From Exploration to Planning

Cornelius Weber and Jochen Triesch

Frankfurt Institute for Advanced Studies, Johann Wolfgang Goethe University,
Ruth-Moufang-Straße 1, Frankfurt am Main, Germany
{c.weber,triesch}@fias.uni-frankfurt.de
http://fias.uni-frankfurt.de/neuro

Abstract. Learning and behaviour of mobile robots faces limitations. In reinforcement learning, for example, an agent learns a strategy to get to only one specific target point within a state space. However, we can grasp a visually localized object at any point in space or navigate to any position in a room. We present a neural network model in which an agent learns a model of the state space that allows him to get to an arbitrarily chosen goal via a short route. By randomly exploring the state space, the agent learns associations between two adjoining states and the action that links them. Given arbitrary starting and goal positions, route-finding is done in two steps. First, an activation gradient spreads around the goal position along the associative connections. Second, the agent uses state-action associations to determine the actions leading to ascend the gradient toward the goal. All mechanisms are biologically justifiable.

1 Introduction

Neuro- and computer scientists are trying to formulate and implement models of human intelligence since decades. A model for learning goal directed actions, reinforcement learning [1], is among the most influential. While these algorithms were primarily developed in the field of machine learning, they have behavioral and algorithmic counterparts in neurobiology, such as the reward prediction error that may be coded by dopamine neurons in the basal ganglia [2,3].

Reinforcement learning requires an extensive randomized exploration phase. During this phase, the agent visits every state and makes any possible transition from one state to another usually several times. Despite this experience, the agent does not learn a general model of the environment that would allow it to plan a route to any goal. Instead, for any position it learns an action that leads to the one trained goal only, or, in the case of multiple goal problems [4], to a fixed set of trained goals.

The topic of this paper are learning forward- and inverse models when the agent explores the environment (state space); no goal is present, and no goal-related value function, as it is often used in reinforcement learning, is assigned to the states in this phase. Further, a planning algorithm is presented that uses these trained models to allow the agent then to get from any position in the environment to any other.

Forward- and Inverse Models. The brain seems to use forward- and inverse models in abundance [5,6]. Such models can be involved in the control of simple movements, but they are also implicated in higher level functions such as action understanding by mirror neurons [7,8] or a representation of the "self" [9].

Computational models usually train the forward- and inverse models in a phase in which random motor actions are taken, which is termed "motor babbling". This concept from infant language development [10] is established in developmental robotics [11,12] for learning body control ("embodiment").

In our quest to represent the agent's actions in the entire environment by internal models, we will not build one sophisticated model of a complex action. Instead, we will build many tiny models, each representing only two neighboring states and the action linking them. All together, they will cover all connected state pairs.

During exploration, the agent learns three kinds of internal models. *(i)* Associative models of the environment represent connectedness of states (Fig. 1). *(ii)* Inverse models calculate the action necessary to move from one state to a desired neighboring state (Fig. 2). *(iii)* Forward models predict a future state from a current state and chosen action (Fig. 3).

All weights are trained by an error rule in which the difference between the weight's prediction and the actual outcome determines the scale and the sign of a learning step. We have used such a learning rule for a predictive model interpretation of V1 complex cells in the visual cortex [13].

Planning. The main idea of planning is to use inverse- and forward models repetitively, so that a trajectory between two states can be formed even if they are distant. The other idea that actually comes first, is to form a kind of energy landscape that is maximized toward the goal. We will use associative weights to spread a "goal hill" as neural activation vector around the goal. Its slope will then guide the trajectory that forms using the inverse- and forward models' weights, starting at the agent position and arriving at the goal.

The spread of the "goal hill" along the connections that associate connected states reminds of the pheromone trails that some ants lay when travelling from a food source back home [14]. This process is much more efficient than to build up a value function as reinforcement learning does, even when using the learnt world model (as opposed to acting in the real world), as in Dyna-Q [1]. This work recapitulates with neural networks part of the work of [15] who explains multiple learning paradigms observed in animal behaviour with formal logic.

2 Methods

2.1 Exploration

During exploration, the agent moves around randomly in the environment and learns the relations between states and actions[1]. At a given time step let the agent

[1] In models of purposive actions, the agent moves so as to acquire specifically useful information [16,17,18,19]. For us, however, simple, purely random exploration suffices.

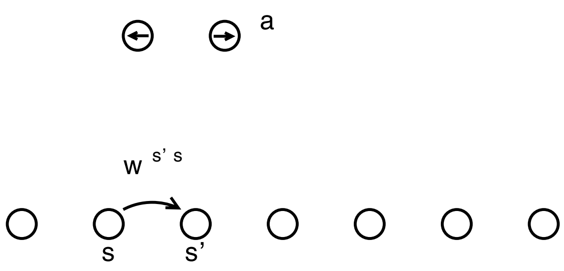

Fig. 1. The model consists of a state space (shown below) and action units (top). For a one-dimensional state space as shown here, only two actions (move left or right) exist. The weights $W^{s's}$ associate neighboring states s and s' irrespective of action taken. For planning, they will be used to find a route between the goal and the agent.

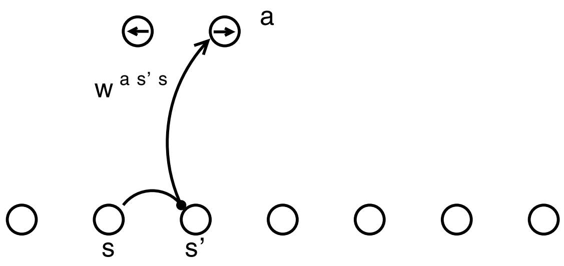

Fig. 2. The weights $W^{a\,s's}$ "postdict" the action a that has been taken in state s when arriving at state s'. For planning, they will be used to identify the action necessary to get to a desired state.

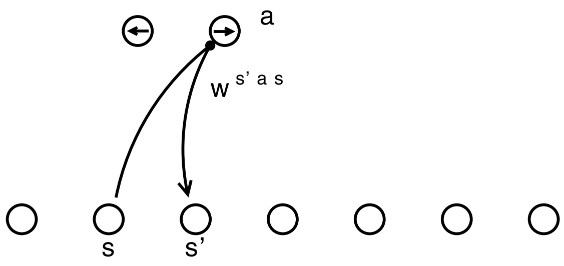

Fig. 3. The weights $W^{s'a\,s}$ predict the next state s' given the current state s and the chosen action a. During planning they will allow mental simulation, without the agent having to move physically.

be in state s and perform action a that leads it to state s'. The environment forms a grid world, and the agent is at one node of the grid. Hence s is a vector with an entry for any node, and all entries are zero except the one corresponding to the position of the agent, which is 1. a is a vector with one component for each action, and the component denoting the chosen action is 1, the others are zero. We assume there is no noise. After sufficient exploration of the environment, we assume that every possible triplet (s, a, s') has been sampled a few times. That is, from every state, every possible action has been taken and every directly reachable (neighboring) state has been reached from it.

To build a model of how actions are related to states, the agent learns three sets of weights. First, $W^{s's}$ tells how likely it is to get to state s' from state s, irrespective of (averaging over) the chosen actions (Fig. 1). Second, $W^{a\,s's}$ tells, which action a has been used to get from s to s' (Fig. 2). Third, $W^{s'a\,s}$ tells, which state s' the agent will arrive at after performing action a in state s (Fig. 3).

The inverse model $W^{a\,s's}$ and the forward model $W^{s'a\,s}$ combine dual input (state-state and state-action, respectively) to generate their output. To activate such a weight, both input conditions must be met, hence it acts as a logical AND gate. Since each state and every action are described by one specific unit being active, the product of the two inputs forms this AND gate. Units that take a sum over such products are called $\Sigma\Pi$ (Sigma-Pi) units.

If s has N elements and a has K elements, then $W^{s's}$ has N^2 elements and $W^{a\,s's}$ and $W^{s'a\,s}$ have KN^2 elements each. Note that most elements of these matrices will be zero, because from a given state only a small number of other states can be reached directly.

At any point during exploration the agent moves from state s using action a to state s'. This triplet (s', s, a) has all the values necessary to learn the three weight types. Each weight type learns to predict its respective outcome value by adjusting the weight incrementally to reduce the prediction error.

For the associative weights $W^{s's}$ the prediction of s' from s is

$$\tilde{s}'_i \;=\; \sum_j w_{ij}^{s's}\, s_j . \tag{1}$$

They are trained to reduce the prediction error $\|s' - \tilde{s}'\|$ according to

$$\Delta w_{ij}^{s's} \;=\; \epsilon\,(s'_i - \tilde{s}'_i)\, s_j \tag{2}$$

with learning rate ϵ. Since these weights do not consider the action being taken, they compute an average over the possible reachable states rather than a precise prediction of the actual next state.

The $W^{a\,s's}$ make a “postdiction” $\tilde{a}$ of the action a that was taken in s before the agent arrived at s':

$$\tilde{a}_k \;=\; \sum_{ij} w_{kij}^{a\,s's}\, s'_i\, s_j . \tag{3}$$

Note that $w_{kij}^{a\,s's}$ is distinguished from $w_{kji}^{a\,s's}$, that is, the direction of movement matters. They are trained to reduce the error $\|a - \tilde{a}\|$ as follows:

$$\Delta w_{kij}^{a\,s's} \;=\; \epsilon\,(a_k - \tilde{a}_k)\, s'_i\, s_j . \tag{4}$$

Here the $\tilde{a}_k$ are continuous values while a_k is 1 for the action that is taken during the random exploration and zero otherwise.

The $W^{s'a\,s}$ predict the actual next state s' by taking into account both s and a:

$$\hat{s}'_i \;=\; \sum_{kj} w^{s'a\,s}_{ikj}\, a_k\, s_j . \tag{5}$$

They are trained to reduce the prediction error $\|s' - \hat{s}'\|$ as follows:

$$\varDelta w^{s'a\,s}_{ikj} \;=\; \epsilon\,(s'_i - \hat{s}'_i)\, a_k\, s_j . \tag{6}$$

2.2 Planning

After learning the weights in the exploration phase, a trajectory between any two states can be planned. The planning process consists of two phases. In the first phase, the unit in state space at the goal position is set active. Activation spreads far around the goal using the associative weights $W^{s's}$. It is important that this emerging activation hill decreases monotonously away from the goal and that the agent position happens to be at its slope. Hence, along a line from the agent to the goal there will be an activation gradient that the agent follows to reach the goal. This gradient ascent is the second phase of the planning process.

Spreading the Goal Hill (Phase 1). Let the goal be at unit g in the state space, so we initialize our “hill” h of activation at the goal with $h_i = 1$, for unit i which is the goal position g, and $h_i = 0$, for all other units.

We spread the activations along those pathways captured in the associative weights $W^{s's}$ by repeating the following three equations for all state space units

$$h'_i = \sum_j w^{s's}_{ij}\; h_j \tag{7}$$

$$h''_i = h_i \,+ h'_i \tag{8}$$

$$h_i = scale(h''_i) \tag{9}$$

until the goal hill reaches the position of the agent. Eq. 7 carries the goal unit's activation one step along all connections $\{w^{s's}_{ij}\}$. Eq. 8 adds up the arriving activations so that they do not become too small away from the goal. In Eq. 9, $scale(.)$ is a non-local function that scales all activities by a constant so that the largest value is 1. It prevents the activations from diverging to large values.

Note that the activation hill spreads primarily along the $W^{s's}$ connections that denote movements *away* from the goal, instead of toward the goal. This is no problem if actions from state unit j to unit i have been taken as often as actions into the other direction during exploration. The $W^{s's}$ ought to be used in the opposite direction by the goal hill as compared to the movements.

Hill Ascent (Phase 2). We start by placing the agent to unit m by defining $f_i = 1$, at position $i = m$, and $f_i = 0$, elsewhere.

Next we activate the action units from combined input of h and f:

$$a_k \;=\; \sum_{ij} w^{a\,s'\,s}_{kij}\, h_i\, f_j. \tag{10}$$

Since f_j is non-zero only at unit m, only those h_i are effective where i is reachable from m. (This is because all $w^{a\,s'\,s}_{kij}$ are zero if state i cannot be reached from state j by any action.) Each of the effective h_i corresponds to one action, and since the h_i nearest to the goal is the largest, the corresponding action unit will be most activated. (This is assuming that the weights are approximately equal.) A winner-take-all function then sets the maximally activated action unit to 1, others to zero.

Finally, we either perform the action of the selected unit in the world, or we mentally simulate it by predicting the next state as follows:

$$f'_i \;=\; \sum_{kj} w^{s'\,a\,s}_{ikj}\, a_k\, f_j \tag{11}$$

A winner-take-all function over the f'_i then determines the next position of the agent. We write this new position as f_i and repeat from Eq. 10 until the agent reaches the goal.

3 Results

First, we tested the model successfully on a plain grid world (not shown). The agent explored the grid and was then able to navigate to several given goal positions. Next, we put some obstacles into the grid world and created the maze shown in Fig. 4 a). The agent explored the space by randomly activating one of its four motor units to move north, west, south or east. Whenever the agent aimed into a wall it was placed to the current position, instead.

All weights were initialized with zero values. Self-connections of $W^{s's}$ were not allowed to grow. 25000 individual movements in the maze, and hence learning steps, were performed with a learning rate of $\epsilon = 0.01$ for all weights. Longer exploration would lead to more homogeneous weights, but was not necessary.

As a result of learning, Fig. 4 b) shows the weights $W^{s's}$ of a few units, those that are boxed in Fig. 4 a). Units on a wall have only zero weights. The other units have non-zero weights from all accessible neighboring fields. The weights are noisy, because the transitions between states s and s' have occurred with different frequencies for different state pairs during exploration.

The planning process is shown in Fig. 5. The two phases, spreading of the "goal hill" and hill ascent are done for a fixed number iterations each, here 48, to cover at least the distance between the agent and the goal. In the first phase, the activation hill h spreads from the goal position throughout the entire maze, avoiding the walls, as shown in Fig. 5 a) and c) for two slightly different goal positions. In the second phase, shown in Fig. 5 b) and d), the agent "travels" without actual interaction with the real world, but only by simulating its movement using Eqs. 10 and 11. Avoiding the wall positions, it navigates toward the goal along a short path. Planning is fully deterministic.

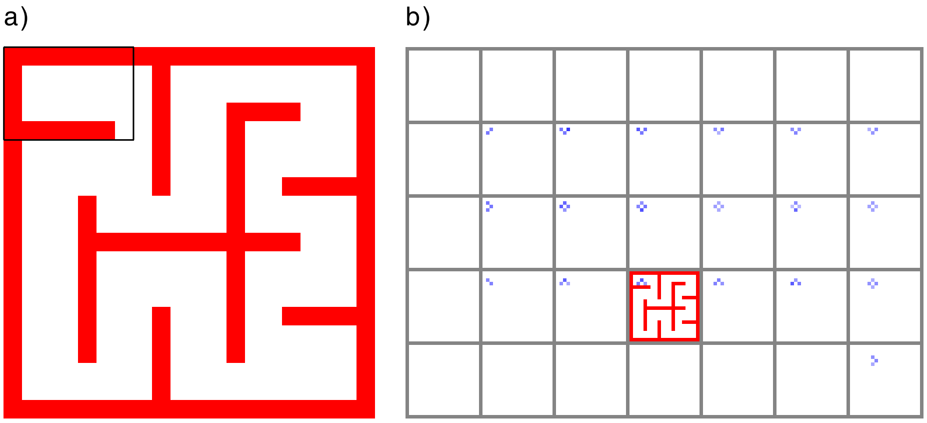

Fig. 4. a) The grid-world, with impenetrable fields shown in red color. b) A part of the associative weight matrix $W^{s's}$. Each little square shows the input weights (receptive field) of one unit; on one unit's input the maze is overlayed. Blue denotes strong weights. Units have no self-connections as seen in the white square "between" a unit's weights to the neighbors. Only units in the upper left part, corresponding to the box in a), are shown.

4 Discussion

Reinforcement Learning. Our small maze example suggests that our planning algorithm greatly enhances reinforcement learning. Are there any cases of goal-finding that can be learnt by reinforcement learning but not by planning?

Planning requires access to the internal representation of the entire state space, including the goal position. Before the agent actually reaches the goal, it might not be possible for it to imagine the goal and to seed the "goal hill" for initializing phase 1 of the planning process. For example, we can plan to navigate a maze if we know the layout, but we have to resort to other strategies if we have only local information.

Experiments which are typical for reinforcement learning often combine incoherent stimuli like a sound, a light and food. In these experiments, the forming of the associative weights $W^{s's}$ may become difficult. Reinforcement learning does not require direct links between neighboring states, but only links from all state space units to a critic and actor units.

In experiments with rats who have to find a hidden platform in a water maze, the rats learn new positions of the platform faster after several blocks of learning. A reinforcement leaning model explained these experiments using a "direction computer" that was not neurally specified [20]. We suggest an alternative explanation, that the rats learn the goals faster, because they have acquired internal models of the water maze.

Robustness. In Fig. 5 d) we can see that the agent misses the goal narrowly. The reason, we conjecture, is that the "goal hill" has become very flat around the

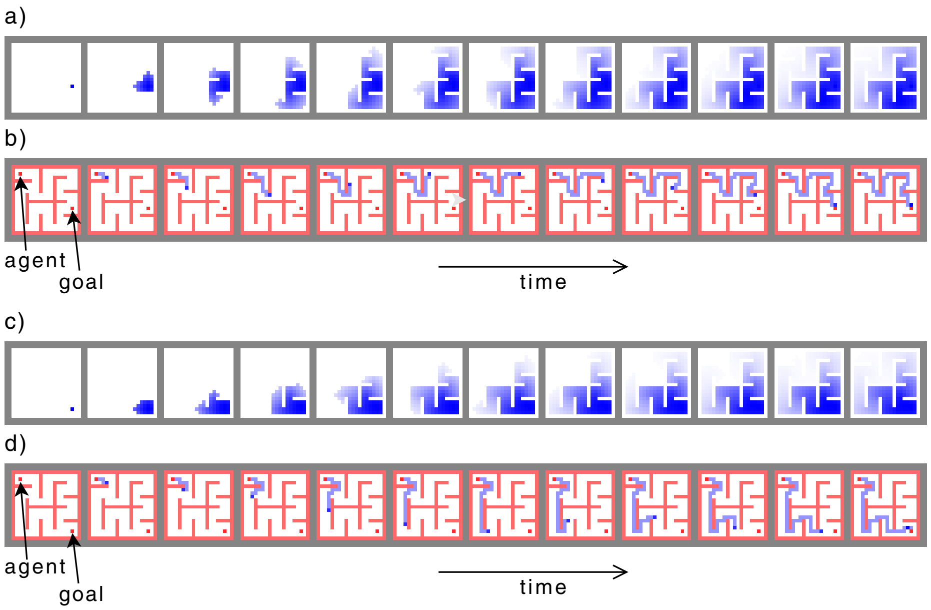

Fig. 5. Network activations in the planning phase. a) and c) show the "goal hill" h spreading over the entire maze during 48 time steps. Only every 4th time step is displayed. To make small activations better visible, the square root of h_i is taken four times. b) and d) show the agent position (dark blue point). The starting- and goal positions are marked by a red point. The agent's trajectory is marked light blue.

goal, while there is noise in the $W^{a\,s'\,s}$ weights that are responsible to choose the action together with information from the gradient of the hill (Eq. 10). Generally for a hill of limited height, the wider it is, the narrower becomes its slope, making it harder to identify[2].

The weights $W^{a\,s'\,s}$ and $W^{s'\,a\,s}$ are not expected to be uneven in a deterministic scenario such as our grid world. However, in our simulation the weights are noisy because we have stopped learning well before convergence (neither have we annealed the learning rates).

A possible remedy to deal with a flat slope is to spread the "goal hill" several times, and let the agent ascend it only when it perceives changes in the hill's activations. Another remedy may be to use the timing of arriving spikes (cf. [21]). The idea is to let spikes (or bursts) travel away from the hill center along the $W^{s'\,s}$ and to let the agent move toward the direction from which it first receives a spike.

A noisy environment as well as a probabilistic description of the model are to be tackled in the future. Furthermore, since the implementation is neurally and

[2] Note that in Eq. 10 the *relative* activations h around the agent are important for determining the action. A slope with a fixed relative gradient could range indefinitely, however, the values would become infinitesimally small.

not table-based, it may not require discrete states. Hence, we may test in future whether the model works with distributed representations.

Short Path. The agent does not necessarily take the shortest path to the goal.

In the exploration and learning phase, the agent may prefer certain actions over others when in certain states (just like in the maze some actions are without an effect when directed at a wall, but in a realistic scenario, an agent might have just a weak tendency to avoid rough terrain). This would lead to uneven $W^{s's}$, and in the first phase of planning, the "goal hill" would be increased along the typically preferred paths. This will lead to — wanted or unwanted — preferences for well experienced paths during goal seeking as well.

Furthermore, when spreading the "goal hill", the input to a unit will be higher when arriving via multiple paths rather than via one path only. Accordingly, this would lead to a preference to travel along "safe" states from which multiple paths lead to the goal. If this is undesired, schemes that consider only the strongest input to spread the "goal hill" are conceivable (cf. Riesenhuber & Poggio's "MAX" operator in a model of visual cortex).

Embedding Architecture. Our simple model includes only state space and action units. Allowing for a variable goal, it is distinguished from a purely reactive sense-action mechanism. However, the question of who sets the goal is not in the scope of the model. This might come from sensory input, for example if an attractive object, such as a fruit, is seen in the visual field. It might as well be determined by a more abstract thought process on a cognitive level.

A challenging question also in reinforcement learning is, how does the state space emerge from the raw sensory stimuli. Models of sensory preprocessing that should fill this gap are mostly trained in an unsupervised fashion, so sensory representations are not optimized for action or planning. Some models learn to represent sensory input following the requirements of reinforcement learning (e.g. [22,23]). Consequentially, one might consider whether sensory processing could be designed to aid planning processes.

Acknowledgments

We acknowledge financial support by the European Union through projects FP6-2005-015803 and MEXT-CT-2006-042484 and by the Hertie Foundation. We thank Felipe Gerhard for a useful discussion, and Cristina Savin, Prashant Joshi, Constantin Rothkopf and the two anonymous reviewers for valuable feedback on the document.

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
2. Ungless, M., Magill, P., Bolam, J.: Uniform inhibition of dopamine neurons in the ventral tegmental area by aversive stimuli. Science 303, 2040–2042 (2004)

3. Tobler, P., Fiorillo, C., Schultz, W.: Adaptive coding of reward value by dopamine neurons. Science 307(5715), 1642–1645 (2005)
4. Foster, D., Dayan, P.: Structure in the space of value functions. Machine Learning 49, 325–346 (2002)
5. Davidson, P., Wolpert, D.: Widespread access to predictive models in the motor system: A short review. Journal of Neural Engineering 2, 8313–8319 (2005)
6. Iacoboni, M., Wilson, S.: Beyond a single area: motor control and language within a neural architecture encompassing broca's area. Cortex 42(4), 503–506 (2006)
7. Miall, R.: Connecting mirror neurons and forward models. Neuroreport 14(16), 2135–2137 (2003)
8. Oztop, E., Wolpert, D., Kawato, M.: Mirror neurons: Key for mental simulation? In: Twelfth annual computational neuroscience meeting CNS, p. 81 (2003)
9. Churchland, P.: Self-representation in nervous systems. Science 296, 308–310 (2002)
10. Plaut, D.C., Kello, C.T.: The emergence of phonology from the interplay of speech comprehension and production: A distributed connectionist approach. In: The emergence of language. B. MacWhinney (1998)
11. Metta, G., Panerai, F., Manzotti, R., Sandini, G.: Babybot: an artificial developing robotic agent. In: SAB (2000)
12. Dearden, A., Demiris, Y.: Learning forward models for robots. In: IJCAI, pp. 1440–1445 (2005)
13. Weber, C.: Self-organization of orientation maps, lateral connections, and dynamic receptive fields in the primary visual cortex. In: Dorffner, G., Bischof, H., Hornik, K. (eds.) ICANN 2001. LNCS, vol. 2130, pp. 1147–1152. Springer, Heidelberg (2001)
14. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization. Computational Intelligence Magazine, IEEE 1(4), 28–39 (2006)
15. Witkowski, M.: An action-selection calculus. Adaptive Behavior 15(1), 73–97 (2007)
16. Schmidhuber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. Connection Science 18(2), 173–187 (1991)
17. Herrmann, J., Pawelzik, K., Geisel, T.: Learning predictive representations. Neurocomputing 32-33, 785–791 (2000)
18. Oudeyer, P., Kaplan, F., Hafner, V., Whyte, A.: The playground experiment: Task-independent development of a curious robot. In: AAAI Spring Symposium Workshop on Developmental Robotics (2005)
19. Der, R., Martius, G.: From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development. In: SAB, pp. 406–421. Springer, Berlin (2006)
20. Foster, D., Morris, R., Dayan, P.: A model of hippocampally dependent navigation, using the temporal difference learning rule. Hippocampus 10, 1–16 (2000)
21. Van Rullen, R., Thorpe, S.: Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. Neur. Comp. 13, 1255–1283 (2001)
22. Roelfsema, P., van Ooyen, A.: Attention-gated reinforcement learning of internal representations for classification. Neur. Comp. 17, 2176–2214 (2005)
23. McCallum, A.: Reinforcement Learning with Selective Perception and Hidden State. PhD thesis, U. of Rochester (1995)