A Brief Survey of Reinforcement Learning

Short walk-through on building learning agents

Giancarlo Frison

December 12, 2018

Introduction

What is Reinforcement Learning

[9]

REINFORCEMENT LEARNING is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Challenges in RL

RL vision is about creating systems that are capable of learning how to adapt in the real world, solely based on trial-and-error. Challenges:

Reward signal
The optimal policy must be inferred by interacting with the environment. The only learning signal the agent receives is the reward.

Credit assignment problem
Agents must deal with long-range time dependencies: Often the consequences of an action only materialise after many transitions of the environment [8].

Non stationary environments
Especially in multi-agent environments, in the same circumstances of state s_t and action a_t outcomes might be different.

Difference from other optimizations

There are major differences within:

- Supervised learning

Difference from other optimizations

There are major differences within:

- Supervised learning
- Mathematical optimization

Difference from other optimizations

There are major differences within:

- Supervised learning
- Mathematical optimization
- Genetic programming

Genetic Programming

Genetic Programming

GP is a technique whereby computer programs are encoded as a set of genes that are then modified using an evolutionary algorithm.

- A number of chromosomes are randomly created.
- Each chromosome is evaluated through a fitness function.
- Best ones are selected, the others are disposed.
- Chromosomes could be breded among the selected for a new generation.
- Offsprings are randomly mutated.
- Repeat until the score threshold is reached [3].

Crossover

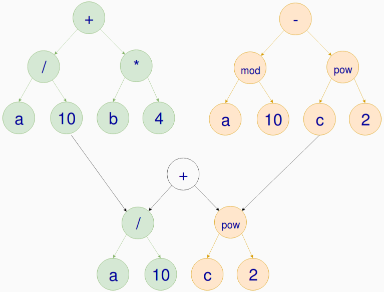


Figure 1: The "breeding" is called crossover. The chromosome (in this case an AST), is merged between two individuals for searching a better function.

Strenghts on many local minima

GP may overtake gradient-based algorithms when the solution space has many local minima

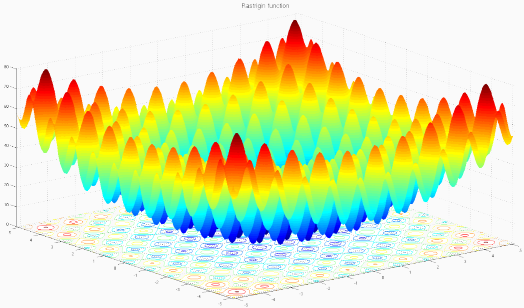


Figure 2: Rastrigin function

Strenghts on no-differentiable functions

$f(x)$ is DIFFERENTIABLE when it has always a finite derivative along the domain.

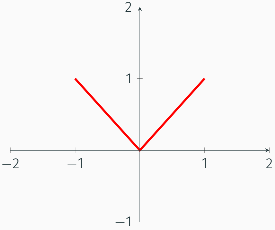


Figure 3: $f(x) = |x|$ has no derivative at $x = 0$.

GP is tollerant to *latent* no-differentiable functions.

Multi-armed Bandit

Multi-armed Bandit

The multi-armed bandit problem has been the subject of decades of intense study in statistics, operations research, A/B testing, computer science, and economics [12]

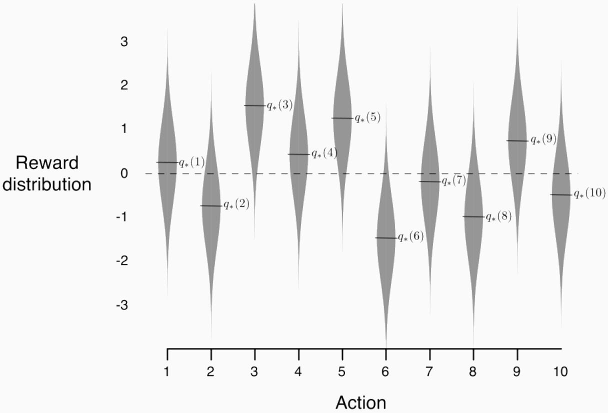


Figure 4: Obtained measures after repeated pullings with 10 arms [8].

Q Value's Action

$\mathbf{Q_n}$ is the estimated value of its action after n selections.

$$Q_{n+1} = \frac{R_1 + R_2 + ... + R_n}{n}$$

A more scalable formula, updates the average with incremental and small constant:

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$$

General expression of the bandit algorithm, recurrent pattern in RL (*Target* could be considered the reward R by now).

$$\mathrm{NewEstimate} = \mathrm{OldEstimate} + \mathrm{StepSize}(\mathrm{Target} - \mathrm{OldEstimate})$$

Gambler's Dilemma

When pulled, an arm produces a random payout drawn independently of the past. Because the distribution of payouts corresponding to each arm is not listed, the player can learn it only by experimenting.

Exploitation
Earn more money by exploiting arms that yielded high payouts in the past.

Exploration
Exploring alternative arms may return higher payouts in the future.

There is a *tradeoff* between EXPLORATION and REGRET.

Markov Decision Process

Definition of MDP

Figure 5: 2015 DARPA Robotics Challange [6]

Despite as in bandits, MDP formalizes the decision making (Policy π) in sequential steps, aggregated in Episodes.

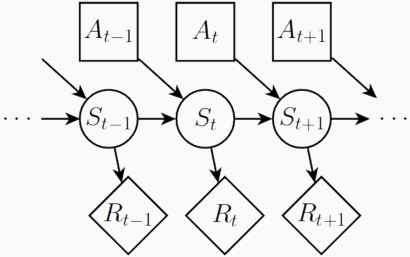


Figure 6: Model representation of MDP [5]

MDP strives to find the best π to all possible states. In Markov processes, the selected action depends **only** on current state.

Bellman equation

In a multi-step process the total return G is the *discounted* sum of rewards, from the begin state to the terminal state:

$$G_\pi = r_0 + \gamma r_1 + \gamma^2 r_2 + ...\gamma^n r_n$$

BELLMAN EQUATION: The value of a state V_s is the *averaged* reward obtained in that state plus the sum of all values following π thereafter.

The goal of RL is to find the *optimal* π with highest G

$$\pi = argmax \sum_s r + \gamma v_\pi(s)$$

Grid World

🏁	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	🏁

- $A = up, down, right, left$
- Terminal states are the flagged boxes.
- $R_t = -1$ for all states.

The problem is to define the best $\boldsymbol{\pi}$. Value function is computed by iterative policy evaluation.

Iteration 1

Calculated V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

Policy π_1

🏁	←	?	?
↑	?	?	?
?	?	?	↓
?	?	→	🏁

Iteration 2

Calculated V_2

0	-1.7	-2	-2
-1.7	-2	-2	-2
-2	-2	-2	-1.7
-2	-2	-1.7	0

Policy π_2

🏁	←	←	?
↑	↵	?	↓
↑	?	↳	↓
?	→	→	🏁

Iteration 3

Calculated V_3

0	-2.4	-2.9	-3
-2.4	-2.9	-3	-2.9
-2.9	-3	-2.9	-2.4
-3	-2.9	-2.4	0

Policy π_3

🏁	←	←	←↓
↑	←↑	←↓	↓
↑	↓→	↓→	↓
↑→	→	→	🏁

Model-free learning
The learning comes only through raw experiences (*episodes*) when there is no prior knowledge of the environment.

Bootstrapping
Value estimations are based partly on other learned estimations.

Bootstrapping helps to increase the stability of estimations.

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

Neural Networks in RL

Beyond the Gridworld

In the Gridworld every state value v_t is stored in a table. The approach lacks scalability and is inherently limited to fairly low-dimension problems.

Figure 7: The state v_t might be a frame of a videogame [11]

Deep Reinforcement Learning

Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces. [1]

- Universal function approximator Multilayer neural networks can approximate any continuous function to any degree of accuracy [4].

Deep Reinforcement Learning

Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces. [1]

- Universal function approximator Multilayer neural networks can approximate any continuous function to any degree of accuracy [4].
- Representation learning Automatically discover the representations needed for feature detection or classification from raw data, also known as FEATURE ENGENEERING [2].

Neural networks can learn the approximated value of V_s or $Q(s, a)$ for any given STATE/ACTION SPACE.

Discrete space

When the ACTION SPACE is limited by a set of options, *(e.g. turn right, left)* the output layer could be a set of neurons as the number of possible actions.

A1	A2	A3	A4
0.21	0.27	**0.30**	0.21

Figure 8: Output layer with softmax

The choosed action is the one with the biggest Q_t (A3).

Exploration in discrete space

The balance between EXPLORATION and EXPLOITATION is one of the most challanging task in RL. The most common is the **ε-greedy**:

$$\pi(s) = \begin{cases} \text{random action}, & \text{if } \xi < \epsilon. \\ argmaxQ(s,a), & \text{otherwise}. \end{cases} \tag{1}$$

With ε-GREEDY, at each time step, the agent selects a random action with a fixed probability, $0 \leq \epsilon \leq 1$, instead of selecting the learned optimal action.

ε-GREEDY must be tuned and it is fixed during the training. In contrast, VALUE-DIFFERENCE BASED EXPLORATION adapts the exploration-probability based on fluctuations of confidence [10].

A1	A2	A3	A4
0.21	0.27	**0.30**	0.21

Figure 9: Low confidence, flat likelihood

A1	A2	A3	A4
0.21	0.07	**0.70**	0.01

Figure 10: High confidence, sharp likelihood

Continuous space

Applications of RL require continuous state spaces defined by means of continuous variables *e.g. position, velocity, torque.*

Single Output
Like in DDPG algorithm. The NN yields only one output.

Probability distribution
The network yields a distribution. For convenience usually it is a NORMAL DISTRIBUTION defined by 2 outputs: μ and σ.

NN comes into play by approximating state-action value pairs $Q_t(s,a)$.

Double network
Other the main network, a sibling network (TARGET) is used in training for adjusting values Q, the second network is periodically aligned the the main network.

Priority experience replay
Past state/action/reward tuples are kept also for future iterations. Tuples with higher error in evaluation are retained longer.

First disruptive advance in DEEP RL in 2015 by DeepMind [7].

Actor-Critic

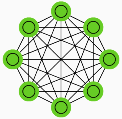
Policy-based algorithm

ACTOR-CRITIC methods lay in the family of POLICY-BASED algorithms. Differently by VALUE-BASED ones - like DQN - the main network (ACTOR) learns directly the action $\mathsf{a}_\pi(\mathsf{s})$ to return for a given state s.

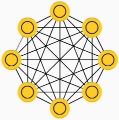
Therefore there is no $argmax\, Q_\pi(s, A)$ among all $Q_\pi(s)$ for a given state

Actor-Critic

ACTOR-CRITIC defines an architecture based on 2 parts:



Actor
Main network which infers the action for a given state
($s_t \rightarrow a$)



Critic
Secondary network used only for training, evaluates the ACTOR output returning the *averaged* value of state.
$s_t \rightarrow V(s)$

Advantage function for incremental learning

The ADVANTAGE technique is used to give a evaluation of the last action *compared* to the average of other actions.

Advantage function for incremental learning

The ADVANTAGE technique is used to give a evaluation of the last action *compared* to the average of other actions.

It answers the following question:

`Is last action more rewarding than others in such scenario?`

Advantage function for incremental learning

The ADVANTAGE technique is used to give a evaluation of the last action *compared* to the average of other actions.

It answers the following question:

```
Is last action more rewarding than others in such scenario?
```

Yes
Action's likelihood is encouraged. Gradients are pushed *toward* this action.

No
Action's likelihood is discouraged. Gradients are pushed *away* from this action.

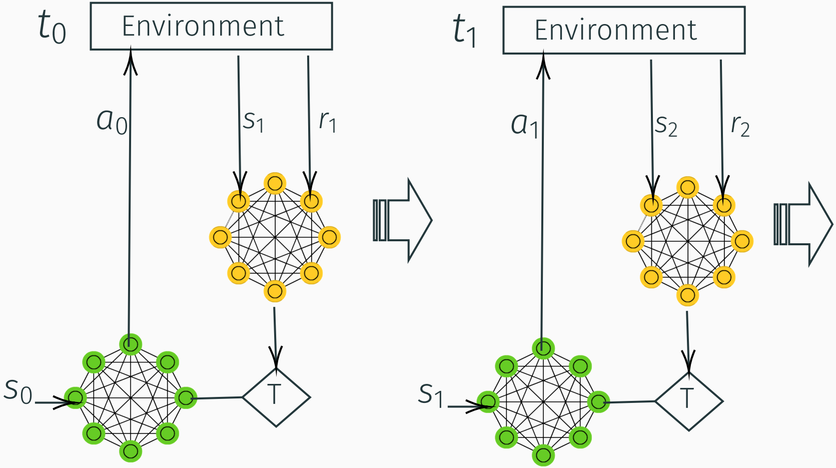
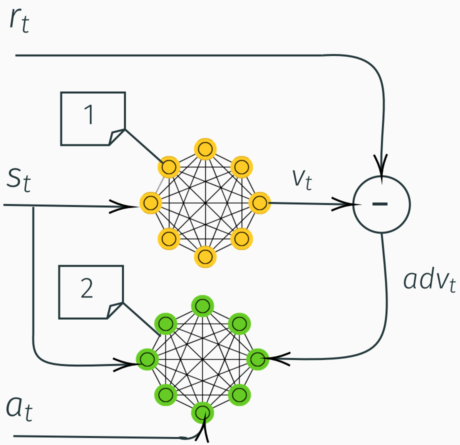


Figure 11: Actor-Critic interactions in an iterative training

Actor training



- 1 Critic returns the averaged value of the state.
- 2 Actor is trained considering s_t, a_t and $adv_t = r_t - v_t$

Deep Deterministic Policy Gradient

DDPG

Especially effective in *continuous space*. Derived from ACTOR-CRITIC and DQN it uses 4 networks.

Value/action networks
Likewise AC there is a distinction between value and action networks.

Duplication of AC networks
For stability those networks are duplicated.

Replay buffer
Likewise DQN an history of past experiences is reused.

Despite others tecniques, DDPG improves the actual π by applying the gradient obtained during the training iteration, from the value network to the output network.

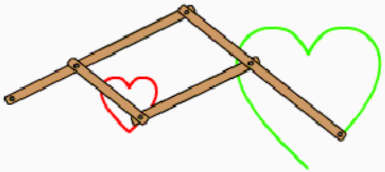


Figure 12: Like with a pantograph, the gradient V_s is applied to the π network for a similar effect, without knowing the value of π.

Asgard

Missing slides for Intellectual Property

Demo

References i

- K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath.
  A brief survey of deep reinforcement learning.
  arXiv preprint arXiv:1708.05866, 2017.
- Y. Bengio, A. Courville, and P. Vincent.
  Representation learning: A review and new perspectives.
  IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828, 2013.
- Giancarlo Frison.
  First Steps on Evolutionary Systems.
  2018.
  link https://labs.cx.sap.com/2018/07/02/first-steps-on-evolutionary-systems/.

References ii

K. Hornik, M. Stinchcombe, and H. White.
Multilayer feedforward networks are universal approximators.
Neural networks, 2(5):359–366, 1989.

J. Zico Kolter.
Markov Decision Process.
http://www.cs.cmu.edu/afs/cs/academic/class/15780-s16/www/slides/mdps.pdf.

John Williams.
2015 darpa robotics challange.
Public domain.

References iii

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529, 2015.

P. Montague.
Reinforcement Learning: An Introduction, by Sutton, R.S. and Barto, A.G.
Trends in Cognitive Sciences, 3(9):360, 1999.

Omanomanoman.
CC BY-SA 4.0, https://commons.wikimedia.org/.

References iv

- M. Tokic and G. Palm.
  Value-difference based exploration: adaptive control between epsilon-greedy and softmax.
  In *Annual Conference on Artificial Intelligence*, pages 335–346. Springer, 2011.
- Valve Corporation.
  A screenshot from the video game Dota 2.
- Yamaguchi??.
  CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)].