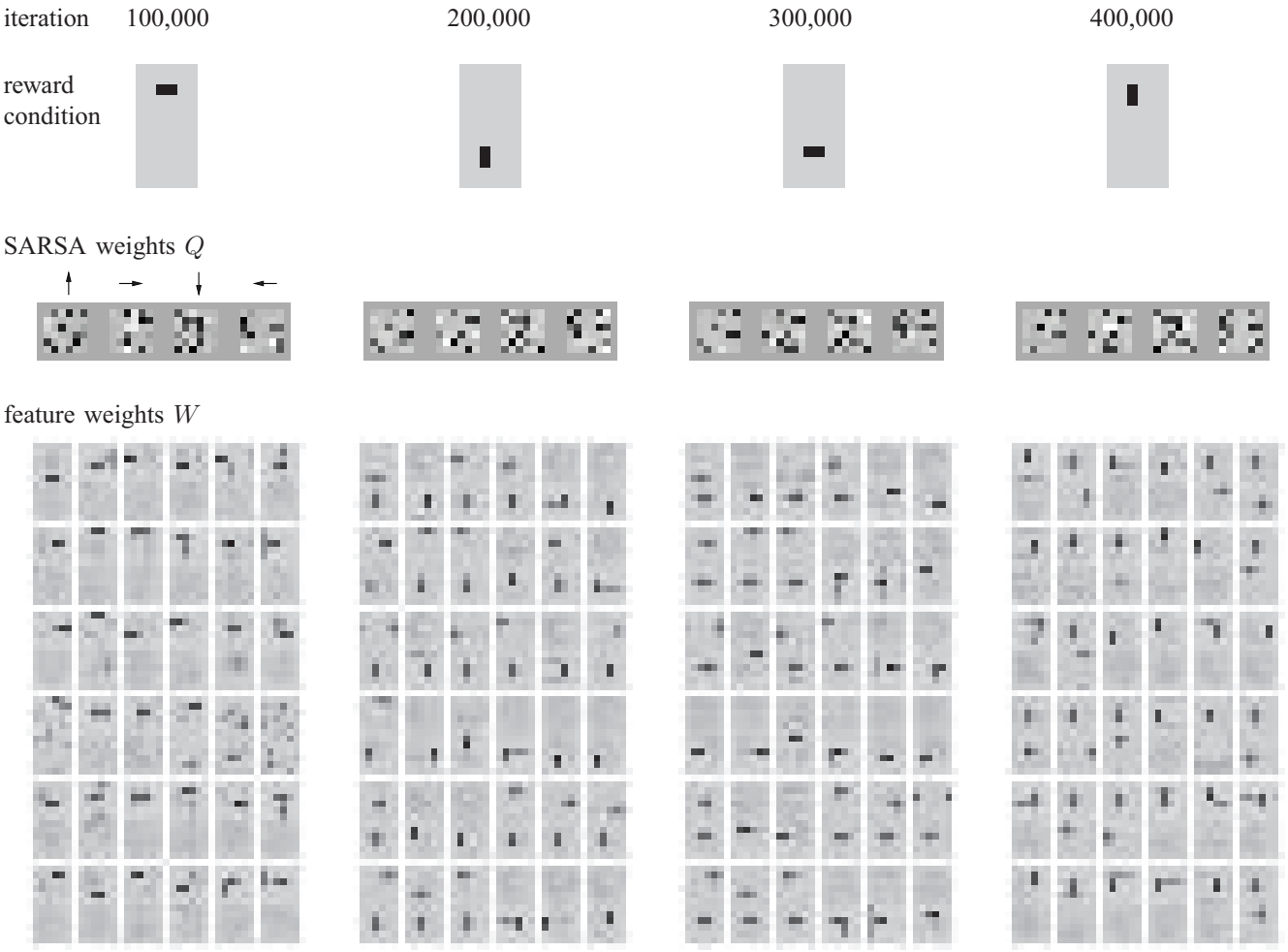

Fig. 3. Weights trained with the Hebb rule of Table I. Weight matrices Q and W from the end of the four training phases are shown, as denoted by the iteration number, and dependent on the reward condition during the respective learning phase. Each of the 4 square fields of Q denotes the input field of an action unit that performs an action to move the bars to the direction that is indicated over the leftmost Q-weights. Each of the 36 rectangular fields of W is a receptive field of a feature unit. Large positive weights are displayed dark. It is clearly seen that the feature units specialize on processing the bars of one of the four classes.

patterns; more than permissible for direct table-based RL[1]. The hidden layer consists of 6×6 units; hence, it has exactly the number of units required to learn one (the relevant) feature class, such as all horizontal bars in the upper half of the input. The output layer consists of 4 action units representing the four directions to move the bars to. Weights were initialised with sparse small positive random values.

The resulting weights trained with the algorithm in Table I and data as in Fig. 2 are shown in Fig. 3. To the left, weights after 100,000 runs are shown during which a reward was given whenever the upper horizontal bar was at a specific position, as indicated at the top of the figure ("reward condition"). It can be seen that most hidden units have feature weights that match one of the upper horizontal bars. However, relevant features that are far from the rewarded position (i.e. in the corners) are not learnt. Instead, irregularities exist, such as weights to the lower input area, and weights with vertical orientation.

Network action performance is shown in Fig. 4 for the four tested learning rules. While it takes on average more than 50 steps for a random actor to stumble upon the reward, after 100,000 iterations the network has learnt to reach the goal in around five steps on average, after random positioning[2]. The gradient-derived "Softmax" learning rule leads to the best performance, requiring only four steps on average to reach the goal.

A. Re-Learning

For continued learning after the initial 100,000 runs, Figure 3 shows weights (for the "Hebb" rule in Table I), and Figure 4 shows the networks' further performances, after the rewarded feature was changed (indicated as "reward condition" in both figures). After this change, the old action strategy based on the formerly relevant features leads – at

[1]The network easily learns larger input sizes, such as 12×12 [34], however, we restrict sizes here for clearer display.

[2]Three steps on average are optimal, which would amount to a practical average of 3.6 steps, because of the 20% probability of an action to fail.

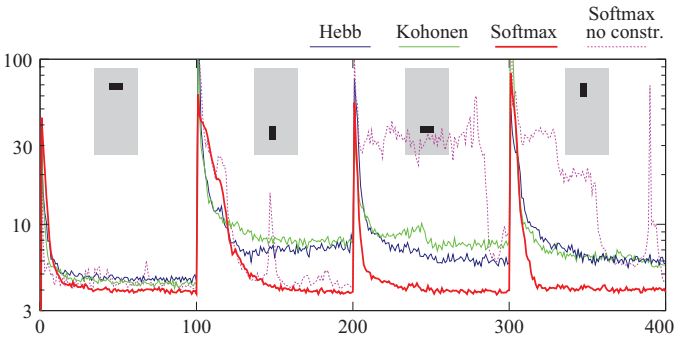

Fig. 4. Learning progress. The graph shows the average number of steps the agent takes until reaching the goal, averaged over 1000 trials, on a log-scale. The x-axis denotes the number of trials. The insets show the "reward condition"; each condition being constant during 100,000 trials.

best – to random movements, and rewards unpredicted by the network. During this phase, the existing action- and feature weights may be unlearned by a negative δ term. For the regular Hebb rule, the feature weights adopt the shape of the relevant feature class (Fig. 3).

In Fig. 4, it can be seen that for the "Softmax" rule the performance reaches a level comparable to the first phase of learning. For the other learning rules, however, improvement during the re-learning phases is worse than in the first phase, indicating susceptibility to initial conditions. The influence of earlier learning phases is reflected in the structure of the feature weights in the second, third and fourth column of Fig. 3: there are some "remnant" feature weights that have been trained in earlier learning phases. It also seems that the action weights Q are "lazy" when adapting to new situations, since they retain similarities in their structure across the different learning conditions.

B. Feature Coding

Both winner-take-all networks, either with "Hebb" or "Kohonen" updates, discover the short bars as relevant features, as seen in the structure of their weights W in Figure 5. The "Softmax" rule, which involves a distributed code, represents the bars less clearly. The Softmax rule without weight constraints, shown in the rightmost column in Figure 5, leads to only few units developing receptive fields. These have negative weights (displayed lighter than the predominant grey that denotes here zero) and positive weights in a horizontal stripe-like fashion, almost reminiscent of Gabor filters, suggesting that a data point is coded by an additive as well as subtractive combination of feature components.

Analogously, non-negativity constraints lead to parts-based representations in unsupervised learning methods [35], particularly when sparse neuronal firing is imposed [36].

VI. Discussion

We have presented a new model that solves the distal reward problem of reinforcement learning while simultaneously performing feature learning. It learns a bars task which is challenging even for humans: when a reward is given whenever one of several features appears at a specific position, it is hard to track which feature consistently appears at the same position over several trials. What limits human performance is not the absence of relevant information, but its accessibility, which is affected by the distractors. Such situations can be mastered by top-down guided learning [37]. Accordingly, in our model, the unsupervised feature learner is aided by a higher-level reinforcement learner to discover the relevant features.

A. Other Methods of Feature Extraction

We have also experimented with a few other models of feature extraction for the first processing step. Some sparse coding models may have difficulties extracting *very* sparse codes, as opposed to just sparse codes: in case of the data consisting of two 2-pixel sized bars, a sparse code is already realized when four units code for one pixel each. The code is, however, *very* sparse only if two units code for the bars. For our data, some sparse coding models would extract the pixels instead of the short bars.

The Földiák model [38] allows for a precise adjustment of the mean unit activations and inter-unit correlations, using neuronal thresholds and lateral weights. It learns the short bars data well, but only if these are presented homogeneously. During the reinforcement learning phase, however, the frequency at which horizontal bars appear varies over the input area, because systematic, goal-directed movements are being made. We would expect similar problems with models that maintain firing rate homeostasis of the feature units (e.g. [39]).

A model by Lücke [40] tolerates uneven activation statistics across the feature layer, and also succeeds on the subspace data. It enforces a soft competition, whereby the winning unit suppresses others, together with a weight normalization. However, it led to less clear horizontal line detectors on the short-lines data. We conclude that a simple winner-take-all model works best for two reasons. First, the data are set up to contain exactly one relevant feature (e.g. the horizontal bar). Second, one unit defines the state for the reinforcement learning algorithm best. Both aspects cannot be improved by allowing more units to become active.

B. Reinforcement Learning vs. Planning

Recently, we have proposed a planning algorithm that can be used in place of reinforcement learning [41]: in the initial exploration phase, the agent learns a world model of every possible state transition, and is later able to plan a route to any location that is its current goal (see also [42], [43]). We conjectured that one must resort to reinforcement learning when it is hard to learn a world model. For the bars data, which we present here, the world model would have to learn also the irrelevant bars, because in the initial exploration phase, no reward related information is being used. Hence, in this specific example, reinforcement learning is more efficient than planning, because it aids in filtering out irrelevant data.

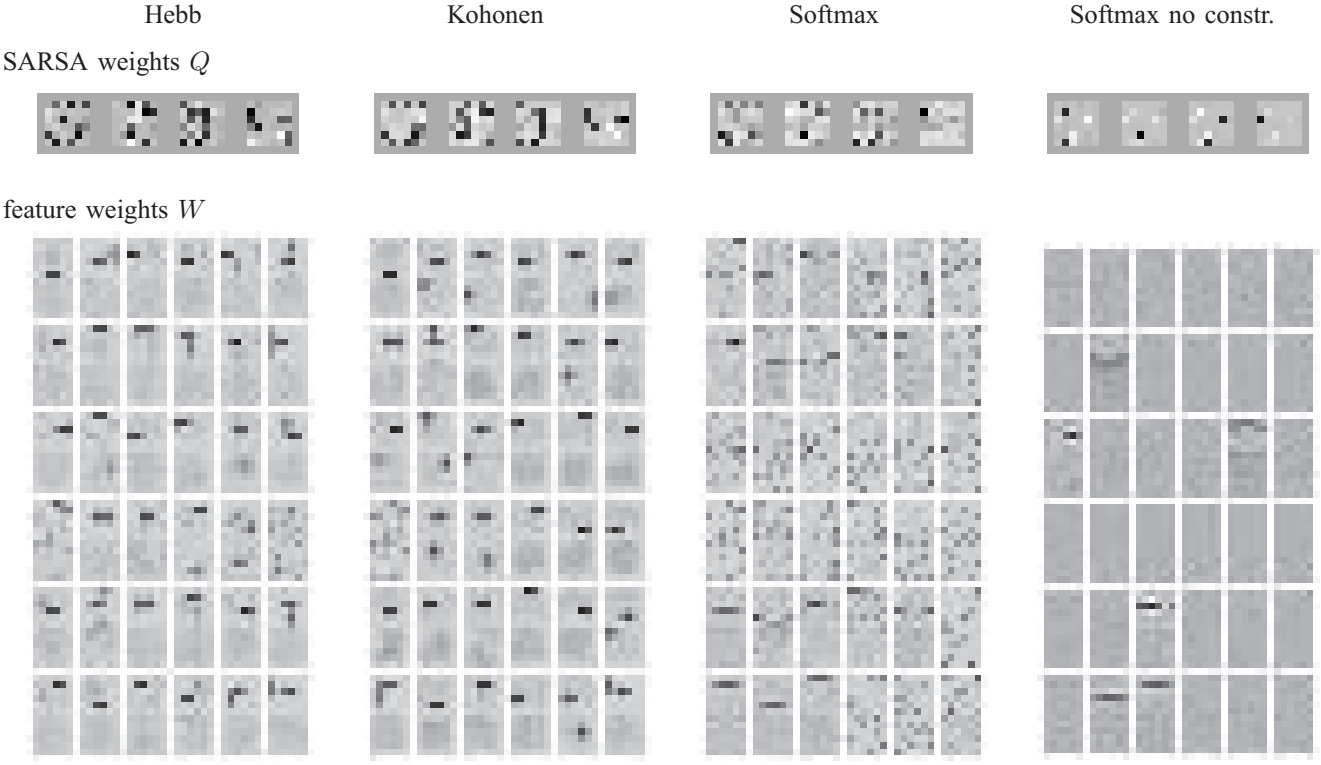

Fig. 5. Weights Q and W resulting from different algorithm versions trained for 100,000 iterations (first reward condition).

C. Applications

There are several possible applications. One is visually-guided balancing in which the model would discover that of all visual features, those that characterize the horizon give useful information about the agent's orientation in space. Another example is grasping an object. A network for grasping would learn to retrieve input from the parietal reach region (PRR) of the posterior parietal cortex, which becomes active when an object is presented at a specific position relative to the hand [44]. For a robotic implementation, we have formerly trained a "what-where" network by supervised learning, which feeds the position of an object into the state space of a reinforcement learner [45]. We would expect that the SARSA-driven feature learner presented here would extract the action-relevant features from the "what" pathway. These would correspond to visual features that convey the location of the object.

D. Extensions

For richer visual processing, it would be worth investigating whether the model can be extended to a hierarchical architecture by applying the δ-signal to more layers concurrently. For example, are visual or visuo-motor cortical neurons, such as in the PRR, themselves trained using reinforcement signals? In the AGREL model, a reward signal helped visual disparity selective cells to emerge, albeit with instantaneous rather than delayed reward [23]. Our SARSA-derived δ signal, which can anticipate a reward, extends the possibilities of reward influence in the cortex.

For the sake of simplicity and generality, we have not exploited topography. In our data, two horizontal bars that are one pixel above/below each other have no overlap. If the bars were spatially blurred (convolved with a Gaussian), then neighboring bars would overlap, and it would be useful to map them to neighboring units on the feature layer, and to perform similar actions. RL can make use of such a blurry state space representation for more robustness and faster learning [46], [45], at the cost of fine-discrimination of states. In our re-learning experiments, topographic neighborhood interactions might help training those units that retain the formerly relevant features and that remain inactive. Further solutions exist to help such inactive units, such as a conscience rule that punishes units that are too frequently active [47].

E. Incremental Learning

The re-learning experiments demonstrate the network's flexibility, akin to neurons of the striatum during habit learning. While rats learnt a T-maze task, striatal neurons were first maximally active at the junction, i.e. when the decision had to be made as to which arm of the maze to go into [1]. When learning progressed, neurons shifted the location of maximal response to the beginning and ending of the maze, suggesting involvement in some higher-level behavior. We proposed that the performance at the junction may have been taken over by the motor cortex [48]; another candidate may be the cerebellum. Given such a module that learns to imitate the RL network (including striatum), the RL

network could then re-assign its sensory resources to perform other, possibly higher-level tasks.

Acknowledgements

We thank Mark Elshaw, Sohrab Saeb, Felipe Gerhard, Constantin Rothkopf, Hauke Bartsch, Dimitri Ognibene and four anonymous reviewers for useful comments on the document.

References

[1] M. Jog, Y. Kubota, C. Connolly, V. Hillegaart, and A. Graybiel, "Building neural representations of habits," *Science*, vol. 286, pp. 1745–9, 1999.

[2] K. Doya, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" *Neural Networks*, vol. 12, pp. 961–74, 1999.

[3] C. Weber, M. Elshaw, S. Wermter, J. Triesch, and C. Willmot, *Reinforcement Learning: Theory and Applications*, 2008, ch. Reinforcement Learning Embedded in Brains and Robots. [Online]. Available: http://intechweb.org/book.php?id=23

[4] M. Shuler and M. Bear, "Reward timing in the primary visual cortex," *Science*, vol. 311, pp. 1606–9, 2006.

[5] A. Schoups, R. Vogels, N. Qian, and G. Orban, "Practising orientation identification improves orientation coding in V1 neurons," *Nature*, vol. 412, pp. 549–53, 2001.

[6] T. Hazy, M. Frank, and R. O'Reilly, "Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system," *Phil. Trans. R. Soc. B*, 2007.

[7] J. Houk, C. Bastianen, D. Fansler, A. Fishbach, D. Fraser, P. Reber, S. Roy, and L. Simo, "Action selection and refinement in subcortical loops through basal ganglia and cerebellum," *Phil. Trans. R. Soc. B*, 2007.

[8] J. Brown, D. Bullock, and S. Grossberg, "How laminar frontal cortex and basal ganglia circuits interact to control planned and reactive saccades," *Neural Networks*, vol. 17, pp. 471–510, 2004.

[9] N. Sengör, Ö. Karabacak, and U. Steinmetz, "A computational model of cortico-striato-thalamic circuits in goal-directed behaviour," in *ICANN*, 2008.

[10] J. Reynolds, B. Hyland, and J. Wickens, "A cellular mechanism of reward-related learning," *Nature*, vol. 413, pp. 67–70, 2001.

[11] M. Botvinick, Y. Niv, and A. Barto, "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective," *Cognition*, vol. doi:10.1016/j.cognition.2008.08.011, (in press).

[12] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–77, 1992.

[13] ——, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[14] A. Moore and C. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, pp. 1–36, 1995.

[15] S. Jodogne and J. Piater, "Closed-loop learning of visual control policies," *J Artificial Intelligence Research*, vol. 28, pp. 349–91, 2007.

[16] A. McCallum, "Reinforcement learning with selective perception and hidden state," Ph.D. dissertation, U. of Rochester, 1995.

[17] M. Farries and A. Fairhall, "Reinforcement learning with modulated spike timing-dependent synaptic plasticity," *J Neurophysiol*, vol. 98, pp. 3648–65, 2007.

[18] R. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neur Comp*, vol. 19, no. 6, pp. 1468–502, 2007.

[19] E. Izhikevich, "Solving the distal reward problem through linkage of STDP and dopamine signaling," *Cerebral Cortex*, vol. 17, pp. 2443–52, 2007.

[20] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback," *PLoS Comput Biol*, vol. 4, no. 10, p. e1000180, 2008.

[21] H. Nakahara, S. Amari, and O. Hikosaka, "Self-organization in the basal ganglia with modulation of reinforcement signals," *Neur Comp*, vol. 14, pp. 819–44, 2002.

[22] P. Roelfsema and A. van Ooyen, "Attention-gated reinforcement learning of internal representations for classification," *Neur Comp*, vol. 17, pp. 2176–214, 2005.

[23] A. Franz and J. Triesch, "Emergence of disparity tuning during the development of vergence eye movements," in *Proceedings of the 6th IEEE International Conference on Development and Learning*, 2007.

[24] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[25] J. Triesch, "Synergies between intrinsic and synaptic plasticity mechanisms," *Neur Comp*, vol. 19, pp. 885–909, 2007.

[26] A. Smith, "Applications of the self-organising map to reinforcement learning," *Neural Networks*, vol. 15, no. 8-9, pp. 1107–24, 2002.

[27] J. Provost, B. Kuipers, and R. Miikkulainen, "Self-organizing perceptual and temporal abstraction for robot reinforcement learning," in *AAAI Workshop on Learning and Planning in Markov Processes*, 2004.

[28] H. Dozono, R. Fujiwara, and T. Takahashi, "Application of the self organizing maps for visual reinforcement learning of mobile robot," in *Proceedings of the 7th WSEAS International Conference on Artificial intelligence, knowledge engineering and data bases*, 2008, pp. 257–62.

[29] J. Martin-Guerrero, G. Gomez-Perez, E. Soria-Olivas, A. Palomares, and E. Balaguer-Ballester, "Aggregating states using the self-organizing map in a marketing application," in *7th European Workshop on Reinforcement Learning*, 2005, pp. 6–7.

[30] C. Touzet, "Modeling and simulation of elementary robot behaviors using associative memories," *Advanced Robotic Systems*, vol. 3, no. 2, pp. 165–70, 2006.

[31] T. Kohonen, *Self-Organizing Maps*, 3rd ed., ser. Springer Series in Information Sciences. Springer, Berlin, Heidelberg, New York, 2001, vol. 30.

[32] I. Szita and A. Lörincz, "Learning Tetris using the noisy cross-entropy method," *Neur Comp*, vol. 18, pp. 2936–41, 2006.

[33] M. Nguyen, "Cooperative coevolutionary mixture of experts. A neuro ensemble approach for automatic decomposition of classification problems," Ph.D. dissertation, University of Canberra, Australia, 2006.

[34] C. Weber, "Studies on goal-directed feature learning," in *Workshop on Machine Learning Approaches to Representational Learning and Recognition in Vision, FIAS, Nov. 27-28*, 2008.

[35] D. Lee and S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–91, 1999.

[36] P. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457–69, 2004.

[37] M. Ahissar and S. Hochstein, "The reverse hierarchy theory of visual perceptual learning," *TRENDS in Cognitive Sciences*, vol. 8, no. 10, pp. 457–64, 2004.

[38] P. Földiák, "Forming sparse representations by local anti-Hebbian learning," *Biol. Cybern.*, vol. 64, pp. 165–170, 1990.

[39] C. Weber and J. Triesch, "A sparse generative model of V1 simple cells with intrinsic plasticity," *Neur Comp*, vol. 20, pp. 1261–84, 2008.

[40] J. Lücke and J. Bouecke, "Dynamics of cortical columns - self-organization of receptive fields," in *Proc. ICANN*. Springer-Verlag Berlin Heidelberg, 2005, pp. 31–7.

[41] C. Weber and J. Triesch, "From exploration to planning," in *Proc. ICANN*, V. Kurkova, R. Neruda, and J. Koutnik, Eds. Springer-Verlag Berlin Heidelberg, 2008, pp. 740–9.

[42] M. Toussaint, "Learning a world model and planning with a self-organizing, dynamic neural system," in *Proc. NIPS*, 2003.

[43] M. Witkowski, "An action-selection calculus," *Adaptive Behavior*, vol. 15, no. 1, pp. 73–97, 2007.

[44] C. Buneo, M. Jarvis, A. Batista, and R. Andersen, "Direct visuomotor transformations for reaching," *Nature*, vol. 416, pp. 632–6, 2002.

[45] C. Weber, S. Wermter, and A. Zochios, "Robot docking with neural vision and reinforcement," *Knowledge-Based Systems*, vol. 17, no. 2-4, pp. 165–72, 2004.

[46] D. Foster, R. Morris, and P. Dayan, "A model of hippocampally dependent navigation, using the temporal difference learning rule," *Hippocampus*, vol. 10, pp. 1–16, 2000.

[47] D. DeSieno, "Adding a conscience to competitive learning," in *IEEE International Conference on Neural Networks*, 1988, pp. 117–24.

[48] C. Weber, S. Wermter, and M. Elshaw, "A hybrid generative and predictive model of the motor cortex." *Neural Networks*, vol. 19, no. 4, pp. 339–53, 2006.