Reinforcement learning

Florentin Woergoetter, BCCN, University of Goettingen, Germany

Dr. Bernd Porr, University of Glasgow

Reinforcement learning (RL) is learning by interacting with an environment. An RL agent learns from the consequences of its actions, rather than from being explicitly taught and it selects its actions on basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially *trial and error* learning. The reinforcement signal that the RL-agent receives is a numerical reward, which encodes the success of an action's outcome, and the agent seeks to learn to select actions that maximize the accumulated reward over time. (The use of the term reward is used here in a neutral fashion and does not imply any *pleasure*, *hedonic impact* or other psychological interpretations.)

Overview

In general we are following Marr's approach (Marr et al 1982, later re-introduced by Gurney et al 2004) by introducing different levels: the algorithmic, the mechanistic and the implementation level.

The Algorithmic level (Machine-Learning perspective)

The best studied case is when RL can be formulated as class of Markov Decision Problems (MDP). The agent can visit a finite number of *states* and in visiting a state, a numerical *reward* will be collected, where negative numbers may represent punishments. Each state has a changeable *value* attached to it. From every state there are subsequent states that can be reached by means of *actions*. The value of a given state is defined by the *averaged future reward* which can be accumulated by selecting actions from this particular state. Actions are selected according to a policy which can also change. The goal of an RL algorithm is to select actions that maximize the expected cumulative reward (the *return*) of the agent.

In general. RL methods are employed to address two related problems: the *Prediction Problem* and the *Control Problem*.

1. **Prediction only:** RL is used to learn the value function for the policy followed. At the end of learning this value function describes for every visited state how much future reward we can expect when performing actions starting at this state.
2. **Control:** By interacting with the environment, we wish to find a policy which maximizes the reward when traveling through state space. At the end we have obtained an *optimal policy* which allows for action planning and optimal control. Since this is really a predictive type of control, solving the control problem would seem to require a solution to the prediction problem as well.

In general there exist several ways for determining the optimal value function and/or the optimal policy.

If we know the state transition function function T(s,a,s'), which describes the transition probability in going from state s to s' when performing action a, and if we know the reward function r(s,a), which determines how much reward is obtained at a state, then algorithms can be which are called *model based algorithms*. They can be used to acquire the optimal value function and/or the optimal policy. Most notably here *Value-Iteration* and *Policy-Iteration* are being used, both of which have their origins in the field of Dynamic Programming (Bellmann 1957) and are, strictly-speaking, therefore not RL algorithms (see Kaelbling et al 1996 for a discussion).

If the model (T and r) of the process is not known in advance, then we are truly in the domain of RL, where by an adaptive process the optimal value function and/or the optimal policy will have to be learned. The most influential algorithms, which will be described below, are:

- Temporal Difference Learning: TD; by itself used for value function learning,
- adaptive Actor-Critics: an adaptive policy iteration algorithm, which approximates the model of the value function by TD where the TD error is used for boththe actor and critic, and
- Q-learning: a unifying algorithm which allows for simultaneous value function and policy optimization.

The mechanistic level (Neuronal Perspective)

Early on, we note that the state-action space formalism used in reinforcement learning (RL) can be also translated into an equivalent neuronal network formalism, as will be discussed below. Note, the neuronal perspective of RL is in general indeed meant to address biological questions. Its goals are usually not related to those of other artificial neural network (ANN) approaches (this is addressed by the machine-learning approach of RL).

Overview: from the algorithmic level to the neuronal implementation

Figure 1 shows a summary diagram of the embedding of reinforcement learning depicting the links between the different fields. Red shows the most important theoretical and green the biological aspects related to RL, some of which will be described below (Wörgötter and Porr 2005). RL plays a role in machine learning (optimal control) but also in theories of animal (human) learning relating RL to some aspects of psychology (classical conditioning and instrumental conditioning). At the same time, RL-concepts developed in machine learning seem to find their correspondence in the response of certain neurons in the brain (see Reward Signals). Furthermore RL is necessarily linked to biophysics and the theory of synaptic plasticity.

RL methods are used in a wide range of applications, mostly in academic research but also in fewer cases in industry. Typical application fields are:

- **Systems control**, e.g. learning to schedule elevator dispatching,

(Crites and Barto 1996);

- **Playing Games**, e.g. TD-Gammon (Thesauro 1994), and

- **Simulations of animal learning** (simulating classical, Sutton and Barto 1981, or instrumental conditioning tasks, Montague et al 1995, simulating tropisms, Porr and Wörgötter 2003).

Background and History

A detailed account of the history of RL is found in Sutton and Barto (1998) [1] (http://www.cs.ualberta.ca/~sutton/book/the-book.html) . Here, we will only summarize the most important contributions.

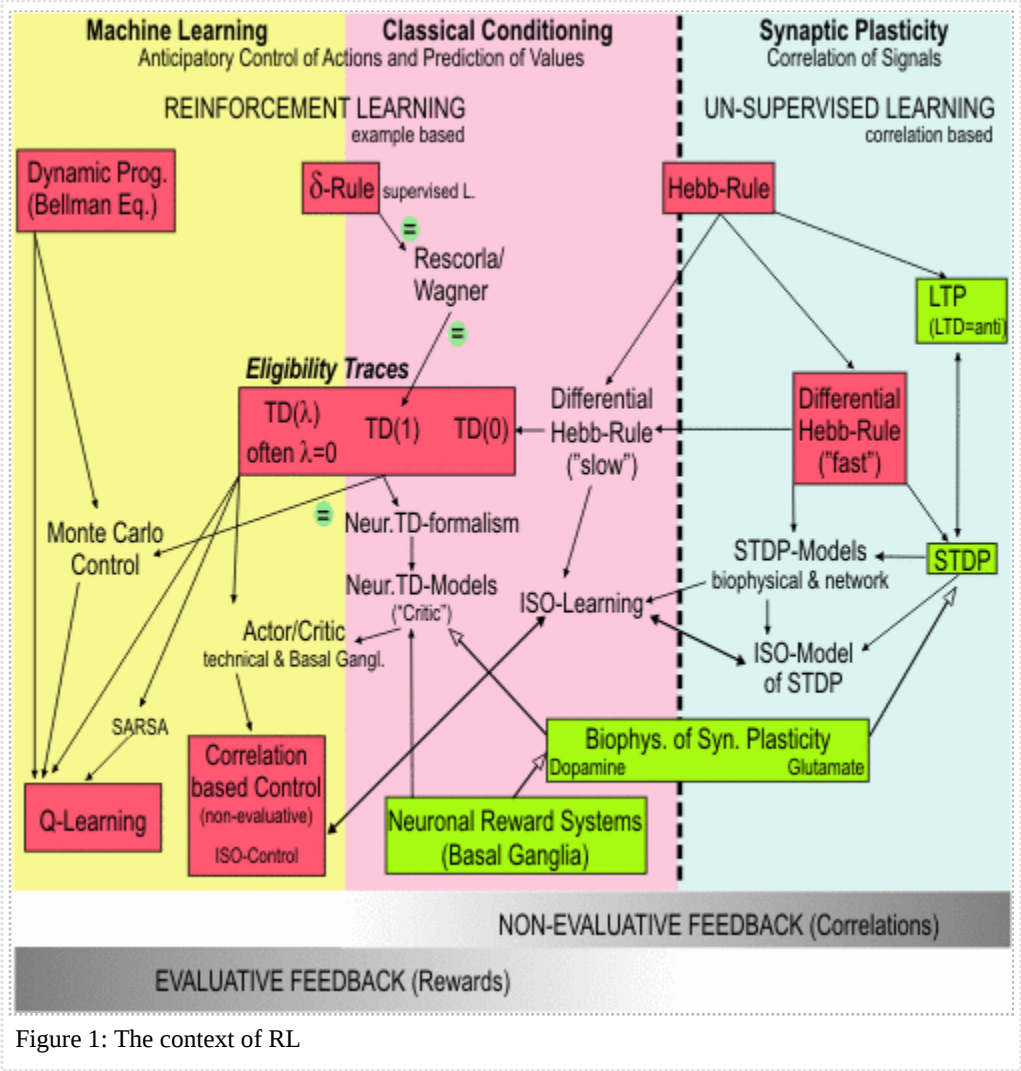

Figure 1: The context of RL

The top part of Figure 1 shows that several academic disciplines have contributed to RL. Most notably there are two:

1. Optimal control (left side).
2. Animal learning by trial and error (middle).

Optimal control problems have been addressed by methods of dynamic programming (Bellmann 1957) which is a large scientific area in its own right (not to be discussed here). Trial-and-error learning has roots in Psychology, especially Classical Conditioning and instrumental conditioning. As a consequence, the first stream (optimal control) was from the beginning governed by highly algorithmical/mathematical approaches, whereas for the second stream (animal learning) it took much longer for the first, still more qualitative, mathematical models to be developed (see, for example, the Rescorla-Wagner Model). Optimal control and instrumental conditioning deal with

closed-loop control problems. However, Classical Conditioning deals with a prediction-only problem because the response of the animal does not influence the experiment, or - in more general terms - does not influence the environment. A good short summary relating algorithmic approaches to real classical conditioning experiments is given by Balkenius and Moren (1998).

Arising from the interdisciplinary study of these two fields, there appeared a very influential computational method, called the method of **Temporal Difference Learning** (TD) (Witten 1977, Sutton and Barto 1981). TD learning was originally mainly associated to animal learning (Classical Conditioning), where an early occurring reinforcer (see the stimuli in Figure 2), the conditioned stimulus (CS), needs to be associated with a later occurring unconditioned stimulus (US) creating a situation where temporal differences of a (value-) function need to be evaluated. Goal of this computation is to assure that after learning the CS becomes a predictor of the US (prediction problem). While TD was originally designed to deal with such prediction problems (Sutton and Barto 1981, Sutton 1988), is was also used to solve optimal control problems. Of particular note is the work of Watkins (1989) on Q-learning, a temporal difference-based control algorithm.

It was essentially the work of Klopf (1972, 1975, 1982, 1988), that began to bring TD-methods together with animal learning theories. He also introduced the difference between evaluative and non-evaluative feedback, where he associated evaluative feedback to [[supervised learning]] (feedback from a teacher) and rightfully stated that the environment does not produce any evaluation. Feedback that arrives from the environment at the sensors of a creature can only be non-evaluative. Any evaluation, in this case, must be performed only internally by the animal itself. Because animals don't receive evaluative feedback, RL would appear to be an example of unsupervised learning. However, this formulation hides the subtle, sometimes very troubling, problem of how the environment is actually defined. The reason this issue is a problem will be discussed later (see section on *Problems* below).

TD methods need to predict future rewards. In order to achieve this, TD learning use value functions V(t) which assign values to states and then calculates the change of those values by ways of a temporal derivative. As a consequence, these methods are related to methods of correlation based, **differential Hebbian learning** (right side of Figure 1), where a synaptic weight changes by the correlation between its input signals with the derivative of its output. Such rules were first discussed by Kosco (1986) as well as Klopf (1986, 1988). Sutton and Barto's 1981 paper, however, really also described a differential Hebbian learning rule. Differential Hebbian rules moved back into the focus of interest only after 1997, when they had been related to spike-timing dependent plasticity (Markram et al 1997). In this new context, Gerstner et al rediscovered these rules in 1996 (Gerstner et al 1996) and they had been applied to RL control problems some years later by Porr and coworkers (Porr and Wörgötter 2003, 2006).

Basic Algorithms

A wide variety of algorithms exist to date with which RL problems can be addressed. As most of them will be covered by separate articles of this encyclopedia, we will only give a brief summary over the most important ones here.

Specifically here we need to distinguish between the machine learning- (Sutton and Barto 1998, Kaelbling et al

1996) and the neuronal perspective (Wörgötter and Porr 2005). The machine learning perspective deals with states, values and actions, etc., whereas the neuronal perspective tries to obtain neuronal signals related to reward-expectation or prediction-error (see below).

Again we divide the discussion into the *prediction-problem* (open loop) as well as the *control problem* (closed loop).

The following side-by-side presentation compares the most important basic approaches and should serve as a direction for further reading.

Prediction

Algorithms: Machine learning

TD-Learning: At the core of most RL algorithm lies the method of Temporal Differences (TD, Figure 2A). We consider a sequence of states followed by rewards

$$s_t, r_{t+1}, s_{t+1}, r_{t+2}, \ldots, r_T, s_T \ .$$

The complete return R_t to be expected in the future from state s_t is, thus

$$R_t = r_{t+1} + \gamma^1 r_{t+2} + \ldots + \gamma^{T-t-1} r_T \ ,$$

where $\gamma < 1$ is a discount factor (distant rewards are less important). Reinforcement learning assumes that the value of a state $V(s)$ is directly equivalent to the expected return

$$V(s) = E_\pi(R_t | s_t = s) \ ,$$

where π is here an unspecified action policy. Thus, the value of state s_t can be iteratively updated with

$$V(s_t) \rightarrow V(s_t) + \alpha[R_t - V(s_t)] \ ,$$

where α is a step-size (often =1). Note, if $V(s_t)$ correctly predicts the expected complete return R_t , the update will be zero in average and we have found the final value for V . This method requires to wait until a sequence has reached its terminal state before the value-update can commence. For long sequences this may be problematic. However, given that $E(R_t) = E(r_{t+1}) + \gamma V(s_{t+1})$ we can also update

Mechanisms: Neuronal

Neuronal-TD: A similar algorithm can be designed for the neuronal perspective (as suggested by Dayan, 2002). We assume that a neuron v can approximately predict a reward r , then we should at every time step $t, t+1$ find that $v(t) \cong R(t)$ and $v(t+1) \cong R(t+1)$. Since this is only approximately true (until convergence) we can in the same way define the error by

$$\delta = r(t+1) + v(t+1) - v(t) = r(t+1) - v'$$

(neglecting discount factors here for brevity). Thus we can update weight w_1 with

$$\delta\omega_1 = [x_1 * E]\delta \ ,$$

where $x_1 * E$ is a convolution with the filter kernel E and denotes the fact that input x_1 needs to be remembered for some time in the system. The function E is usually a first order low pass response and is also known as eligibility trace. Because the error δ occurs later than x_1 , the correlation of δx_1 would be zero without this type of memory. Figure 2B shows the basic TD-rule for a neuron with one predictive CS-input x_1 and a reward line r , the US. When combining this with a delay line which splits x_1 into many inputs, each delayed with respect to each other by a unit delay (serial compound representation) this algorithm emulates the backward-TD(1) procedure.

ISO/ICO-learning: An alternative neuronal approach

iteratively by

$$V(s_t) \rightarrow V(s_t) + \ :$$

$$\alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \ ,$$

which is the TD(0) procedure. The elegant trick is to assume that, if the process converges, the value of the next state $V(s_t + 1)$ should be an accurate estimate of the expected return downstream to s_{t+1} . We define the δ-error as

$$\delta_t = [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \ .$$

Normally we would only assign a new value to one state by performing $V(s_t) \rightarrow V(s_t) + \alpha\delta$, not considering any other previously visited states. This, however, can be desirable and can be achieved by so called eligibility traces E , which are used to update earlier visited states "a little bit". We define $E_t = 1$ at the currently visited state and let E decay gradually along states visited in the past with a decay factor $\lambda = 1$, so we can define

$$V(s_{t+j}) \rightarrow V(s_{t+j}) + \alpha\delta_j E_{t+j}$$

for $j \geq 0$. This procedure is known as $Backward - TD(\lambda)$. If $\lambda = 1$ then we are equally considering all previously visited states.

Properties of TD-learning: TD-learning will converge to the final value function assigning to each state its final value, if all states have been visited "often enough". This can, however, lead to very slow convergence if the state space is large. The expectation value of the δ-error denoted by $\Xi(\delta)$ will converge to zero, while δ itself can - for example - also alternate between positive and negative values. For large state spaces and/or sparse rewards convergence may require many steps and can be very slow. It is not possible to a priori assess if TD(λ) will

(Figure 2B) uses a correlation based differential Hebbian learning rule given by:

$$\Delta\omega_i = \mu[x_i * E] \frac{d}{dt} v$$

(ISO-rule), or alternatively using pure input correlations:

$$\Delta\omega_i = \mu[x_i * E] \frac{d}{dt} [x_0 * E]$$

(ICO-rule), where $\mu < 1$ is the learning rate.

Properties of ISO/ICO: In general x_0 is the unconditioned input which drives the post-synaptic potential whereas the conditioned input x_1 converges through a plastic synapse on the neuron. After learning the neuron's output will co-occur with the CS, but since the US-input x_0 converges onto the neuron, this circuit (Figure 3B) can also before learning be used for generating (motor-)output. In ISO all weights are allowed to change, in ICO weight ω_0 remains fixed. The weight change curve is similar to those observed with spike-timing dependent plasticity. Proof exists that ω_1 will converge if $x_0 = 0$, which is a condition that needs to be fulfilled by an input taking a certain value and is called input control.

See Wörgötter and Porr (2005) for a review and Porr and Wörgötter (2006).

perform better than TD(0). See Sutton and Barto (1998).

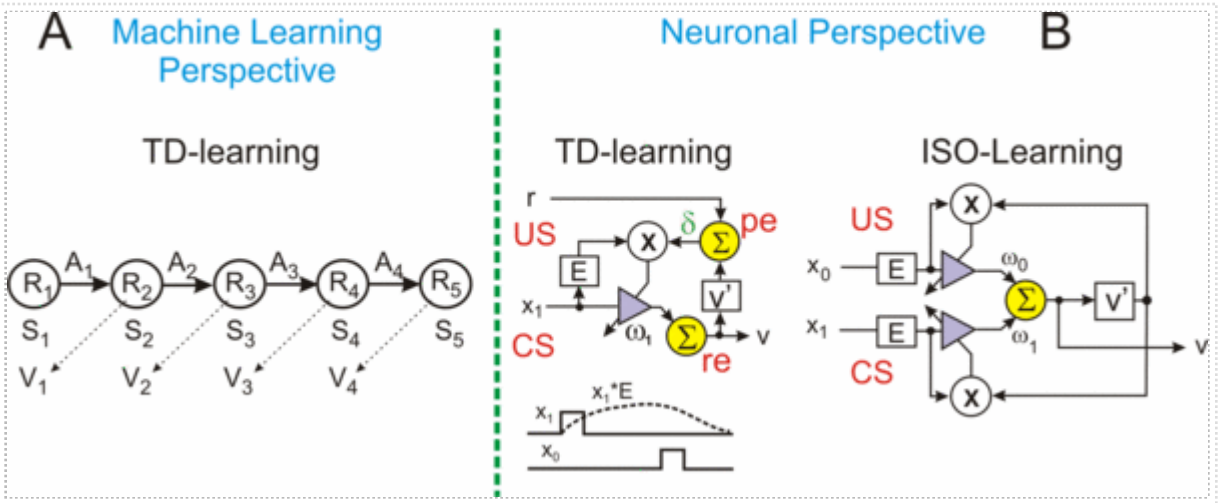

Figure 2: Diagrammatic representation of the different learning methods for the prediction problem. A) An agent travels through states S_i by ways of actions A_i. When reaching a new state a reward R_i will be collected. TD-learning uses these rewards to update the values V_i of previously visited states. TD(0) only updates the value of the most recently visited state (dashed arrows). B) Neuronal TD learning correlates a CS (r=reward) with a US (x_1), where the CS is prolonged by trace E. pe=Prediction Error Neuron, re=Reward Expectation Neuron. The blue amplifier symbol denotes a changeable synapse, the 'X' a multiplication and the prime symbol a temporal derivative. In TD only ω_1 can change. C) In ISO-learning both synapses ω_0 and ω_1 can change.

Control

Algorithms: Machine learning

SARSA (initially known as **modified Q-learning** Rummery and Niranjan, 1994): Probably the nicest aspect of the TD-formalism is that it can be used almost unaltered to address the control problem. We note first that the value of state-action pairs is given by the same formal expectation value of an expected total return R_t as before:

$$Q(s,a) = E_\pi(R_t|s_t = s, a_t = a) \ .$$

The difference is that we have to calculate this now assuming that at moment t we are visiting state s from where we take the specific action a , whereas above the action was left unspecified. The same TD(0) rule can be used to approximate Q with

Mechanisms: Neuronal

Actor-Critic Architectures: Neuronal approaches, which address the control problem and can generate behavior, in general follow a control-loop architecture. Figure 3A shows a conventional feedback control system. In neuronal terms this is a reflex-loop. A controller provides control signals to a system, which is influenced by disturbances. Feedback allows the controller to adjusts it signals. In addition, a set-point is defined which the control loop tries to approximate. Part B shows how to extend this into an Actor-Critic architecture (Witten 1977, Barto et al. 1983, Sutton 1984, Barto 1995). The Critic produces evaluative, reinforcement feedback for the Actor by observing the consequences of its actions. The Critic takes the form of a TD-error, which gives an indication if things have gone

$$Q(s_t, a_t) \rightarrow Q(s_t, a_t) + \quad :$$

$$\alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \ .$$

To calculate this we must for t and t+1 go through the transition: **state, action, reward, state, action**; which gives this update rule its name **SARSA** (Sutton 1996). This method starts with a policy π which is continuously updated during learning (on-policy update).

Q-learning: Uses the rule

$$Q(s_t, a_t) \rightarrow Q(s_t, a_t) + \quad :$$

$$\alpha[r_{t+1} + \gamma \mathrm{max}_a[Q(s_{t+1}, a) - Q(s_t, a_t)] \ .$$

Taking the maximum across all actions a which are possible at state s_t seems to be only a minor modification as compared to SARSA. In effect, however, it makes learning independent of the starting policy π and it allows keeping this policy throughout the whole learning process (off-policy update). When Q-learning has finished, the optimal policy and the optimal value function have been found, without having to continuously update the policy during learning. Formulations for SARSA(λ) and Q(λ) can be derived in a similar way as above (see Sutton and Barto 1998 for a discussion).

Properties of Q-learning and SARSA: Q-learning is the reinforcement learning algorithm most widely used for addressing the control problem because of its off-policy update, which makes convergence control easier. SARSA and Actor-Critics (see below) are less easy to handle. It can be shown that under certain boundary conditions SARSA and Q-learning will converge to the optimal policy if all state-action pairs are better or worse than expected with the preceding action. If the TD-error is positive the tendency to select this action should be strengthened or else, lessened. Thus, Actor and Critic are adaptive through reinforcement learning. On the side of machine learning, Actor-Critics are related to interleaved value/policy-iteration methods (Kaelbling et al 1996). On the side of control, they are related to advanced feed-forward control and feed-forward compensation techniques.

Properties of Actor-Critics: They rely on the return maximization principle trying to maximize the expected return by choosing the best actions. They allow for the learning of goal-directed actions. The Actor uses in general a set of predefined actions. Actions are not easily generated de novo. The Critic cannot generate actions on its own but must work together with the Actor. Convergence is slow if these methods are not augmented by additional mechanisms (Touzet and Santos 2001). Actor-Critics use evaluative feedback from the environment labelled reward=positive or punishment=negative. As $\Xi(\delta) = 0$ is the convergence condition, these systems are governed by output-control. Actor-Critic Architectures are specifically being discussed in conjunction with the Basal Ganglia where different models have been proposed (Gurney et.al. 2004).

Closed-loop ISO/ICO: Figure 3C shows a different approach, where it is assumed that the environment will provide temporally correlated signals x_1, x_0 about upcoming events like a CS-US pair. Learning goal is to minimize the later signal x_0 , which represents an error signal. After learning the primary reflex input x_0 converges to zero. Actor and Critic are not separate in this architecture which does not allow so far the sequencing of actions. Because the system uses only correlations between signals for the learning, it receives strictly non-evalutative feedback from the environment. Convergence is very fast (one-shot learning of ICO, Porr and Wörgötter 2006). As $x_0 = 0$ is the convergence

visited infinitely often.

Actor-Critic Architectures: These play a specific role because originally they had been designed in the context of machine learning as an adaptive policy iteration algorithm. More recently Actor-Critics, however, have been much more discussed in conjunction with the architecture of the basal ganglia (Joel et al 2002). Hence, their properties are being described on the right side of this page under "neuronal control".

Note: for Q-learning and SARSA no neuronal architectures exist so far. Recent results suggest that animals might rather follow a SARSA-like, on-policy update as opposed to a Q-learning like, off-policy update (Morris et al 2006, see also commentary by Niv et al 2006).

condition which is defined at the input, this system performs input control. See Wörgötter and Porr (2005).

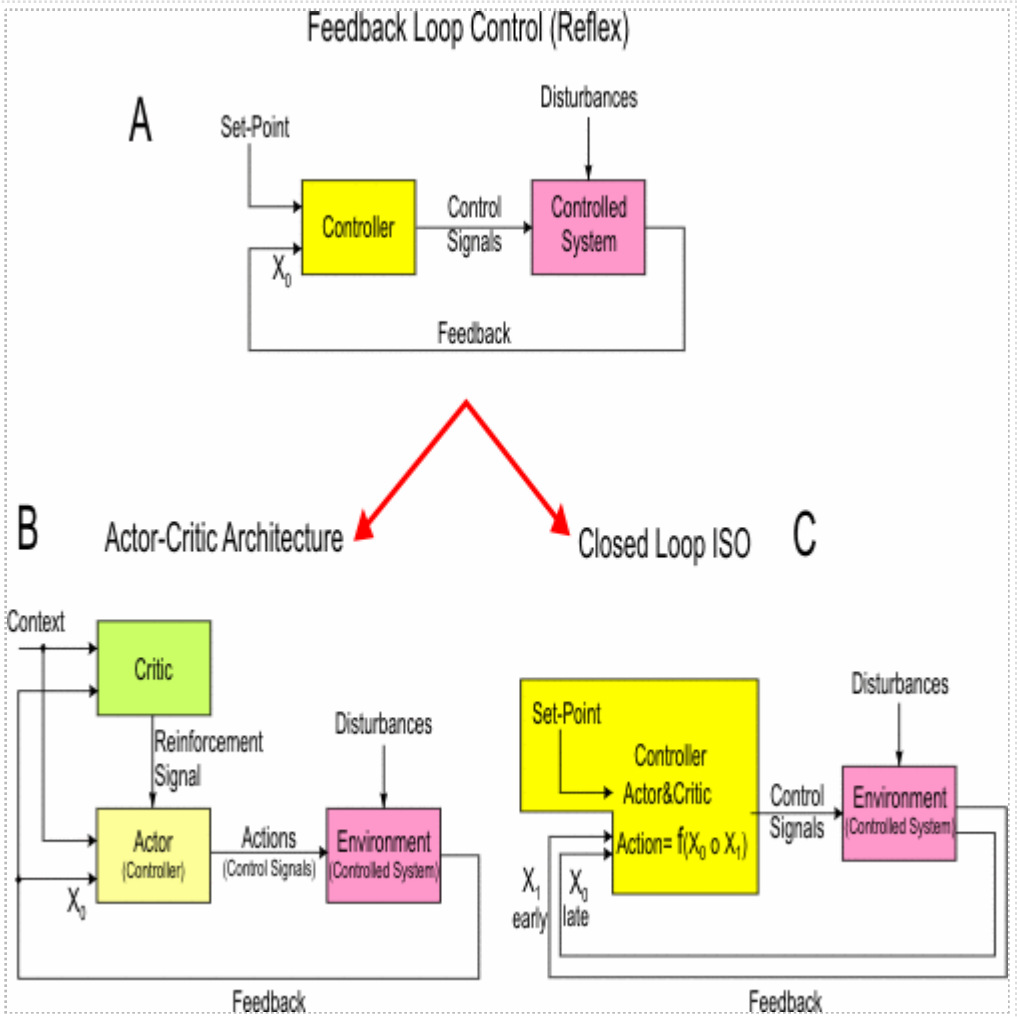

Figure 3: Diagrammatic representation of closed-loop reinforcement methods mainly applied using the neuronal perspective.

Different RL Strategies

RL-methods can be used for learning to **reach a goal step by step** (Goal Directedness). They can however also be used to **learn avoiding a disturbance** (Homeostasis). TD methods can be used to learn goals, ISO/ICO methods are better suited for homeostasis learning. ISO/ICO methods have also been employed to learn attractive (food retrieval) or repulsive (obstacle avoidance) **tropisms**, but so far not for learning step-wise goal-directed actions.

The Implementation-level (Neuroscience)

In the section we are establishing the link between the mechanistic level and the neuroscience, hence establishing a link between the abstracts ANNs presented in the previous sections with neurophysiological findings such as spike timing dependent plasticity and dopaminergic modulation.

Reinforcement learning is also reflected at the level of neuronal sub-systems or even at the level of single neurons. In general the Dopaminergic system of the brain is held responsible for RL. Responses from dopaminergic neurons

have been recorded in the Substantia Nigra pars compacta (SNc) and the Ventral Tegmental Area (VTA) where some reflect the *prediction error* δ of TD-learning (see Figure 3B *pe*). Neurons in the Striatum, orbitofrontal cortex and Amygdala seem to encode reward expectation (for a review see Reward Signals, Schultz 2002, see Figure 3B *re*). These neurons have been discovered mostly in conjunction with appetitive (food-related) rewards. Figure 4 shows some examples of prediction error- as well as reward expectation neurons.

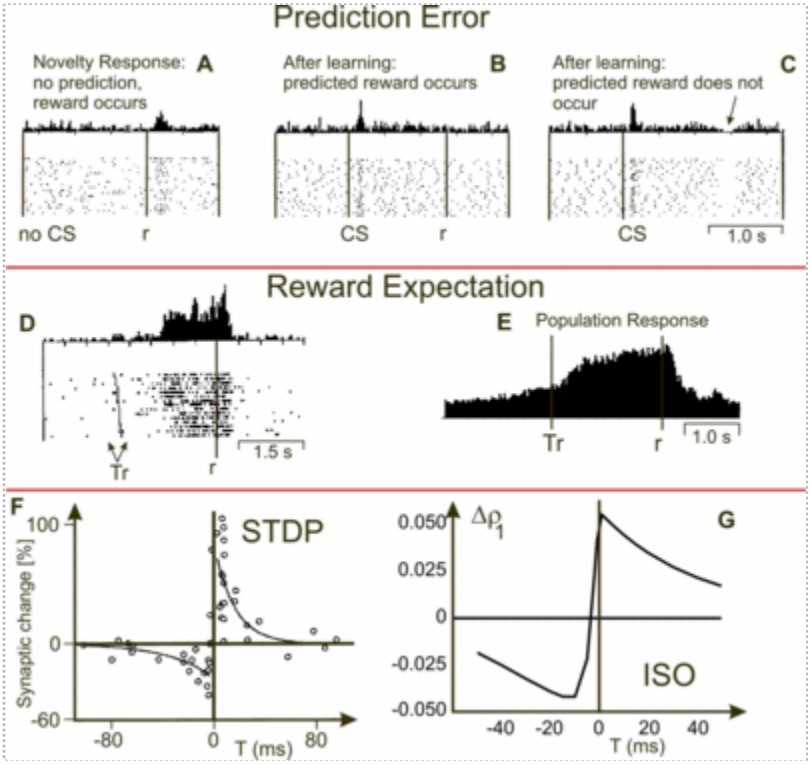

Figure 4: Examples of a Prediction Error (*pe*, A-C) and some Reward Expectation (*re*, D,E) neurons (Schultz, 2002). The bottom panels show the similarity of real STDP curves (F, Markram et al 1997) with the ones obtained from ISO-learning (G, Porr et al 2003).

However, only few dopaminergic neurons produce error signals that comply with the demands of reinforcement learning. Most dopaminergic cells seem to be tuned to arousal, novelty, attention or even intention and possibly other driving forces for animal behavior.
Furthermore the TD-rule reflects a well-defined mathematical formalism that demands precise timing and duration of the δ error, which cannot be guaranteed in the basal ganglia or the limbic system (Redgrave et al. 1999). Consequently, it might be difficult to calculate predictions of future rewards. For that reason alternative mechanisms have been proposed which either do not rely on explicit predictions (derivatives) but rather on a Hebbian association between reward and CS (O'Reilley et al. 2007), or which use the DA signal just as a switch which times learning after salient stimuli (Redgrave and Gurney 2007, Porr and Wörgötter 2007). Hence the concept of derivatives and therefore predictions has been questioned in the basal ganglia and the limbic system and alternative more simpler mechanisms have been proposed which reflect the actual neuronal structure and measured signals.

Differential Hebbian learning (e.g. ISO-rule) seem to be to some degree compatible with novel findings on spike-timing dependent synaptic plasticity (STDP, Markram et al 1997). In this type of plasticity, synapses potentiate (become stronger) when the presynaptic input is followed by post-synaptic spiking activity, while else they are depressed (become weaker). The multiplicative (correlative) properties necessary to emulate a Hebb rule can be traced back to second messenger chains, which phosphorilate AMPA receptors and the required *differential* aspect appears to arise from the sensitivity of real synaptic plasticity to Calcium *gradients* (Lindskog et.al. 2006). Figure 4 shows two examples of weight change curves (often called *learning window*) from a real neuron and from a differential Hebbian learning rule emulated to be compatible with some basic biophysical characteristics

(Saudargiene et al 2004).

Challenges and extensions to RL

In spite of its influence across different fields RL is confronted with a variety of problems, which we will list here. Note that in any given RL scenario the different problems listed here occur with different relevance. Unfortunately this has in general led to a situation in the literature where solutions are tailor-made for the one given problem(-domain). The elegance of the basic RL-methods is this way often lost in a wide variety of add-on mechanisms and add-on parameters.

Curse of Dimensionality

In general it is difficult to define appropriate state- and action spaces in all real-world RL problems. Most often the tiling of the state space has to be rather fine to cover all possibly relevant situations and there can also be a wide variety of actions to choose from. As a consequence there exists a combinatorial explosion problem when trying to explore all possible actions from all possible states. Solutions to this problem apply scale-spacing methods and/or function approximation methods to reduce and/or interpolate the searchable value-space. Both methods try to generalize the value function.

(Temporal) Credit Assignment Problem

This is a related problem. It refers to the fact that rewards, especially in fine grained state-action spaces, can occur terribly temporally delayed. For example, a robot will normally perform many moves through its state-action space where immediate rewards are (almost) zero and where more relevant events are rather distant in the future. As a consequence such reward signals will only very weakly affect all temporally distant states that have preceded it. It is almost as if the influence of a reward gets more and more diluted over time and this can lead to bad convergence properties of the RL mechanism. Many steps must be performed by any iterative reinforcement-learning algorithm to propagate the influence of delayed reinforcement to all states and actions that have an effect on that reinforcement. Similar strategies as above are applied to solve this problem.

Partial Observability Problem

In a real-world scenario an RL-agent will often not know exactly in what state it will end up after performing an action. Basic RL-algorithms cannot be used in this case, because they require full observability. Furthermore states must be history independent. Hence it must be irrelevant how one has reached a certain state (systems must be *Markovian*). Often POMDP methods (POMDP=partial observable Markov decision problems) are used to address this problem. Many solutions to POMDPs have been designed and cannot be reviewed here. A simple introduction is given by (Anthony R. Cassandra: [2] (http://www.cs.brown.edu/research/ai/pomdp/tutorial/index.html)) for more advanced literature one should mainly consult the work of Littman and Kaelbling et al [3] (http://www.cs.duke.edu/~mlittman/topics/pomdp-page.html) .

State-Action Space Tiling

In view of the above problems it turns out that deciding about the actual state- and action-space tiling is difficult as it is often critical for the convergence of RL-methods. Alternatively one could employ a continuous version of RL, but these methods are equally difficult to handle.

Non-Stationary Environments

As for other learning methods, RL will only work quasi stationary environments if the dynamics change slowly. This is a fundamental problem and cannot be mitigated. If the world changes too fast, you cannot learn. As indicated above, many times RL-algorithms do not converge very fast. Hence, slowly converging RL-methods may even fail in slowly changing environments.

Credit Structuring Problem

After deciding about the basic structure on which the RL-agent should operate we are still not done, because one also need to decide about the reward-structure, which will affect the learning. Several possible strategies exist:

- **external evaluative feedback**: The designer of the RL-system places rewards and punishments *by hand*. This strategy generally works only in very limited scenarios because it essentially requires detailed knowledge about the RL-agent's world.
- **internal evaluative feedback**: Here the RL-agent will be equipped with sensors that can measure physical aspects of the world (as opposed to 'measuring' numerical rewards). The designer then only decides, which of these physical influences are rewarding and which not.

For a driving robot with battery, finding the charging station ought to be very rewarding, while hitting a wall should create a punishment. This strategy can be more generally applied, but might create a partially observable situation. In addition, evaluative feedback will *always* require influence of the designer. This can even lead to substantial problems if the world-model of the designer does not match to that of the RL-agent. Pure correlation based methods (e.g. differential Hebbian methods like ISO/ICO) do not use evaluative feedback, because their feedback from the environment is not pre-labeled "good" or "bad" by the designer. Rather the task defines the learning goal which has the advantage that feedback is not limited to a one-dimensional signal such as the reward but can use multidimensional feedback from the environment.

Alternatively one could employ evolutionary methods that evolve their own reward function over a series of generations and avoid the assignment of rewards to environmental stimuli by the experimenter.

Exploration-Exploitation Dilemma

RL-agents need to explore their environment in order to assess its reward structure. After some exploration the agent might have found a set of apparently rewarding actions. However, how can the agent be sure that the found actions where actually the best? Hence, when should an agent continue to explore or else, when should it just exploit

its existing knowledge? Mostly heuristic strategies are employed for example *annealing-like* procedures, where the naive agent starts with exploration and its exploration-drive gradually diminishes over time, turning it more towards exploitation. The annealing rate, however depends also on the structure of the world and especially also on the graining of the state space and cannot be decided without guided guessing. Recently Singh et al, 2002 have developed more efficient solutions for the exploration/exploitation dilemma.

References

- Balkenius, C. and Moren, J. (1998). Computational models of classical conditioning: A comparative study. LUCS 62 ISSN 1101-8453, Lund University Cognitive Studies.
- Barto, A. (1995). Adaptive critics and the basal ganglia. In Houk, J. C., Davis, J. L., and Beiser, D. G., editors, Models of Information processing in the basal ganglia, 215-232. MIT Press, Cambridge, MA.
- Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. In IEEE Transactions on Systems, Man, and Cybernetics, volume 13, 835-846
- Bellman, R. E. (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.
- Crites, R.H. and Barto, A.G. (1998). Elevator Group Control Using Multiple Reinforcement Learning Agents. Machine Learning, 33:235-262.
- Dayan, P (2002). Matters temporal. Trends in Cognitive Sciences, 6, 105-106.
- Gerstner, W., Kempter, R., van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. Nature, 383:76-78.
- Gomi, H. and Kawato, M. (1993). Neural network control for a closed-loop system using feedback-error-learning. Neural Netw., 6(7):933-946
- Gurney, K. N., Prescott, T. J., Wickens, J. R. and Redgrave, P. (2004). Computational models of the basal ganglia: from robots to membranes. Trends Neurosci 27(8): 453-459.
- Joel, D., Niv, Y., and Ruppin, E. (2002). Actor-critic models of the basal ganglia: new anatomical and computational perspectives. Neural Networks, 15:535\u2013547.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A Survey. Journal of Artificial Intelligence Research, 4:237-285.
- Klopf, A. H. (1972). Brain function and adaptive systems - a heterostatic theory. Technical report, Air Force Cambridge Research Laboratories Special Report No. 133, Defense Technical Information Center, Cameron Station, Alexandria, VA 22304.
- Klopf, A. H. (1982). The hedonistic neuron: A theory of memory, learning, and intelligence. Hemisphere, Washington, DC.
- Klopf, A. H. (1986). A drive-reinforcement model of single neuron function. In Denker, J. S., editor, Neural networks for computing: AIP Conf. Proc. , volume 151. New York: American Institute of Physics.
- Klopf, A. H. (1988). A neuronal model of classical conditioning. Psychobiol., 16(2):85-123.
- Kosco, B. (1986). Differential Hebbian learning. In Denker, J. S., editor, Neural networks for computing: AIP

Conference Proc. proceedings, volume 151. New York: American Institute of Physics.

- Lindskog, M., Kim, M., Wikström, M.A., Blackwell, K.T. and Hellgren-Kotaleski J. (2006). Transient Calcium and Dopamine Increase PKA Activity and DARPP-32 Phosphorylation. PLoS, 2:9.
- Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science, 275:213-215.
- Marr, D., & Poggio, T. (1976). From understanding computation to understanding neural circuitry. MIT AI Laboratory.
- Montague, P. R., Dayan, P., Person, C., and Sejnowski, T. J. (1995). Bee foraging in uncertain environments using predictive hebbian learning. Nature, 377:725-728.
- Morris, G., Nevet, A., Arkadir, D., Vaadia, E. and Bergman, H. (2006). Midbrain dopamine neurons encode decisions for future action, Nature Neurosci., 9(8), 1057-1063.
- Niv, Y., Daw, N. D. and Dayan, P. (2006). Choice Values (Commentary to Morris et al 2006), Nature Neurosci., 9(8), 987-988.
- O'Reilly, R.C., Frank, M.J., Hazy, T.E. & Watz, B. (2007). PVLV: The Primary Value and Learned Value Pavlovian Learning Algorithm. Behavioral Neuroscience. In press.
- Porr, B. and Wörgötter, F. (2003). Isotropic sequence order learning. Neural Comp., 15:831-864.
- Porr, B. and Wörgötter, F. (2007). Learning with *Relevance*: Using a third factor to stabilise Hebbian learning. Neural Comp. In press.
- Redgrave, P, Prescott, T.J. and Gurney, K. (1999). Is the short-latency dopamine response too short to signal reward error? Trends neurosci. 22:4 146-151.
- Redgrave, P and Gurney, K.N. (2007). What does the short-latency dopamine signal reinforce? Nature Reviews Neuroscience. In press.
- Rescorla, R. A. and Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H. and Prokasy, W. F., editors, Classical Conditioning II: Current Research and Theory. Appleton Century Crofts, New York.
- Saudargiene, A., Porr, B., and W¨org¨otter, F. (2004). How the shape of pre- and postsynaptic signals can influence STDP: a biophysical model. Neural Comp., 16:595-626.
- Singh, S., Kearns, M. (2002). Near-Optimal Reinforcement Learning in Polynomial Time. Machine Learning journal, Volume 49, Issue 2, pages 209-232, 2002.
- Schultz, W. (2002). Getting formal with dopamine and reward. Neuron, 36:241-263.
- Sutton, R. S. (1984). Temporal credit assignment in reinforcement learning. PhD thesis, University of Massachusetts, Amherst
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. Mach. Learn., 3:9-44.
- Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coding. In D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (eds.) Advances in Neural Information Processing, pp. 1038-1044, MIT Press, Cambridge, CA.
- Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. Bradford Books, MIT Press,

Cambridge, MA, 2002 edition.

- Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural Comp. 6/2:215-219.
- Touzet, C. and Santos, J. F. (2001). Q-learning and Robotics. IJCNN'99, European Simulation Symposium, Marseille.
- Watkins, C. J. C. H. (1989). Learning from delayed rewards. PhD thesis, University of Cambridge, Cambridge, England.
- Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. Information and Control, 34:286-295
- Wörgötter, F. and Porr, B. (2005) Temporal sequence learning, prediction and control - A review of different models and their relation to biological mechanisms. Neural Comp. 17:245-319

Internal references

- Joseph E. LeDoux (2008) Amygdala. Scholarpedia, 3(4):2698.
- Peter Redgrave (2007) Basal ganglia. Scholarpedia, 2(6):1825.
- Valentino Braitenberg (2007) Brain. Scholarpedia, 2(11):2918.
- Nestor A. Schmajuk (2008) Classical conditioning. Scholarpedia, 3(3):2316.
- Jeremy Seamans and Daniel Durstewitz (2008) Dopamine modulation. Scholarpedia, 3(4):2711.
- James Meiss (2007) Dynamical systems. Scholarpedia, 2(2):1629.
- Howard Eichenbaum (2008) Memory. Scholarpedia, 3(3):1747.
- Victor M. Becerra (2008) Optimal control. Scholarpedia, 3(1):5354.
- Robert Rescorla (2008) Rescorla-Wagner model. Scholarpedia, 3(3):2237.
- Wolfram Schultz (2007) Reward. Scholarpedia, 2(3):1652.
- Wolfram Schultz (2007) Reward signals. Scholarpedia, 2(6):2184.
- David H. Terman and Eugene M. Izhikevich (2008) State space. Scholarpedia, 3(3):1924.
- Andrew G. Barto (2007) Temporal difference learning. Scholarpedia, 2(11):1604.

Acknowledgements

F. Woergoetter acknowledges the support by BMBF (Federal Ministry of Education and Research), BCCN (Bernstein Center for Computational Neuroscience) - Goettingen project W3.

External Links

See Also

Actor-Critic Method, Basal Ganglia, Conditioning, Dynamic Programming, Eligibility Trace, Exploration vs. Exploitation, Neural Networks, Neuroeconomics, Q-Learning, Rescorla-Wagner Model, Reward, Reward Signals, Supervised Learning, TD-Gammon, Temporal Difference Learning, Unsupervised Learning

Sponsored by: Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia
Reviewed by (http://www.scholarpedia.org/w/index.php?title=Reinforcement_learning&oldid=16462) : Anonymous
Accepted on: 2007-09-18 05:37:33 GMT (http://www.scholarpedia.org/w/index.php?title=Reinforcement_learning&oldid=20972)

Categories: Conditioning | Reinforcement Learning | Computational Neuroscience | Computational Intelligence