Latest Advances in Reinforcement Learning

Johann Lussange, Boris Gutkin

École Normale Supérieure, Paris

5th NUS-USPC Workshop on Machine Learning & Fintech

29-30 November 2017, University Paris Diderot

Contents

Overview of ML

Introduction to RL

Technology Intelligence

Conclusion

ML paradigms

1. **Supervised learning**: mapping inference from a labelled training data set of input-output pairs

2. **Unsupervised learning**: mapping inference from unlabelled data set to its pattern/structure discovery (anomaly detection, PCA…)

3. **Reinforcement learning**: action inference from trial & error in a given environment to maximize an ultimate given reward (c.f. model-based RL, model-free RL)

4. **Multi-Agent Systems (MAS)**: several agents learning a same task together perform better than one, c.f. evolutionary algorithms, asynchronous methods, Multi-Agent Learning (MAL), etc.

Fintech & disruption

- **Next hot topics**:

  1. **Unsupervised learning**, maths research
  2. **Long-term dependencies**, memory networks
  3. **Multi-Task Learning (MTL)**, generalizing quickly from few inputs
  4. **Natural Language Processing (NLP)**, understanding & reasoning
  5. **Deep Reinforcement Learning (DQN)**

- **Next obstacles**:

  - **Training & testing over enough data**, e.g. ImageNet
  - **Memory & planning**, e.g. Monte Carlo Tree Search (MCTS)
  - **Scalability & adaptability**, e.g. physics & ML
  - **Bio correspondence**, synaptic connectivity half of human genome

Contents

Overview of ML

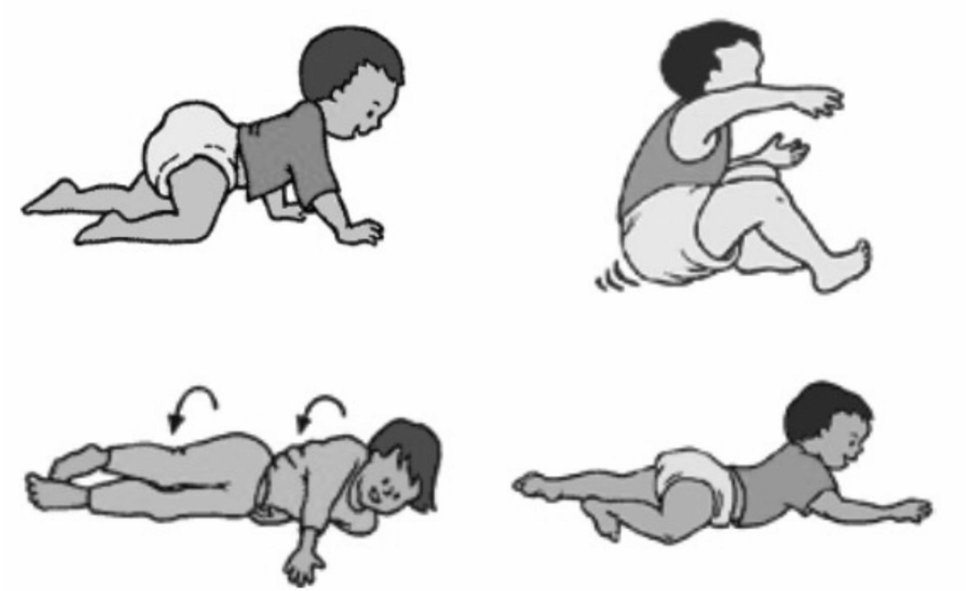
Introduction to RL

Technology Intelligence

Conclusion

Decision theory

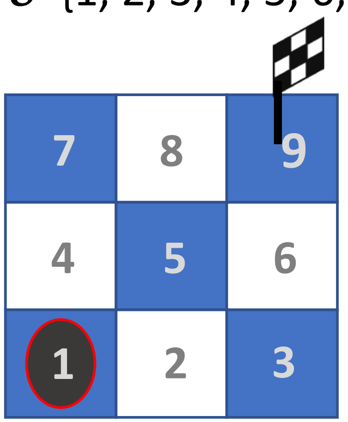
- **Origin**: RL derived from *decision theory*, which studies the reasoning leading to an agent's choices: rational or irrational, active or passive (e.g. pavlovian), with bounded rationality (c.f. game theory, finance)
- **Example**: Perhaps our ‘programming’ is not embedded with supervised crawling but simply with a reward for movement?



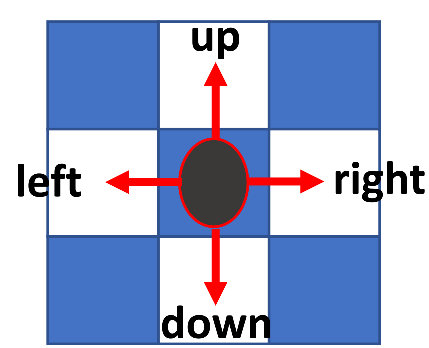
RL inputs

- **States s∈$\mathcal{S}$**: modelling the part of the agent's environment it cannot control, potentially incl. parts of its physical integrity (e.g. sensors, skeleton) or reward system (e.g. battery, stomach).
- **Actions a∈$\mathcal{A}$**: modelling basic low-level controls (e.g. applying a voltage) to high-level decisions (e.g. going to college).
- **Rewards r∈$\mathcal{R}$**: modelling rewards as (negative or positive) scalars, using an extra time-discounting parameter $0<\gamma\leq 1$ to define return.

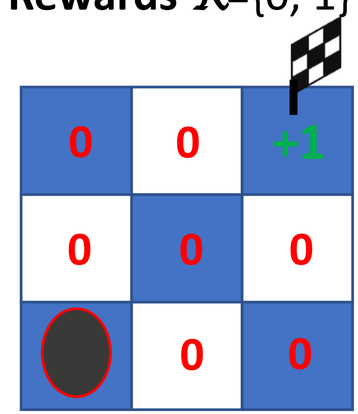
States $\mathcal{S}$={1, 2, 3, 4, 5, 6, 7, 8, 9}



Actions $\mathcal{A}$={right, left, up, down}

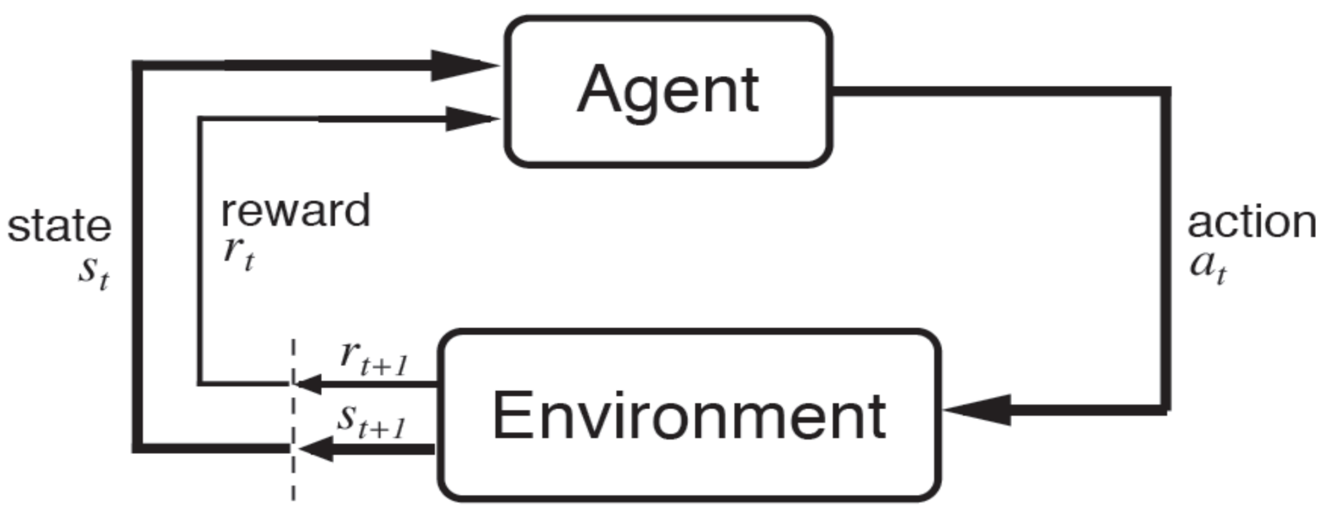


Rewards $\mathcal{R}$={0, 1}



Reinforcement

- **Iteration**: The agent monitors its environment signal s_t, and takes accordingly an action a_t so as to maximize the return R_t.
- **Question**: how does it know which action to take? Via a form of trial & error to infer a mapping from states to rewards (model-based RL), or states-actions pairs to rewards (model-free RL).
- **Exploration vs. exploitation**: the goal of an agent is to *exploit* its policy, but it must first *explore* to find it (e.g. finding restaurant).



MDP

- **MDP**: Like most other ML approaches, RL assumes *Markov state signals*, i.e. the best policy for choosing actions as a function of a Markov state is just as good as the best policy for choosing actions as a function of complete histories.

$$\Pr\{s_{t+1} = s', r_{t+1} = r' \mid s_t, a_t\}$$

$$= \Pr\{s_{t+1} = s', r_{t+1} = r' \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_1, a_1, r_1, s_0, a_0\}$$

- **Non-stationarity**: challenging frameworks when stochastic, uncertain, game theoretic, etc… => c.f. issue of classical N-arm bandit with static vs. dynamic distribution

- **POMDP**: not always possible to know state s_{t+1} after action a_t => but RL requires full observability & states must be history-independent => POMDP methods.

RL functions

Policy $\pi_t(s,a) = \Pr\{a_t = a | s_t = s\}$

Transition probability $\mathcal{P}^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$

Expected value $\mathcal{R}^a_{ss'} = \mathbb{E}\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$

State-value function $V(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right]$

Action-value function $Q(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right]$

Update rule (*example*) $V(s) \leftarrow V(s) + \alpha\left[V(s') - V(s)\right]$

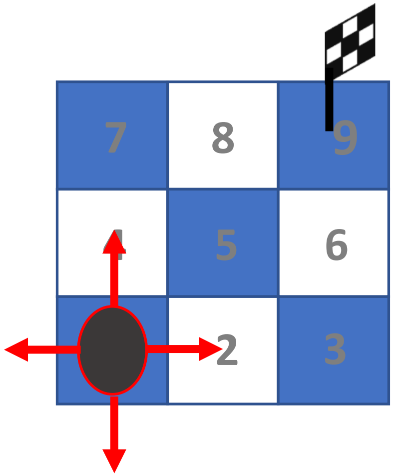
Model-based RL (critic): $\mathcal{P}, \mathcal{R}$ → V → π → $\mathcal{P}, \mathcal{R}$ …

Model-free RL (critic): Q → π → Q → …

Policy search RL (actor): π → …

RL families

- **Model-based RL**: Dynamic Programming (DP) methods, c.f. Bellman's equations => assumes finite MDP.
- **Model-free RL**: Monte Carlo (MC) & Temporal Difference (TD) methods[1], which can be unified with *eligibility traces*, c.f. TD(λ) methods, esp. TD(0) methods & Q-learning => assumes non-finite MDP.
- **Actor-Critic RL**: Policy gradient reinforcement learning, c.f. Finite-Difference Methods, Likelihood Ratio Methods, etc.



[1] *Schultz, W., Dayan, P. & Montague, P. R. A neural substrate of prediction and reward. Science 275, 1593–1599 (1997).*

Features (i)

Feature #1: Exploration & exploitation

- **Methods**: Each action a depends on how good the policy π(s,a), so when should the agent explore new policies & exploit already found policies? Methods of *ε-greedy, softmax, pursuit, on/off-policy learning*, etc...

- **GPI**: The goal is to achieve *convergence* to optimal policy π*****(s,a) via *Generalized Policy Iteration* (GPI) theorem[2], by doing a policy evaluation (i.e *prediction problem*) and then policy improvement (i.e *control problem*) repeatedly unto convergence to Q* and π*.

$$E(\pi_0) = I(V^{\pi_0}) = E(\pi_1) = I(V^{\pi_1}) = E(\pi_2) = \cdots = E(\pi^*) = V^*$$

[2] *R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA, A Bradford Book (1998)*

Features (ii)

Feature #2: **Curse of dimensionality**

- **Continuous v.s. discrete**: Since RL works with (s,a) or (s,a,r), the exploration becomes quickly intractable in real-world applications.
- **Function Approximation**: Major breakthrough historically reached with Watkin's Q-learning[3], links to Artificial Neural Networks, and especially Deep Reinforcement Learning.
- **Action space**: one can consider the action set space also from a tree perspective, c.f. hierarchical RL, MCTS.
- **Models**: model-based RL (planning methods to model $\mathcal{S}$), model-free RL (learning methods to model $\mathcal{S}$ x $\mathcal{A}$)

---

[3] *C. J. Watkins, Learning from delayed rewards, PhD thesis, Kings College, Cambridge (1989)*

Features (iii)

Feature #3: **Temporal credit assignment problem**

- **Core RL theory**: one should only define the final reward (what to achieve), never the intermediary rewards (how to achieve), c.f. chess, c.f. *shaping rewards*, *homeostatic RL*.

- **Delayed reward**: bio correspondence taking into account utility via time-discounting parameter $0<\gamma\leq 1$ to define return, c.f. economic utility & Saint-Petersburg paradox.

- **Regret**: difference between optimal return and actual return.

Contents

Overview of ML

Introduction to RL

Technology Intelligence

Conclusion

Technology intelligence

RL features v.s. research fields:

1. **Exploration-exploitation** => policy learning
2. **Dimensionality** => state representation
3. **Temporal credit** => modular/hierarchical RL, inverse RL, homeostasis

Policy learning (i)

Direct policy search: maximizing the return by searching the subset of policy space (c.f. stochastic optimization problem), while traditional value function approximation derives policies from the value function (requiring more parameters)

- **Policy gradient-free methods**
- **Policy gradient methods**: optimizing a parametrized control policy with respect to the return by gradient descent
  - **Finite-Difference Methods**
  - **Likelihood Ratio Methods /REINFORCE**
  - **Natural Policy Gradients, Stochastic Policy Gradients, Deterministic Policy Gradients**[4] (operating over continuous action spaces, e.g. pole balance, robotics).

---

[4] *D. Silver et al., ICML 2014*

Policy learning (ii)

- **Transfer RL**: transferring experience gathered from one task to another task[5].
- **Imitation RL**: learning a task from observing another agent.
- **Self-play RL & Multi-Agent Learning (MAL)**: agent policy learnt by playing[6] against another agent which also learns, e.g. game theoretic framework, TD-Gammon, AlphaGo.
- **Multitask/asynchronous RL**: multitask RL is learning multiple tasks & exploiting their similarity to improve single-task learning performance. Asynchronous RL[7] executes in parallel many instances of an agent while using a shared model, thus obtaining data diversification, c.f. DeepMind's A3C asynchronous DL method for Atari games.
- **Modular RL**: dividing task into smaller subtasks that individually learn their own policy by RL, and then constructing a global policy by combining them all, in general via a centralized arbitrator[8].

---

[5] *A. Lazaric "Transfer in RL: A Framework and a Survey", Springer (2012)*

[6] *J. Heinrich & D. Silver, "Deep RL from Self-Play in Imperfect-Information Games" (2016)*

[7] *V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning" (2016)*

[8] *J. Andreas et al. "Modular Multitask Reinforcement Learning with Policy Sketches" (2017)*

Policy learning (iii)

- **Lifelong RL**: learning multiple consecutive tasks sequentially[9], c.f. issues of forgetting outliers, remembering efficient information.
- **Multi-step RL**: unifying according to a parameter different RL methods that are part of a given family[10], e.g. unifying TD methods like Sarsa, Q-learning, and Expected Sarsa.
- **Actor-critic**: Critic-only (or value-based) methods estimate the value function while the policy is implicit, while actor-only (or policy-based) methods estimate the policy function without the value function. Actor-critic methods learn the value function in order to then update the policy, c.f. GPI theorem.

---

[9] *C. Tessler et al. "A Deep Hierarchical Approach to Lifelong Learning in Minecraft" (2016)*

[10] *De Asis et al. "Multi-step Reinforcement Learning: A Unifying Algorithm" (2017)*

State representation (i)

- **End-to-end & Deep RL**: ANN trained via an RL approach, allowing the DQN to learn policies directly from high-dimensional input, e.g. Atari games[11], later on coupling it with MCTS, e.g. AlphaGo[12], poker[13]
- **Adversarial RL**: huge gap between simulation to real world RL because of generalization failure & data scarcity => modelling uncertainties via an adversarial agent that applies perturbation to the system, potentially learning to efficiently do that by RL[14]

---

[11] *V. Mnih et al. "Human-level control through deep reinforcement learning" (2015)*

[12] *D. Silver et al. "Mastering the game of Go with deep neural networks and tree search" (2016)*

[13] *J. Heinrich "Smooth UCT Search in Computer Poker" (2015)*

[14] *L. Pinto et al. "Robust Adversarial Reinforcement Learning" (2017)*

State representation (ii)

Uncertain/Partial/Biased information: MDPs assume the agent knows the complete state of the environment, which is highly unrealistic (e.g. robot within a room)

- **POMDPs (Partially Observable MDP) models**: specifying a function from the hidden state to the observables, by finding a mapping from observations (or an MDP constructed belief state, but not real state!) to actions => difficult to construct.
- **BA-POMDPs (Bayes Adaptive POMDP) models**: learning this POMDP model during execution via a Bayesian approach[15] => intractable in non-trivial domains
- **BA-POMCP (Bayes-Adaptive Partially Observable Monte-Carlo Planning)**: extending Monte Carlo Tree Search (MCTS) to solve BA-POMDPs[16].

---

[15] *S. Ross et al. "A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes" (2011)*

[16] *S. Katt et al. "Learning in POMDPs with Monte Carlo Tree Search" (2017)*

Credit assignment

- **Shaping rewards**: incorporating background knowledge on sub-rewards[17] in order to improve convergence rates, e.g. robotics.
- **Hierarchical RL**: defining reward via temporal abstraction[18].
- **Homeostatic RL**: defining reward as a manifold of several sub-rewards[19], e.g. biological correspondence of temperature, water, food, etc.
- **Inverse/apprenticeship RL**: extracting reward function from observed optimal behaviour[20].

---

[17] *A. Y. Ng et al. "Policy invariance under reward transformation: theory and application to reward shaping" (1999)*

[18] *K. Frans et al. "Meta Learning Shared Hierarchies" (2017)*

[19] *M. Keramati & B. Gutkin "A Reinforcement Learning Theory for Homeostatic Regulation" (2011)*

[20] *P. Abbeel, A. Coates, A. Ng, "Autonomous Helicopter Aerobatics through Apprenticeship Learning," vol. 29, Issue 13 IJRR (2010)*

Contents

Overview of ML

Introduction to RL

Technology Intelligence

Conclusion

Conclusion

RL basics

- **RL inputs**: states $\mathcal{S}$, actions $\mathcal{A}$, rewards $\mathcal{R}$
- **RL types**: model-based, model-free, policy search
- **RL features**: exploration-exploitation, curse of dim., reward estimation

RL research

- **Policy learning**: policy-gradient methods, multi-task/asynchronous methods, MAL/self-play methods
- **State representation**: deep reinforcement learning, BA-POMDP models
- **Credit assignment**: modular/hierarchical RL, inverse RL, homeostatic RL

Conclusion

- Recent spectacular results simply mix methods within RL (eligibility traces, multi-step RL), or mix RL with other ML methods (DQN, A3C/MAL, MCTS)

Thank you for your attention