Reinforcement Learning

Andrea Bonarini

Artificial Intelligence and Robotics Lab
Department of Electronics and Information
Politecnico di Milano

E-mail: bonarini@elet.polimi.it
URL:http://www.dei.polimi.it/people/bonarini

Machine Learning

What is it?

- identify a **model** automatically, without any human intervention

In this course, we are only concerned with three of the many techniques:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

Input:

samples, input/output pairs for the model, either correct or wrong

How does it operate?

It changes the model structure to average among the examples, so that the model gives the expected output also when input different from the samples is presented: focus on generalization

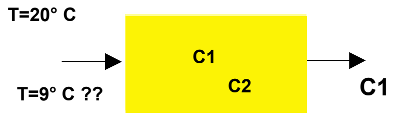
Unsupervised Learning

Input:

samples of model input

How does it operate?

It changes the model so to identify the peculiar features of the input. Focus on classification



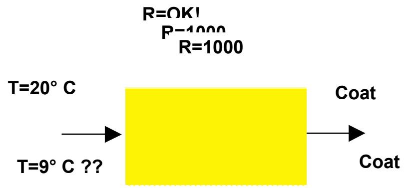
Reinforcement learning

Input:

evaluation of model performance (reinforcement)

How does it operate?

it favors best parts of the model against the worst ones (block composition hypothesis): focus on performance

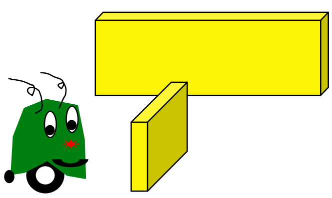


Reinforcement learning: reinforcement

Let's focus ideas on an example:

An autonomous robot has to learn to perform a task

E.g.: move in the environment while avoiding obstacles



What do we have to learn?

The control system of the robot: an input-output model.

Supervised learning??

Yes, if we have examples of good control

No, if we cannot collect or define good examples

Unsupervised learning??

Too poor.
Good only to classify input, not to associate output

Reinforcement learning??

Yes. if we can provide an evaluation of behavior

Reinforcement?

A reinforcement is provided to evaluate the robot behavior

E.g.: r_t=0 if the robot hits something at time t

r_t=1 if the robot moves without hitting anything

Reinforcement learning: model

The model provides a relationship between input and output

In our example, the input are sensorial data to describe the environment, and the output the actions taken by the robot

We have to learn the model (possibly from scratch)

Representation

In order to be able to learn in a reasonably short time, a reasonably compact representation is preferred.

Let's consider to have a small number of possible actions (e.g., forward, back, left, right), and a small number of input values (e.g., in which of the 8 main directions is detected an obstacle at a predefined distance, say, 50 cm).

In this case, the model could be a table that defines a relationship among input configurations and actions. Each element of the table represents the value of the corresponding action for that configuration.

	FW	BW	R	L
Front	0	1	0.3	0.2
Back	1	0	0.4	0.6
Left...	...	...		

The reinforcement learning activity

Two main points:

- **Reinforcement distribution**: how to distribute reinforcement to the elements of the table?
- **Policy**: how to select among alternative actions while learning?

Reinforcement distribution: aims

First of all, we have to define the learning aims

Maximize reinforcement in the long term

... but what reinforcement?

Infinite horizon expected reinforcement $E\left[\sum_{k=0}^{\infty} r_{t+k+1}\right]$

Finite horizon expected reinforcement $E\left[\sum_{k=0}^{T} r_{t+k+1}\right]$

Average reinforcement $E\left[\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n} r_{t+k+1}\right]$

Discounted expected reinforcement $E\left[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}\right]$

What should we learn?

Value function

value of the states (input)

$$V : S \rightarrow \Re$$

E.g., How much is good to face a wall?

Action-value function

values of the state-action pairs

$$Q : S \times A \rightarrow \Re$$

E.g., How much is good to move back when facing a wall?

Updating the value function

We have to consider:

- the so-far <u>accumulated experience</u>
- the <u>present performance</u>

A convex formula is preferred, since it keeps values bounded:

$$v_{t+1}(s) = (1-\alpha)v_t(s) + \alpha\, \Delta_{t+1}$$

where $0<\alpha\leq 1$ and

Δ_{t+1} represents the "value" of the present performance

Analogous formula is used for the Q function.

Reinforcement distribution: Q-learning

Q-learning is the most popular reinforcement learning algorithm

Q-learning is designed to find the policy to optimize the future discounted reward

$$E\left[\sum_{k=0}^{\infty}\gamma^k \cdot r_{t+k+1}\right] = r_{t+1} + \gamma \cdot E\left[\sum_{k=0}^{\infty}\gamma^k \cdot r_{t+k+2}\right]$$

We want to optimize this reward, so we might assume that, once learned:

$$E\left[\sum_{k=0}^{\infty}\gamma^k \cdot r_{t+k+2}\right] = \max_{a\in A} Q(s_{t+1}, a)$$

So, the quantity to be optimized can be represented as:

$$E\left[\sum_{k=0}^{\infty}\gamma^k \cdot r_{t+k+1}\right] = r_{t+1} + \gamma \max_{a\in A} Q(s_{t+1}, a)$$

Q-learning: updating formula

The general updating formula for Q values is:

$$Q_{t+1}(s_t, a_t) = (1-\alpha) Q_t(s_t, a_t) + \alpha\, \Delta_{t+1}$$

If we imagine to have Q…

$$Q_{t+1}(s_t, a_t) = (1-\alpha) Q_t(s_t, a_t) + \alpha\, (r_{t+1} + \gamma \cdot \max_{a\in A} Q(s_{t+1}, a))$$

Δ_{t+1} depends on the reinforcement obtained and on the reached state

Q-learning is proved to converge if $t\rightarrow\infty$ while $\alpha \rightarrow 0$

Q-learning algorithm

```
1.   t:=0
2.   for a∈A, s∈S  || random initializazion of Q-values
3.         Q(s,a) := random();
4.   st:=random(S);  || start from a random state
5.   repeat
6.     if t > 0 {  || update Q-value of the current (st,at)
7.           Δ := rt+g*max_{a∈A} (Q(snext,a));
8.           Q(st,at) := (1-a)*Q(st,at)+a*Δ;
9.           st:= snext}
10.    at:= select(A, Q(st,.));  || select an action
11.    send(at);  || perform the action
12.    get(snext);  || get the state where the action brought
13.    rget(rt);  || get the obtained reinforcement
14.    t := t+1;
15.  until stop();
```

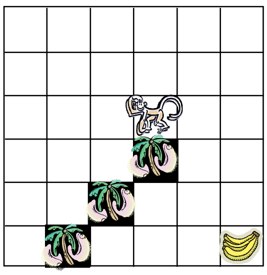

Policy and exploration

A policy is a set of actions to be done in the states.
We would like to learn the optimal policy.

It is wrong to apply (at point **10** of the algorithm) the optimal policy found since a given moment, since it might delay the identification of the really optimal policy

State (3,4)
Action West
Value 0.7

What if it was:
State (3,4)
Action East
Value 0.3 ????



This action would never be selected

Exploration vs. Exploitation

Usually, the actions to be tested are selected by considering their values, but selecting them in probability, so exploration is preferred when actions are more or less equivalent, and the so-far best action is preferred (in probability) only if it really emerges.