Reinforcement Learning

Embodied AI VL WS16/17 Supplement

Oswald Berthold

Adaptive Systems Group, Dept. of Computer Science, Humboldt-Universität zu Berlin

February 6, 2017

Outline

Basics

Reinforcement Learning – Verstärkungslernen

This means

There is something (a signal) that reinforces

There is something (structure) which is reinforced

Plain language definition

Learn from, and while interacting with, the environment. The reinforcing signal comes from this interaction.

Amplify correlation between any action and its causes (states) yielding favourable consequences. The associative link between states and actions is reinforced.

Assertion: Any algorithm that "solves" this is an RL algorithm. Question: what do we accept as a solution? This is where RL and DR disagree?

Quote: Any algorithm . . .

Reinforcement learning, [. . .], is simultaneously a problem, a class of solution methods that work well on the class of problems, and the field that studies these problems and their solution methods. Reinforcement learning problems involve learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. In an essential way these are closed-loop problems because the learning system's actions influence its later inputs. Moreover, the learner is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These three characteristics — being closed-loop in an essential way, not having direct instructions as to what actions to take, and where the consequences of actions, including reward signals, play out over extended time periods — are the three most important distinguishing features of the reinforcement learning problem.
[. . .]
Any method that is well suited to solving such problems we consider to be a reinforcement learning method. [1]

Aha!

[1] SB pg. 1-2

Genealogy

What is the history of ideas leading to Reinforcement Learning?

Psychology conditioning, behaviourism, explain how animals acquire skills and behaviour.

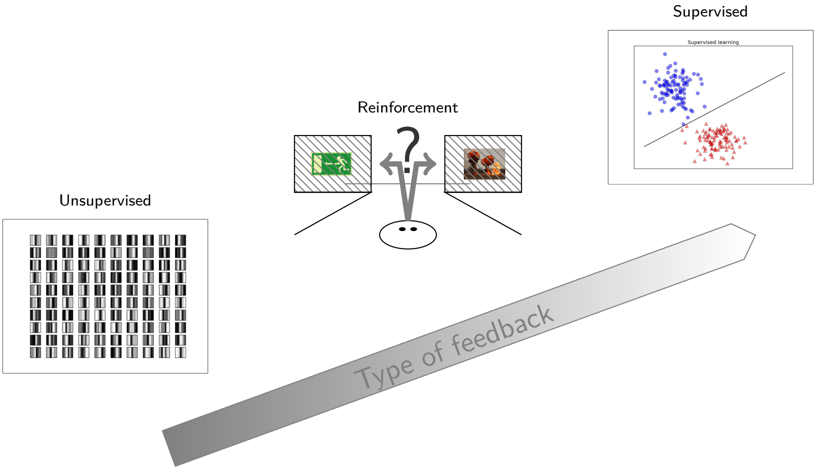
$\searrow$

RL

$\nwarrow$

Computational approach, formal definition of the problem, algorithms for solving it **AI**

Machine learning context



Feedback spectrum: Types of learning principles populate spectrum of feedback types [2]

[2] Images from Sullivan, `https://commons.wikimedia.org/wiki/File:Explosions.jpg`, PD and Benedicto16, `https://de.wikipedia.org/wiki/Datei:Fire_exit.svg`, CC-BY-SA

Methodology

We explore designs for machines that are effective in solving learning problems of scientific or economic interest, evaluating the designs through mathematical analysis or computational experiments.[3]

[3] Sutton & Barto: Reinforcement Learning - An Introduction (2nd Ed.), 2017, Unpublished, `https://webdocs.cs.ualberta.ca/~sutton/book/the-book-2nd.html` (**SB**)

Formal definition: Markov Decision Process

MDP: Markov Decision Process, one way to formalize the RL problem, a 5-tuple

$$\text{MDP} := (S, A, T, r, \gamma)$$

with states S, actions A, state transition probabilities through an action T, reward r and discount factor γ.

Markov property allows to only consider current $s \in S$ but puts the burden on state representation.

Solving the MDP

The solution to an MDP is a policy π, like $a = \pi(s)$, or $\pi(a|s)$, ideally π^* denoting the optimal policy.

Idea: Guide search for policy with a Value function. Describes the "value" (expected return) of a given state s or state-action pair (s, a) with respect to a given policy.

$$v_\pi(s) = E_\pi(G_t|S_t = s) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right]$$

or

$$q_\pi(s, a) = E_\pi(G_t|S_t = s, A_t = a) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

Bellman equations

Bellman equation for v_π [4]

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

It expresses a relationship between the value of a state and the values of its successor states.

[4] SB pg. 63/81 Eq. 3.12

Bellman optimality equation

Bellman optimality equations for v and q for completeness sake [5]

$$
\begin{aligned}
v_*(s) &= \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]
\end{aligned}
$$

$$
\begin{aligned}
q_*(s, a) &= E[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\
&= \sum_{s',r} p(s', r | s, a)[r + \gamma \max_{a'} q_*(s', a')]
\end{aligned}
$$

which can be turned into iterative update equations for successive approximations of v or q.

[5] SB pg. 79/97, pg 80/98

Three main methods

$\boxed{\text{Dynamic Programming (DP)}}$

$\boxed{\text{Temporal Difference Learning (TD)}}$

$\boxed{\text{Monte Carlo Methods (MC)}}$

TD is a "bridge" connecting DP (bootstrapping) and MC (sampling).

DP solution

Assumes a perfect model = knowledge of T.

Fundamental method: **Policy iteration** [6]: Policy evaluation computing v_π, Policy improvement by adopting any action improving v which is not in the current policy, with alternative correspondences: Evaluation / Prediction, Improvement / Control.

Variant Value iteration [7] is a modified Policy iteration. These modifications lead to Generalized Policy Iteration (GPI): alternating between Policy evaluation and Policy improvement with different grades of convergence for each.

[6] SB pg. p87/105

[7] SB pg. 90/108

Monte Carlo solution

Monte Carlo: requires $\underline{\text{no}}$ model apart from an "explorer" and uses only experience (sampling).

MC prediction: Sample full episode, for each occuring state s accumulate the return from the first occurence of s into v_π.

Policy improvement same as in DP using GPI.

Temporal Difference Learning

Combine MC (sampling) and DP (bootstrapping). Simplest method is TD(0) prediction,

$$V(S) \quad \leftarrow \quad V(S) + \alpha \underbrace{(\overbrace{R + \gamma V(S')}^{\text{target}} - V(S))}_{\text{TD error}}$$

and the variants Q-Learning (off-policy control)

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_{a} Q(S', a) - Q(S, A)]$$

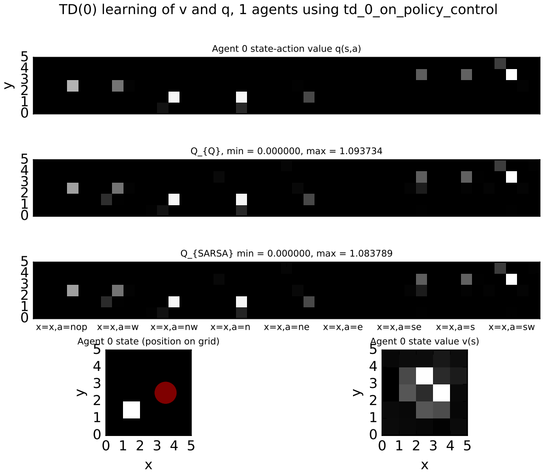
and SARSA (on-policy control)

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$$

Still using GPI.

Temporal Difference Learning example

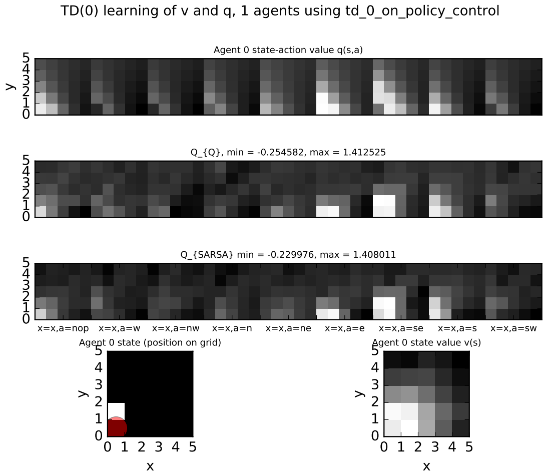
Different v and q functions using vanilla tables, using rl.py from [8]



[8] https://github.com/x75/playground

Temporal Difference Learning example

Same settings, different goal state, *v* and *q* functions using an Multilayer Perceptron approximation.



Continuous state-action spaces

From discrete to continuous spaces: tables can be replaced with learnable Function Approximators (FA) like Neural Networks (Connectionist RL), Gaussian Processes (GP), etc. In this case, the update equations from one slide ago have to be changed to comply with the training paradigm, e.g. for a Q approximator using the SARSA update:

$$\begin{aligned} X &= (S_{t-1}, A_{t-1}) && \text{Input} \\ y &= R + \gamma FA((S_t, A_t)) && \text{Target} \\ FA(X) &\rightarrow y && \text{Update } FA(X) \text{ towards } y \end{aligned}$$

Continuous state-action spaces

Policy search

Use a direct encoding of a policy e.g. parameterized functions of the state $a = \pi(s, \theta)$. Evolutionary algorithms can be seen as policy search.

Actor-Critic methods

On-policy methods, the *actor* proposes an action (= policy), the *critic* evaluates the action and its outcome (value). The the actor is changed using the critic's feedback.

Policy gradient

Sample episodes controlled by current policy while putting noise on the parameters or on the output. Estimate the gradient of the loss with respect to the noise dimensions and update the policy using that gradient.

Dangling bits

Episodic vs. non-episodic

How many lives?

On-policy vs. off-policy

Change behaviour immediately or observe for while and change later.

Exploration / Exploitation

Are the known options good? Any options we haven't tried yet?

Temporal Credit Assignment

Which part of the learner at what time is to blame for good or bad consequences?

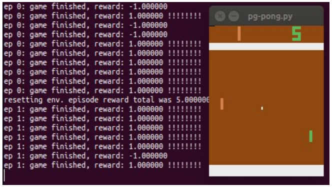
Eligibility traces

Extend the temporal horizon over which action/reward correlations are observed in a continuous manner. Usually parameter λ, with the range $\lambda = 0$ (TD(0)) to $\lambda = 1$ (MC).

Current trends

Many recent results from combining deep learning and reinforcement learning.

Some buzzwords are Neural Fitted Q-Learning, Deep Q-Networks, deep learning of dynamics models, end-to-end reinforcement learning, From pixels to torques, Atari (Pong), OpenAI Gym and Universe, Torchcraft, Malmo, AlphaGO, . . .

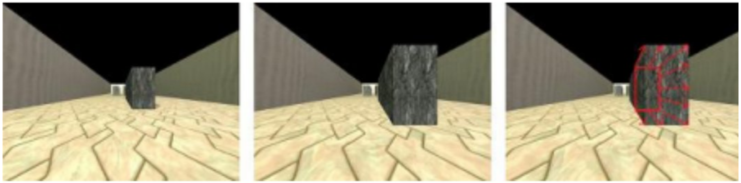


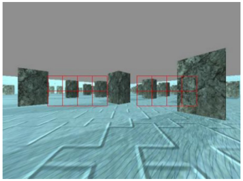
```
ep 0: game finished, reward: -1.000000
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: -1.000000
ep 0: game finished, reward: -1.000000
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: 1.000000 !!!!!!!!
ep 0: game finished, reward: 1.000000 !!!!!!!!
resetting env. episode reward total was 5.000000
ep 1: game finished, reward: 1.000000 !!!!!!!!
ep 1: game finished, reward: 1.000000 !!!!!!!!
ep 1: game finished, reward: 1.000000 !!!!!!!!
ep 1: game finished, reward: 1.000000 !!!!!!!!
ep 1: game finished, reward: 1.000000 !!!!!!!!
ep 1: game finished, reward: -1.000000
ep 1: game finished, reward: 1.000000 !!!!!!!!
```

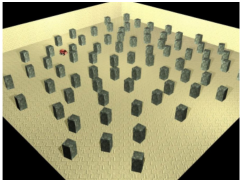
Example: Learning to play Pong with visual input (OpenAI Gym) using Policy gradient.

Example: Policy Gradient with the Signed Derivative

Learning obstacle avoidance with Signed Derivate Policy Gradient method, DA Nestler 2016. Learn linear combination of visual input features to control speed and turning without hitting obstacles.


(a) Optical flow input

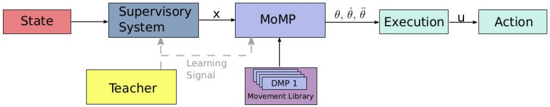

(b) Left/right input

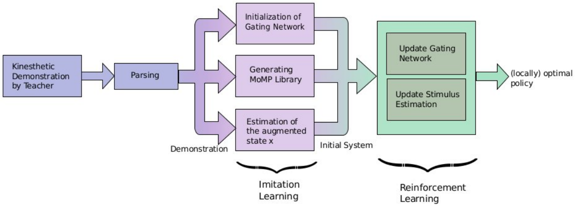

(c) Left/right input

(d) Turtle bot

Example: Policy Gradient with Motion Primitives

Playing table tennis with a robot arm [9] from J. Peters' lab @TU Darmstadt.

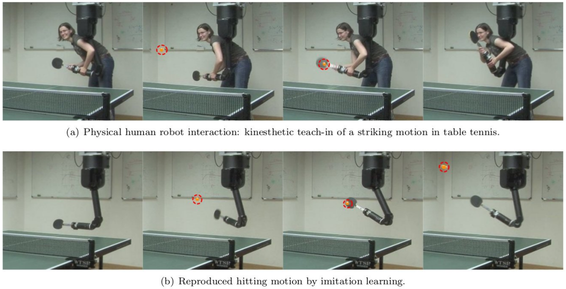




[9] Mülling and others, 2013, Learning to Select and Generalize Striking Movements in Robot Table Tennis, http://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Kupcsik_AAAI_2013.pdf, Videos: https://www.youtube.com/watch?v=SH3bADiB7uQ, https://www.youtube.com/watch?v=BcJ4S4L1n78,

Example: Policy Gradient with Motion Primitives

Playing table tennis with a robot arm [9] from J. Peters' lab @TU Darmstadt.


(a) Physical human robot interaction: kinesthetic teach-in of a striking motion in table tennis.

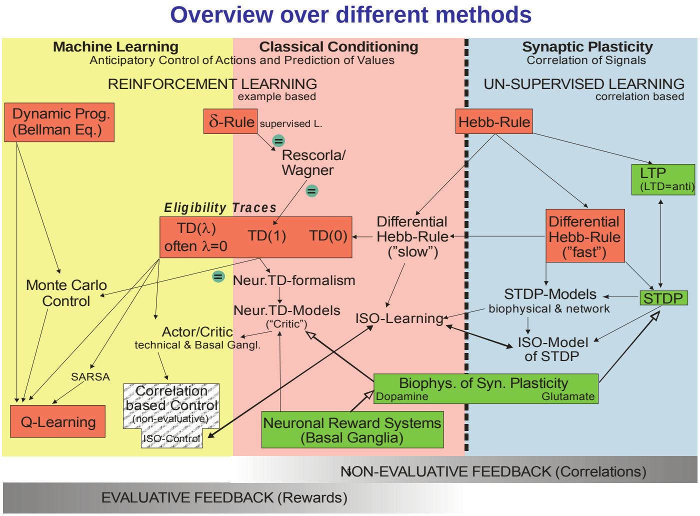
(b) Reproduced hitting motion by imitation learning.

(a) Robot setup

Examples: more of them (incomplete)

- Tak Kit Lau: Prisca: A Policy Search Method for Extreme Trajectory Following
- J. Zico Kolter: Policy Search via the Signed Derivative (PGSD)
- Lau & Liu: Stunt Driving via Policy Search
- Lau & Liu: Learning Hover with Scarce Samples
- Tedrake: Learning to Fly like a Bird
- Michels, Saxena, Ng: High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning
- Rottmann et al.: Autonomous Blimp Control using Model-free Reinforcement Learning
- Riedmiller and others: The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting; Learn to Swing Up and Balance a Real Pole Based on Raw Visual Input Data; Autonomous reinforcement learning on raw visual input data in a real world application;

Overview of different methods and their relationship



Borrowed from F. Wörgötter, Lecture Slides Learning and Adaptive Algorithms 2014, with perms.

Neuromodulation and the reward system

Reward system: Cortico-basal ganglia-thalamic loop

Neuron groups related to *Dopamine*, Glutamate, and GABA, which act as modulators on other groups of neurons.

Temporal difference reward prediction error: unpredicted reward leads to activation, fully predicted reward to no response, omission of predicted reward to depression of activity.

Conditioning

Rescorla-Wagner model of classical conditioning, association between conditioned (CS) and unconditioned (US) stimulus. Looks Hebbian.

$$\underbrace{\Delta V_X^{n+1}}_{\text{weight change}} = \underbrace{\alpha_X}_{\text{CS (pre)}} \underbrace{\beta}_{\text{US (post)}} \underbrace{(\lambda - V_{\text{tot}})}_{\text{stabil. / modul. term}}$$

... the dopamine response seems to convey the crucial learning term $(\lambda - V)$ ... [10]

[10] W. Schultz, `http://www.scholarpedia.org/article/Reward_signals`, 201701

Reward modulation

This modulatory input can be modelled as an additional multiplicative term in a Hebbian update rule.

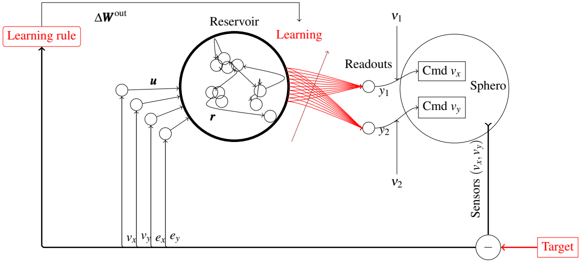
$$dW = \eta \cdot \overbrace{x}^{\text{pre}} \cdot \overbrace{y}^{\text{post}} \cdot \overbrace{r}^{\text{reward}} \cdot \overbrace{\ldots}^{\text{other factors}}$$

Binary reward $\in$ [0, 1] leading to greedy updates toward "improvements", depending on how the reward is computed.

Is faster with dense reward. Learning a value function with FA can be used to obtain a continuous reward prediction from an initially sparse primary reward signal.

Reward modulation example

Exploratory Hebbian Learning e.g. for acquiring behaviour on the Sphero robot [11]. Prior value function for continuous reward. Fast bootstrapping of dynamic control.



[11] Berthold & Hafner, 2015, Closed-loop acquisition of behaviour on the Sphero robot, https://mitpress.mit.edu/sites/default/files/titles/content/ecal2015/ch084.html
https://www.youtube.com/watch?v=NepvHrF7AfU

Summary

Classical RL is complete but at the cost of exhaustive exploration.

Accelerate learning with approximators (NFQ, DMPs, EH, . . .) and on-policy learning at the cost of suboptimality.

Explore sampling with Exploration functions that observe the state of the learner (confidence, error behaviour, information theoretic measures)

Priors and coarse estimates for value functions as initializations?

Temporal Difference- / Prediction-Learning are necessary prerequisites for adaptive behaviour? "Prediction-scope" as a measure of an agent's complexity?

Further reading

- Sutton & Barto, Reinforcement Learning - An Introduction (2nd Ed.), 2017, Unpublished, `https://webdocs.cs.ualberta.ca/~sutton/book/the-book-2nd.html`
- Wörgötter, Lecture Slides: Learning and Adaptive Algorithms (BCCN Göttingen / Dept. Comp. Neurosci)
- Dayan & Abbott, Theoretical Neuroscience, 2000
- Yael Niv, Reinforcement learning in the brain, JMathPsy, 2009
- Hasselt, Insights in Reinforcement Learning, 2010, PhD Thesis
- Riedmiller, 10 Steps and Some Tricks To Set Up Neural Reinforcement Controllers, 2011?
- Deisenroth & others, A Survey on Policy Search for Robotics, 2011
- Kober & others, Reinforcement Learning in Robotics: A Survey, 2012
- Nguyen-Tuong & Peters, Model Learning for Robot Control: A Survey, 2010

Software environments

Some packages that can serve as starting points for exploring closed-loop learning

- pybrain: `http://pybrain.org/`
- openai: gym: `https://gym.openai.com/`, and universe: `https://universe.openai.com/`
- Modular Data Processing Toolkit (steaming hot) with Online and RL Nodes: `https://github.com/varunrajk/mdp-toolkit`
- explauto: `https://github.com/flowersteam/explauto`