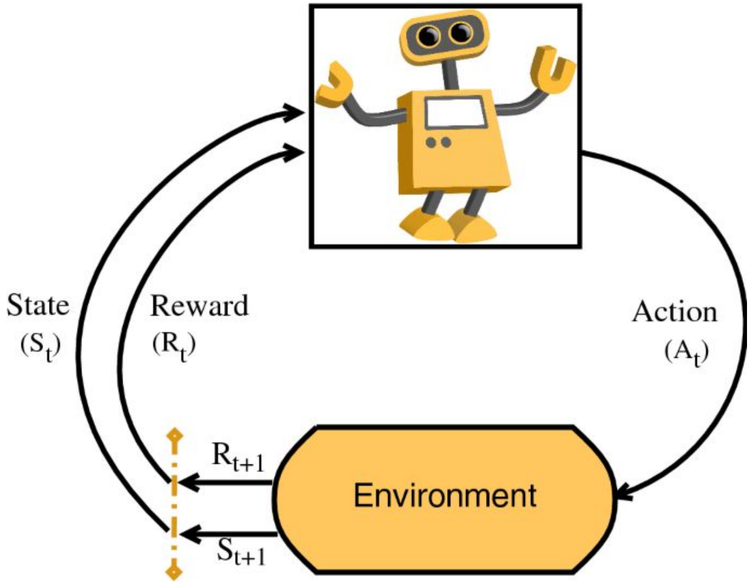
Reinforcement Learning

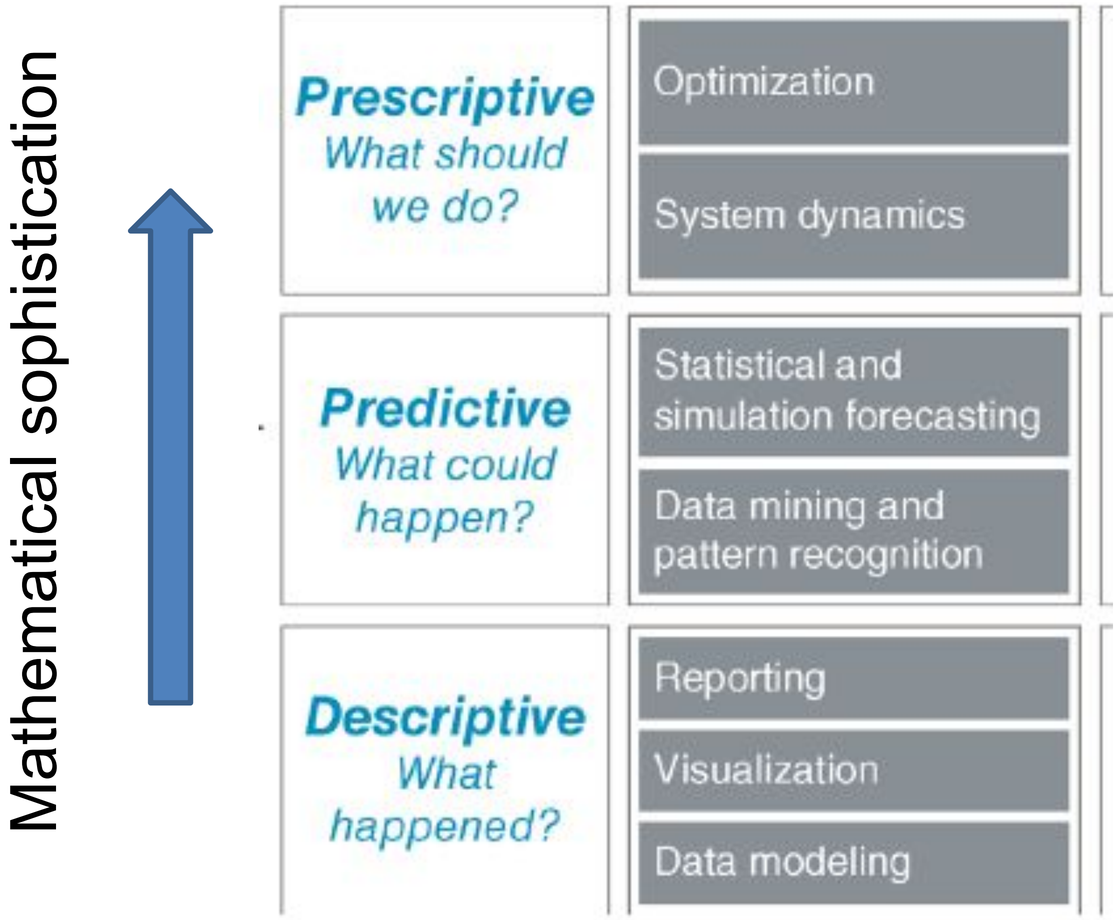


DA344: Jul-Dec 2023

Office: MB 113, AV 106

vidyapradananda@gm.rkmvu.ac.in

Analytics Framework



Mathematical sophistication

Prescriptive *What should we do?*	Optimization System dynamics
Predictive *What could happen?*	Statistical and simulation forecasting Data mining and pattern recognition
Descriptive *What happened?*	Reporting Visualization Data modeling

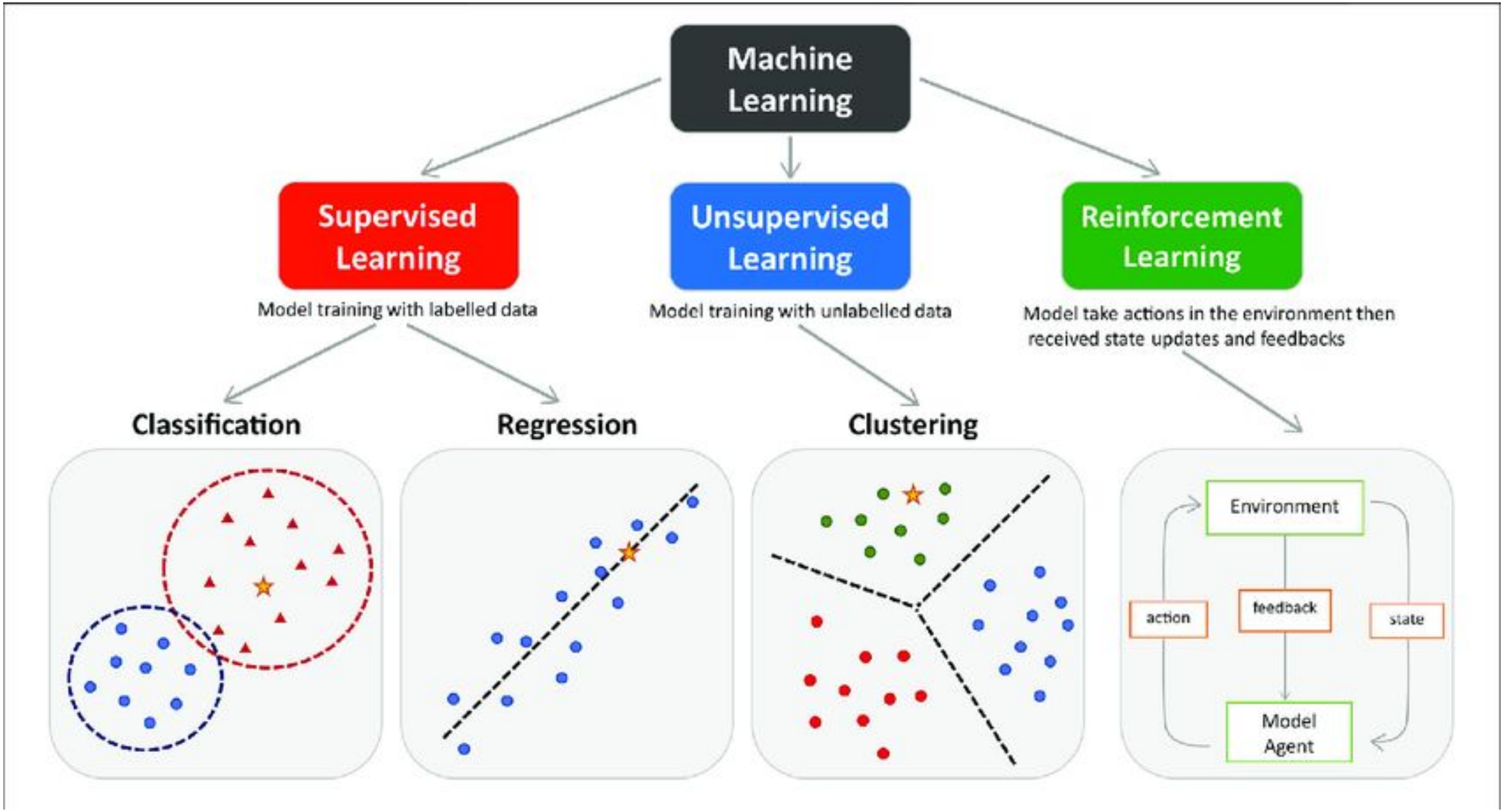
Make actionable recommendation about what should be done

forecast the likely future state

reveal and summarize facts

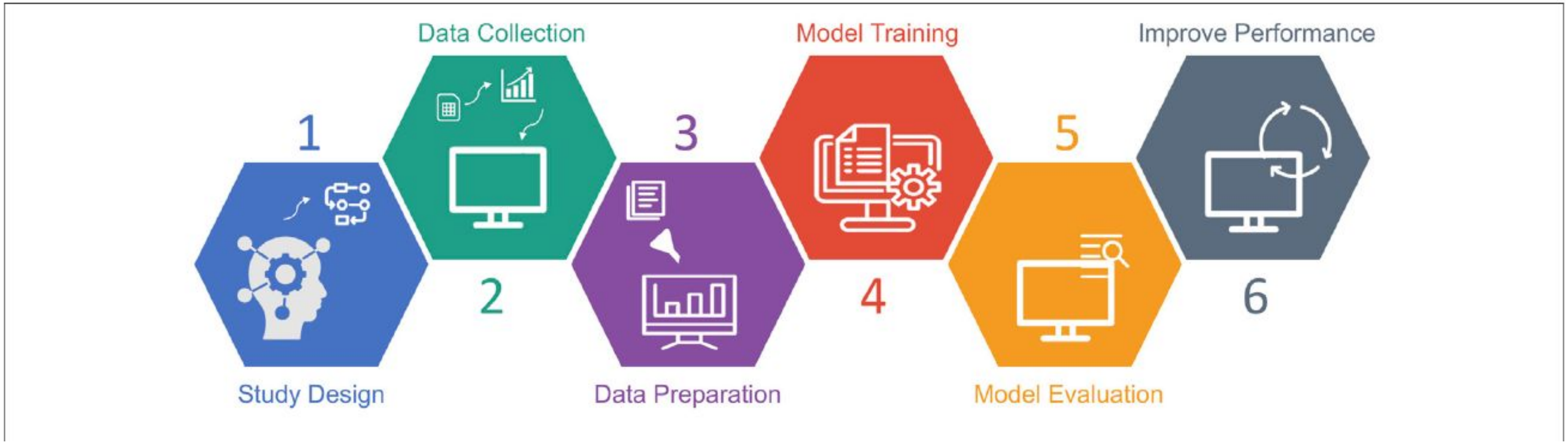
Lustig I, Dietrich B, Johnson C, Dziekan C (2010) The analytics journey, *Analytics Magazine*, 11–18.

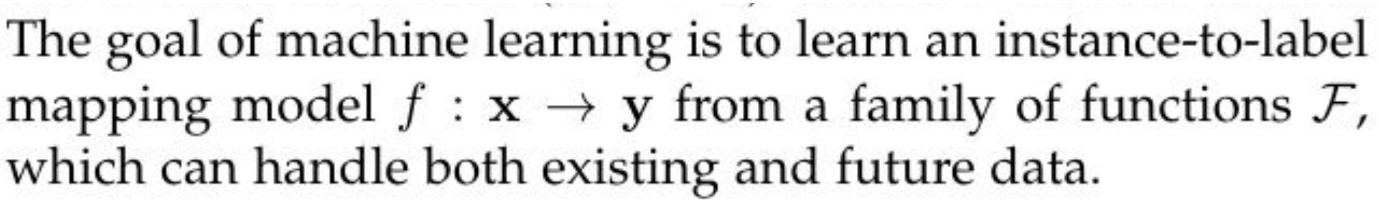
Branches of Machine Learning



Peng J, Jury EC, Dönnes P and Ciurtin C (2021) Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges. *Front. Pharmacol*. 12:720694.

Review of Machine Learning





The goal of machine learning is to learn an instance-to-label mapping model $f : \mathbf{x} \rightarrow \mathbf{y}$ from a family of functions $\mathcal{F}$, which can handle both existing and future data.

ML Setup

- **Feature vector**: a_j, $j=1,2,\ldots,m$
- Outcome/observation: y_j for each a_j.

- The outcomes could be
  - ✔ y_j real : **regression**
  - ✔ y_j is a label indicating a_j lies in one of N (N>=2) classes: **classification**
  - ✔ Multiple labels: classify according to multiple criteria
  - ✔ No labels (y_j is null) : Partition a_j into few clusters: **clustering**

ML Setup

- Find a function $\Phi(x_j)$ that approximately maps x_j to y_j for each j : $\Phi(x_j) \approx y_j$ for *j = 1; 2; : : : ;m* - MODEL SELECTION
- We define $\Phi(.)$ in terms of some parameter vector **w**
- Identification of $\Phi(.)$ becomes a data-fitting problem: Find the best **w** so that **x** (data) fits **y** (label)
- Objective function in this problem is built up of *m* terms that capture mismatch between predictions and observations for each (x_j; y_j). LOSS FUNCTION
- The process of finding $\Phi(.)$ is called learning or training. MINIMIZING LOSS
- Prediction: Given new data vectors x_k predict output $y_k \rightarrow \Phi(x_k)$

ML: Error analysis

Total Error

= Approximation error + estimation error + optimization error

$$\mathcal{E}_{app} + \mathcal{E}_{est} + \mathcal{E}_{opt},$$

approximation error measures how closely the chosen model/ function *Φ* can approximate the optimal solution

estimation error evaluates the effect of minimizing the empirical risk instead of the expected risk;

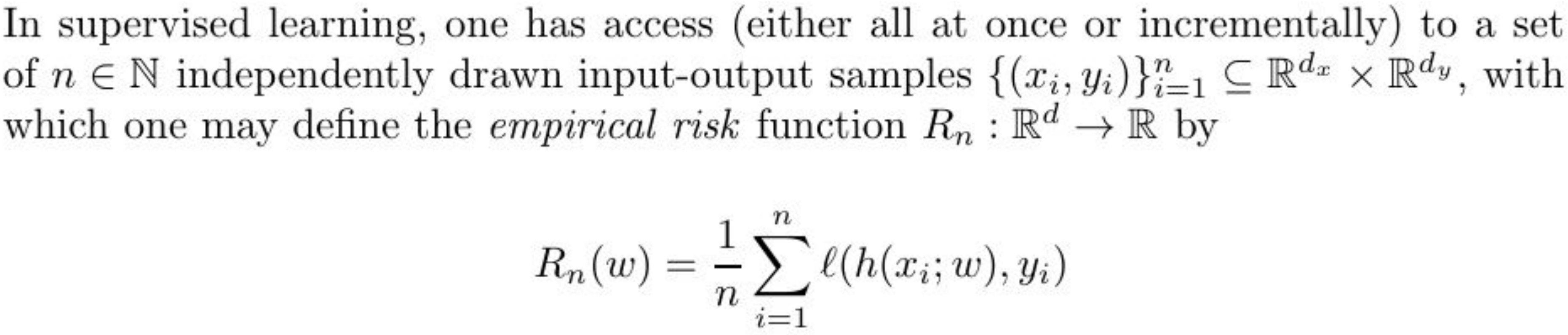
optimization error, measures the impact of the optimization algorithm on the generalization performance.

ML: Error analysis

Approximation error

how closely the function *Φ* can approximate the optimal solution beyond *Φ*

Estimation error



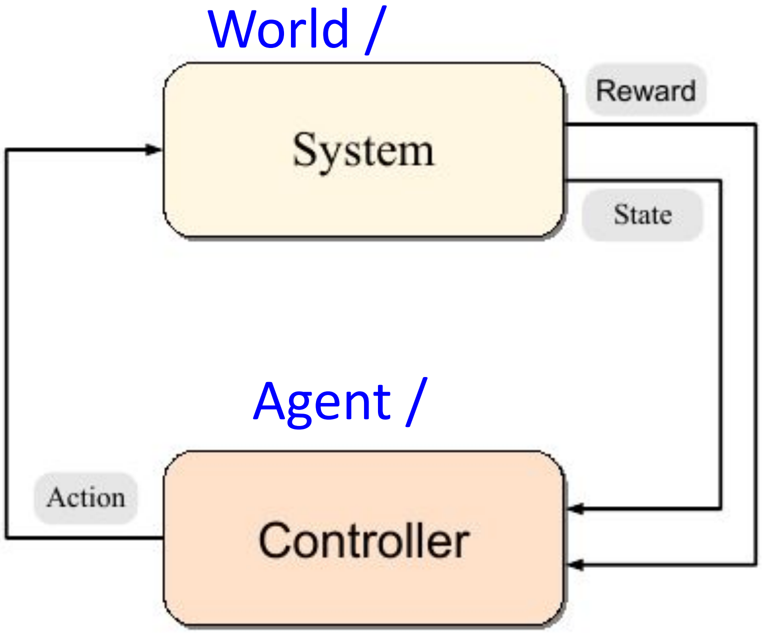
In supervised learning, one has access (either all at once or incrementally) to a set of $n \in \mathbb{N}$ independently drawn input-output samples $\{(x_i, y_i)\}_{i=1}^{n} \subseteq \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, with which one may define the *empirical risk* function $R_n : \mathbb{R}^d \to \mathbb{R}$ by

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i)$$

Optimization error

is affiliated with optimization algorithms used to minimize loss (e.g., mini batch GD vs SGD. Algorithms play a crucial role in computational efficiency (reduction in the optimization error per computation unit).

RL Scenario : Sequential Decision



- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Some examples

Learning Dexterity

https://youtu.be/jwSbzNHGflM

Learning to walk

https://www.youtube.com/shorts/v9l2iDVCSpM?feature=share

Learning to save

https://www.youtube.com/shorts/HD2_PHpxv9g?feature=share

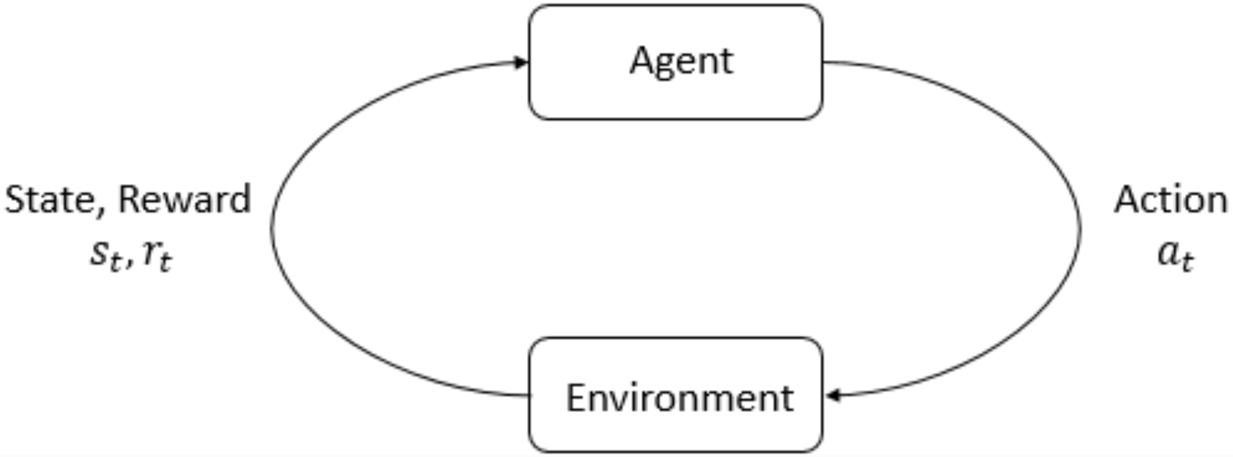
Automobile racing : Gran Turismo

https://www.cs.utexas.edu/~pstone/media/Stone_UofM_041522.mp4

Human -to- Assistant conversation

dynamic planning for human-to-assistant conversations,

RL Scenario explained



The **environment** is the world that the **agent** lives in and interacts with.

At every step of interaction, the agent sees a (possibly partial) observation of the **state** of the world, and then decides on an **action** to take.

The environment changes when the agent acts on it, but may also change on its own.

The agent also perceives a **reward** signal from the environment, a number that tells it how good or bad the current world state is.

The goal of the agent is to maximize its cumulative reward, called **return**.

RL Scenario: Issues

Optimization -

- Goal is to find an optimal way to make decisions yielding best outcomes or at least very good outcomes. Notion of utility of decisions

Delayed consequences

- Decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications

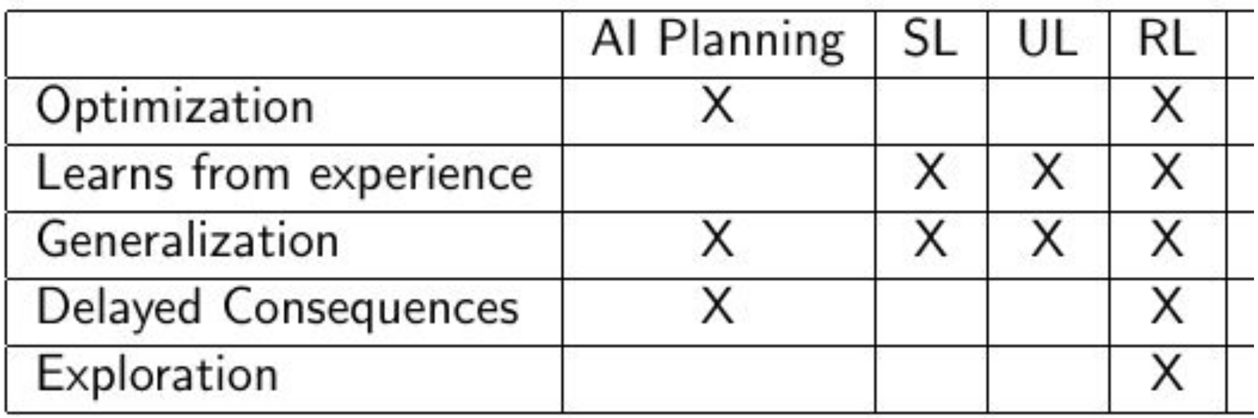
Exploration

- Learning about the world by making decisions, Decisions impact what we learn about

Generalization

- mapping from past experience to action (policy)

RL vs Other AI and Machine Learning



	AI Planning	SL	UL	RL
Optimization	X			X
Learns from experience		X	X	X
Generalization	X	X	X	X
Delayed Consequences	X			X
Exploration				X

SL = Supervised learning; UL = Unsupervised learning; RL =Reinforcement Learning

Source: https://web.stanford.edu/class/cs234/modules.html

Model-Free vs Model-Based RL

- Agent has access to (or learns) a model of the environment. By a model of the environment, we mean a function which predicts state transitions and rewards.
- it allows the agent to plan by thinking ahead, seeing what would happen for a range of possible choices, and explicitly deciding between its options
- ground-truth model of the environment is usually not available to the agent.it has to learn the model purely from experience
- What to learn?
  - policies, either stochastic or deterministic,
  - action-value functions (Q-functions),
  - value functions, and/or environment models.

Go, AlphaGo & AlphaGo Zero

- Go originated in China over 3,000 years ago. Winning this board game requires multiple layers of strategic thinking.

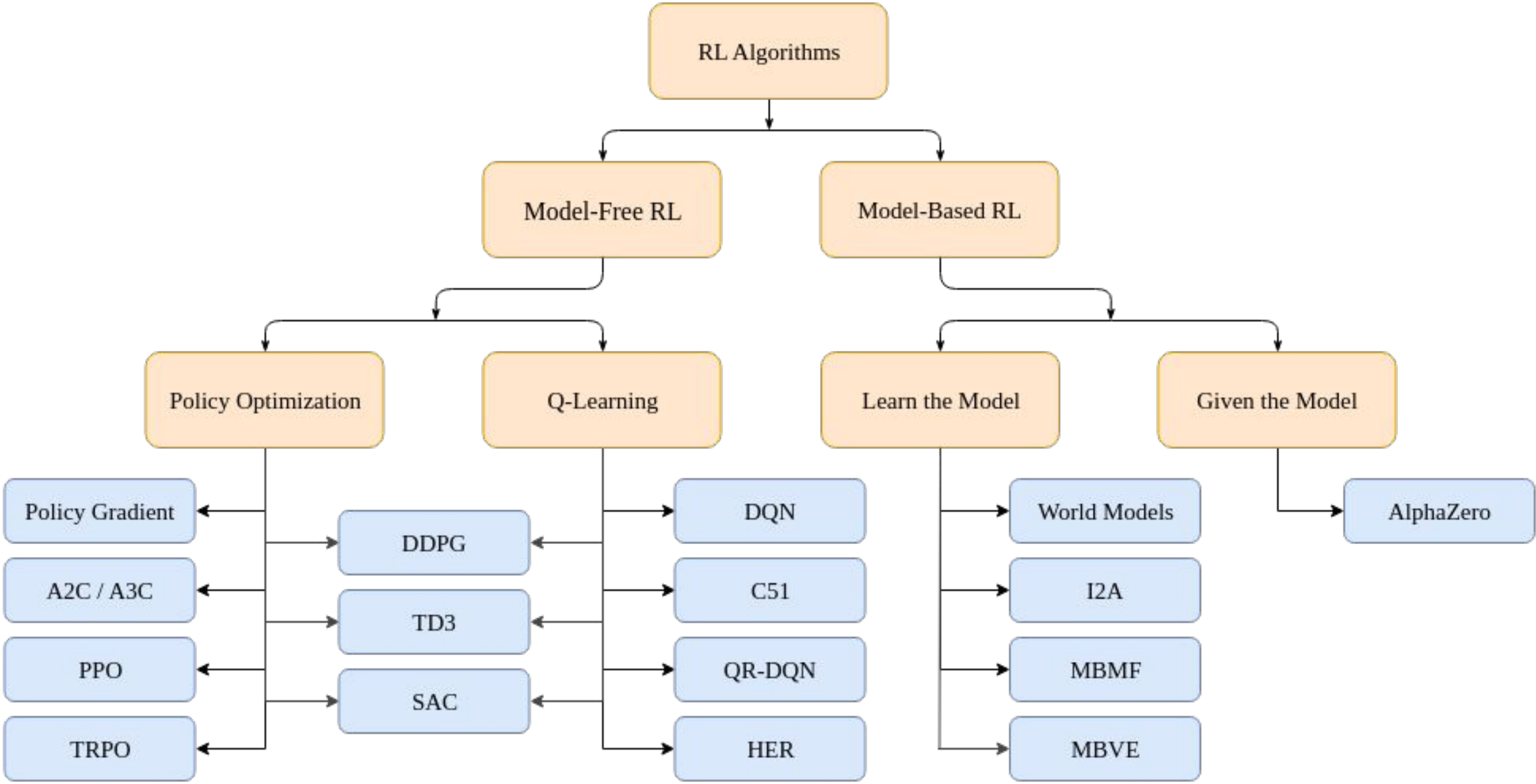
- Go is profoundly complex. There are 10^{170} possible board configurations - See Go Rules

- DeepMind's AlphaGo is a computer program that combines advanced search tree with deep neural networks. The "policy network", selects the next move to play (ACTION). The other neural network, the "value network", predicts the winner (REWARD) - See AlphaGo - The Movie

- AlphaGo Zero learnt by playing against itself, starting from completely random play. See paper

See **Deepmind AlphaZero**

Taxonomy of algorithms in modern RL



https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Stock option Problem

Suppose there is a stock whose price changes randomly every day. Let S_k be the price of the stock on the k-th day *(k >* 1) and suppose that on the k+1-th day the price is

$$S_{k+1} = S_k + X_{k+1} = S_0 + \sum_{i=1}^{k+1} X_i,$$

Where, $X_{1,}\ X_2$ … *are* independent and identically distributed random variables with distribution *F.* This is called the *random-walk* model for stock prices.

Suppose you have an option to buy one share of the stock at a fixed price ***c***, and you have ***N* days** in which to exercise the option.

If you exercise it at a time when the stock's price is ***s***, then your profit is ***s* − *c.***

What strategy maximizes the expected profit?

Formulating as DP

n stages: *i =1 to n*

State: stock price *s*

Decision: exercise the option at price *s* or do nothing

Transition: (*s*-*c*) if option is exercised at price *s*

-*c* if option is not exercised at price *s*

Cost/reward : expected profit : *s*-*c* or 0

Boundary condition: $V_0(x) = \max(s - c, 0)$ (in the end if price is *s*)

Value function: maximal expected profit when the stock's current price is *s* and the option has *i* additional days to run

$$V_i(x) = \max\left[s - c, \int V_{i-1}(s + x) dF(x)\right], i \geq 1$$

Formulating as DP

Optimal policy: No closed form ! Can be obtained iteratively.

Property of the optimal policy:

If there are n-days to go and the present price is s, and if the stock price is monotonically decreasing with time, i.e., there are increasing numbers $s_1 < s_2 < \ldots < s_n$, then one should exercise the option if and only if $s \geq s_n$

Intuition: If the current price is s and n days remain, then it is optimal to exercise the option if $V_n(s) < s - c$.

Investment Problem

- Suppose we have *D* units available for investment.

- During each of *N* time periods an opportunity to invest will occur with probability *p* independent of the past.

- If the opportunity occurs, the investor must decide how much of his remaining wealth to invest.

- If he invests *y,* then a return *R(y)* is earned at the end of the *N* time periods.

- Assuming that both the amount invested and the return become unavailable for future investment, the problem is to decide how much to invest at each opportunity to maximize the expected sum of investment returns

Formulating as DP

N stages: $i = 1 \text{ to } N$

State: Amount available for investment *A*

Decision: Amount to invest *y*

Transition: (*A-y*) if investment is done, else *A*

Reward : $R(y)$ [nondecreasing concave function with *R(0)* = 0]

Boundary condition: $V_0(A) = 0$ (in the end there is no reward)

Value function: Let V_i *(A)* denote the maximal expected additional profit attainable when there are *i* time periods to go, *A* amount is available for investment, and an opportunity is at hand

$$V_i(A) = \max_{0 \leq y \leq A} \left[R(y) + \overline{V}_{i-1}(A-y) \right], i > 0$$

where, $\overline{V}_{i-1}(A-y) = \sum_{j=0}^{i-1} p(1-p)^j V_{i-1-j}(A-y)$, is the maximal expected additional sum of returns when: *A-y* units remain for investment; there are i -1 time periods to go ; and it is not yet known if an investment opportunity is available.

Formulating as DP

Let $y_n(A)$ be the optimal amount to invest when the available investment capital equals A, there are n time periods remaining, and an opportunity to invest is at hand.

Property of the optimal policy:

(i) $y_n(A)$ is nondecreasing function of A,
(ii) $y_n(A)$ is nonincreasing function of n.

- When the return from an investment $R(y)$ is a concave function of the amount invested y, then "the more one has, the more one should invest," and "the more time one has, the less one should invest."
- Also, as the number of opportunities increases stochastically the amount invested decreases
- If $R(y)$ is convex, it is optimal to invest everything when an opportunity presents itself

More on RL

Reinforcement Learning for Real Life Workshop @ NeurIPS 2022

SIMONS INSTITUTE
for the Theory of Computing

Theory of Reinforcement Learning

RL THEORY VIRTUAL SEMINARS