Machine Learning in Robotics

Nuno Lau

Universidade de Aveiro/DETI
IEETA

Robótica Móvel e Inteligente
17/11/2023

Outline

Outline

Motivation

Programming robots is a hard work!

- No high-level programming language;
- Sensors and actuators are noisy;
- Robotics is moving towards increasingly unstructured environments.

If only robots could learn how to perform tasks by themselves. . .

Machine Learning

- Machine Learning is:
  - “A **computer program** is said to **learn** from **experience** E with respect to some **class of tasks** T and **performance measure** P, if its performance at tasks in T, as measured by P, **improves with experience** E” *Tom Mitchell. Machine Learning, 1997*
- Key concepts:
  - Experience (data);
  - Task;
  - Performance Measure (metric);
  - Improvement
- Machine Learning can be seen as a search problem of finding a policy that maps states to responses, ($\pi : S \rightarrow R$), to perform a desired task.

Machine Learning

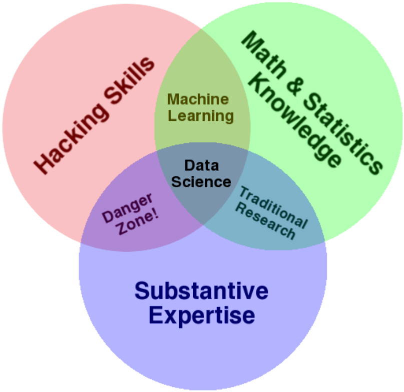


Figure: Data Science Venn Diagram, by Drew Conway

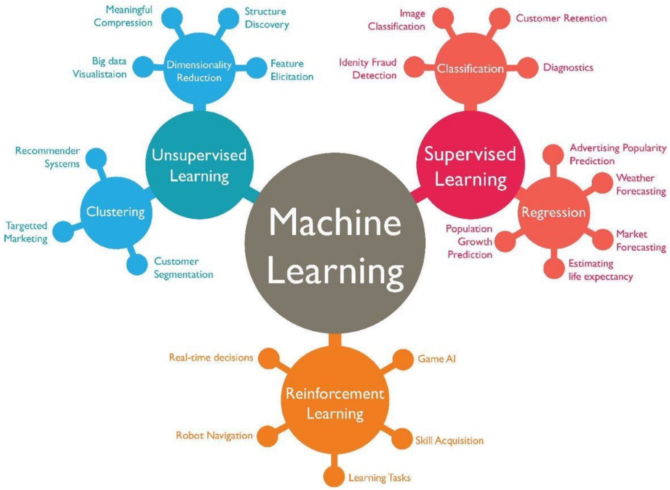
Challenges in Robot Learning

- Limited data;
- Generalization;
- Curse of Dimensionality;
- Which is the best action to take?
- Different Machine Learning paradigms
  - Supervised;
  - Unsupervised;
  - Semi-supervised Learning

Machine Learning Paradigms

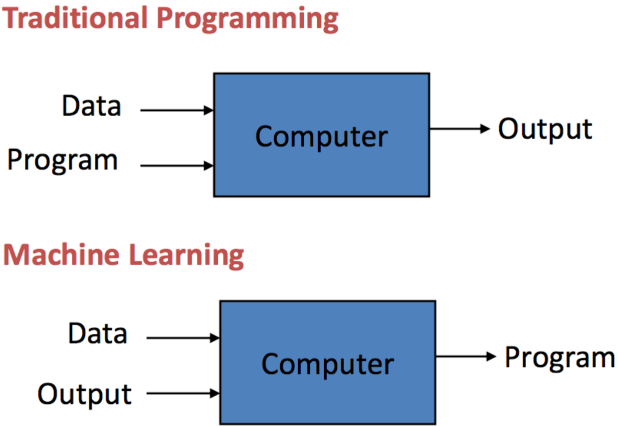
- Supervised Learning
  - Learn from examples
- Unsupervised Learning
  - Discover structure in data
- Reinforcement learning
  - Learn from interaction

Machine Learning Paradigms



Outline

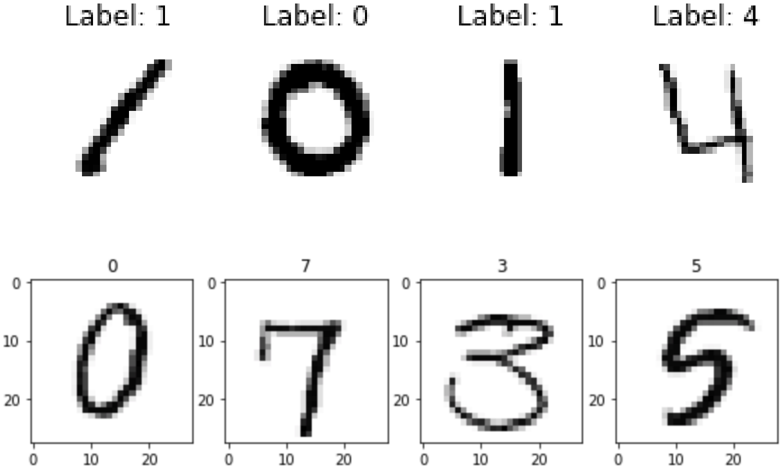
Supervised Learning



Supervised Learning

- A “teacher” **provides** training data consisting of **states** (S) and the **desired response** (R).
  - The learning process knows the desired output for several inputs.
  - The supervisor indicates what is the best action to take (sometimes).
- The robot must learn to **fit** the training data data and **generalize** a policy to the states not covered by the training data.
- Common methods:
  - k Nearest Neighbor, Logistic Regression, Neural Networks, Decision Trees, Support Vector Machines, Gaussian Processes, . . .

Supervised Learning



Supervised Learning

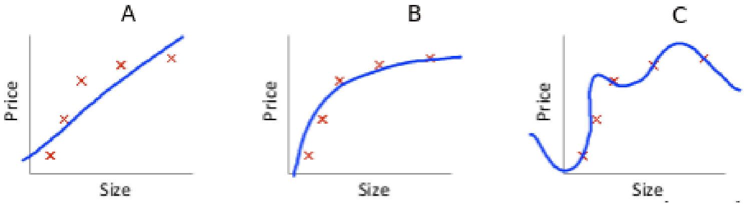
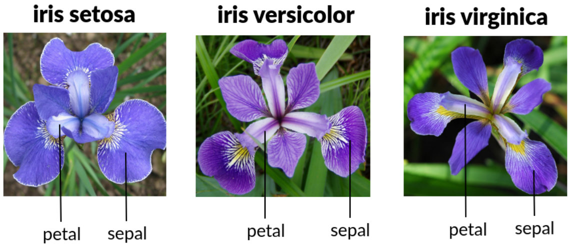


Figure: A regression problem: bias vs. variance

Supervised Learning

IRIS dataset

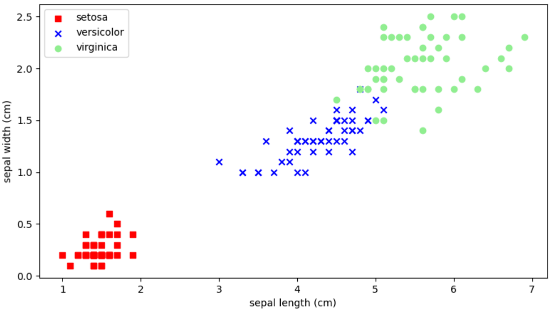
- Find flower species from:
  - Sepal width
  - Sepal length
  - Petal width
  - Petal length



Supervised Learning

IRIS dataset

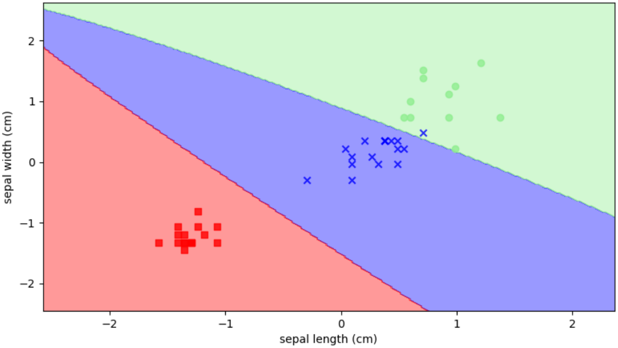
- Considering only:
  - Sepal width
  - Sepal length



Supervised Learning

IRIS dataset - SVM

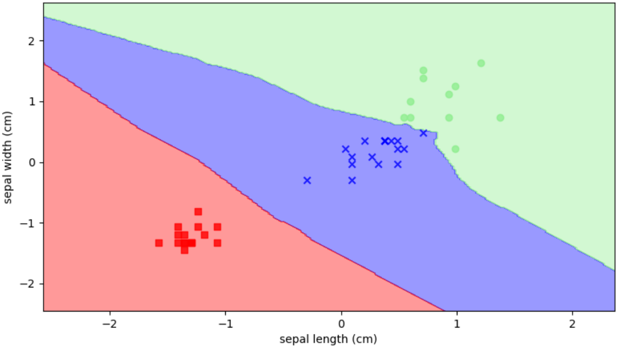
- Support Vector Machine Classification
  - Find the maximum margin hyperplane that separates categories



Supervised Learning

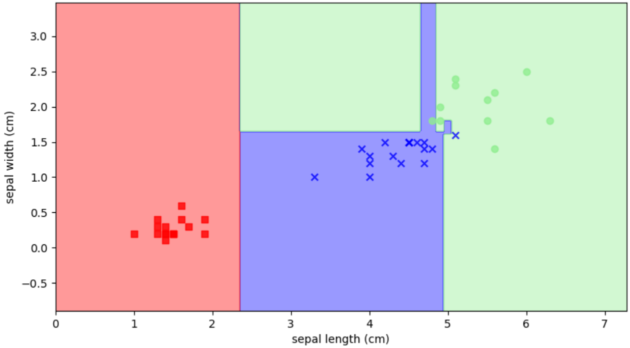
IRIS dataset - KNN

- K-Nearest Neighbor Classification
  - Find the majority among closest neighbors



Supervised Learning
IRIS dataset - Decision Tree

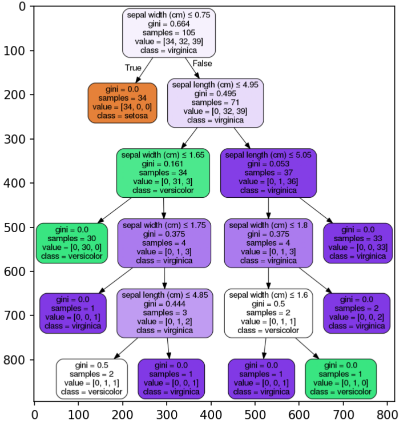
- Decision Tree Classification
  - Find best splitting rules based on feature values



Supervised Learning

IRIS dataset - Decision Tree

- Decision Tree Classification
  - Find best splitting rules based on feature values



Supervised Learning

Applications

- Very powerful when applied for Perception in Robotics!
  - Pattern Recognition for Robot Vision.
  - Probabilistic Models for Kalman and Particle Filters.
  - Fault detection.
  - Change detection.
- Learning from Demonstration for Control

Supervised Learning

Examples

- Autonomous Car Driving
  - A human drives a car and the driving behaviour is recorded.
  - ALVINN: Autonomous Land Vehicle In a Neural Network (Dean Pomerleau, 1988)
- Robotic Soccer
  - Four Legged League, Sony Aibo Platform.
  - Based on Accelerometer data, the robot was able to determine the surface it was moving on. *(Vail-2004)*
  - Also learned when it was moving freely or stuck or even entangled in other robots. *(Vail-2004)*
- Robotic Arm Control
  - Learn a probabilistic model for control over time.

Supervised Learning
Examples

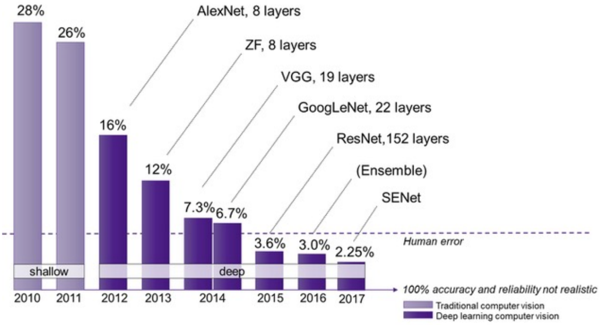


Figure: ImageNet Visual Recognition Challenge Results.

Outline

Unsupervised Learning

- Learns directly from data, without any labeled dataset;
- Finds patterns/structure in data;
- Examples:
  - Clustering;
  - Anomaly detection;
  - Autoencoders, Self-organizing maps.

Clustering

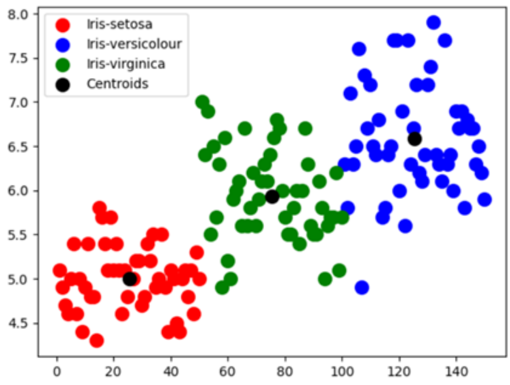


Figure: Clustering flowers.

Outline

Evolutionary Learning

- Unsupervised/Semi-supervised Learning Paradigm.
- A **policy may be encoded** in **strings** (Genetic Algorithms) or in **computer programs** (Genetic Programming).
- The learning process works on a **set of policies** (generation).
- The robot is provided a **fitness function**.
- Each individual on the generation is evaluated given the fitness function.
- **Genetic operators**: selection, crossover, mutation.
- Generations are **regenerated** trying to get better fitness values

Evolutionary Learning
Examples

- Robot Motion
  - Applied in-house to learn biped walking gaits (Picado-2009).
  - Applied in-house to learn kick behavior (Abdolmaleki-2016).
  - Applied to learn the model of the robot (Lipson-2008).
- Robot Hardware
  - Applied to evolve the shape of the robot, to obtain previously unknown robot shapes (Lipson-2006).

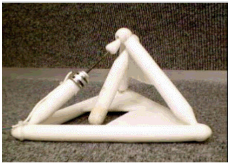


Figure: One of Hod Lipson morphologically evolved robots.

Outline

Reinforcement Learning

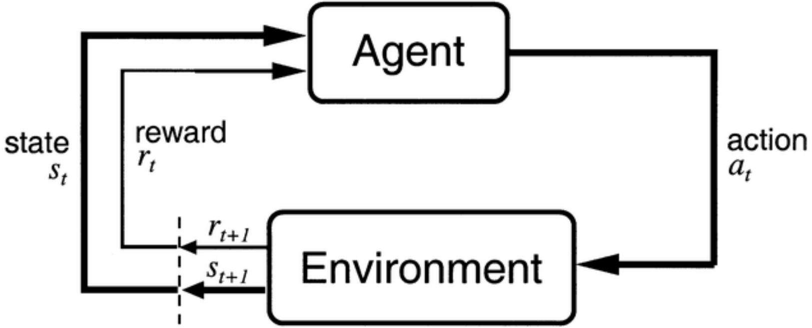


Figure: A Reinforcement Learning System.

Reinforcement Learning

- Unsupervised/Semi-supervised Learning Paradigm.
- Modeled as a Markov Decision Process:
  1. a set of states S;
  2. a set of actions *A(s)*;
  3. a state transition model: $P(s, a, s') \rightarrow [0, 1]$
  4. a reward function : $R(s, a, s') \rightarrow r_t \in \mathbb{R}$
- The goal?
- To determine a policy that maximizes the return (total reward), $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
- How?
- By calculating a *Value Function*

Reinforcement Learning

- OpenAI gym (Markov Decision Process API)

```
import gym

# load the environment
env = gym.make('Example')

# perform N episodes
while True:

    # prepare the environment for the next episode
    obs = env.reset()

    # flag for episode completion
    done = False

    # run the episode
    while not done:

        # choose how to act based on the current state
        # usually the output of a neural network
        action = env.action_space.sample()

        # advance the simulation by one timestep
        # by interacting with the world
        obs, reward, done, info = env.step(action)

# cleanup
env.close()
```

Reinforcement Learning

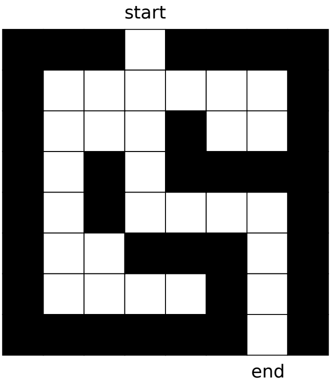


Figure: A maze environment.

Reinforcement Learning

- A Value Function estimates the expected return for *all* states
- $V^{\pi}(s_t) = \mathbb{E}[R_t]$

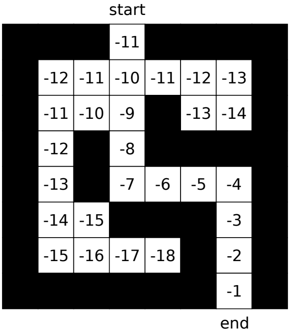


Figure: The optimal value function.

Reinforcement Learning

- A *Value Function* represents "how good" it is to be in a given state, $V^{\pi}(s_t) : s_t \rightarrow \mathbb{E}[R_t]$
- Bellman Equation:

  $V^{\pi}(s) = r(s, \pi(s), s') + \gamma V(s')), s' = f(s, \pi(s))$
- How can we learn a Value Function?

Reinforcement Learning

- A *Value Function* represents “how good” it is to be in a given state, $V^{\pi}(s_t) : s_t \rightarrow \mathbb{E}[R_t]$
- Bellman Equation:

  $V(s) = \sum_{s'} P(s, \pi(s), s')[r(s, \pi(s), s') + \gamma V(s')]$
- How can we learn a Value Function?

Reinforcement Learning

Algorithm 1 Policy Iteration (Policy Evaluation)

Require: $V(s)$ arbitrarily initialized, $\forall s \in S$

repeat
    repeat
        $\Delta \leftarrow 0$
        for all $s \in S$ **do**
            $v \leftarrow V(s)$
            $V(s) \leftarrow \sum_{s'} P(s,\pi(s),s')[r(s,\pi(s),s') + \gamma V(s')]$
            $\Delta \leftarrow max(\Delta, |v - V(s)|)$
        end for
    until $\Delta \leq \theta$ ▷ a small positive value
    ...
until StablePolicy == `True`

Reinforcement Learning

Algorithm 2 Policy Iteration (Policy Improvement)

Require: $V(s)$ arbitrarily initialized, $\forall s \in S$

repeat
    ...
    StablePolicy $\leftarrow$ `True`
    for all $s \in S$ **do**
        $b \leftarrow \pi(s)$
        $\pi(s) \leftarrow \arg\max_a \sum_{s'} P(s,a,s')[r(s,a,s') + \gamma V(s')]$
        if $b \neq \pi(s)$ **then**
            StablePolicy $\leftarrow$ `False`
        end if
    end for
until StablePolicy == `True`

Reinforcement Learning

Algorithm 3 Value Iteration

Require: $V(s)$ arbitrarily initialized, $\forall s \in S$

1: **repeat**
2: $\quad \Delta \leftarrow 0$
3: $\quad$ **for all** $s \in S$ **do**
4: $\quad\quad v \leftarrow V(s)$
5: $\quad\quad V(s) \leftarrow max_a \sum_{s'} P(s,a,s')[r(s,a,s') + \gamma V(s')]$
6: $\quad\quad \Delta \leftarrow max(\Delta, \|v - V(s)\|)$
7: $\quad$ **end for**
8: **until** $\Delta \leq \theta$ $\triangleright$ a small positive value

Reinforcement Learning

- That's nice, but what do we do with a *Value Function*?
- Extract a *policy!*
- A *policy* maps states to actions, $\pi : s \rightarrow a$
- $\pi(s) = \arg\max_a \sum_{s'} P(s, a, s')[r(s, a, s') + \gamma V(s')]$

Reinforcement Learning

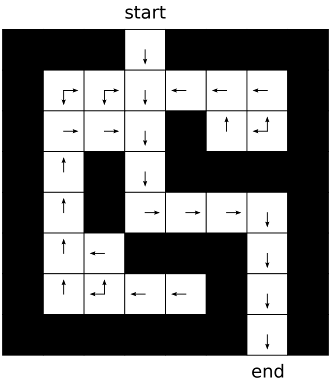


Figure: The optimal policy

Reinforcement Learning

1. Finite State Space
2. Finite Action Space
3. Curse of Dimensionality

Reinforcement Learning

1. Finite State Space
2. Finite Action Space
3. Curse of Dimensionality

Question?

What if we don't know the state transition model?

Example

- Where is the opponent dribbling the ball?
- To where in the goal is the opponent shooting the ball?

Reinforcement Learning

1. Finite State Space
2. Finite Action Space
3. Curse of Dimensionality

Question?

What if we don't know the state transition model?

Example

- Where is the opponent dribbling the ball?
- To where in the goal is the opponent shooting the ball?

Reinforcement Learning

- We use an *action* Value Function $Q^{\pi}(s_t, a) = \mathbb{E}[R_t | a_t = a]$
- We let the robot interact with the world and observe the collected rewards
- From that we build a *Value Function*

Reinforcement Learning

Algorithm 4 The Q-Learning algorithm

Require: $Q(s,a)$ initialized arbitrarily $\forall s \in S, \forall a \in A(s)$

loop

    Initialize $s = s_0$

    repeat

        $a \leftarrow \pi(s)$

        take action a; observe reward, r, and successor state s'

        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_b Q(s',b) - Q(s,a)]$

        $s \leftarrow s'$

    until s is terminal

end loop

Reinforcement Learning

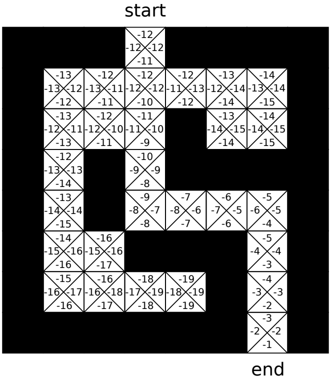


Figure: The optimal Q-function.

- $\pi(s) = \arg\max_a Q(s, a)$

Reinforcement Learning

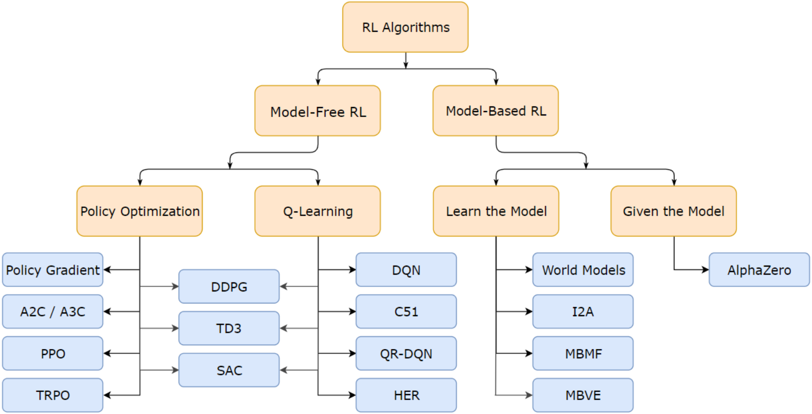


Figure: Reinforcement Learning Taxonomy.

Summary

- Machine Learning is a valid software development tool for programming robotic agents.
- Many paradigms, many challenges, many solutions, even more problems. . .
- The future of Robotics?

References

- Pattern Recognition and Machine Learning, Christopher Bishop, Springer, 2006
- Machine Learning, Tom Mitchell, McGraw Hill, 1997
- Reinforcement Learning: An Introduction, Richard Sutton, Andrew Barto, MIT Press, 2018
- OpenAI, scikit-learn, TensorFlow, PyTorch, stable baselines, Keras, RapidMiner, clsquare, OpenCV, Shark, . . .