# Aprenentatge per reforçament

| | | | |
|---|---|---|---|
| Iloc: | Institut d'Ensenyaments a Distància de les Illes Balears | Imprès per: | Carlos Sanchez Recio |
| Curs: | Sistemes d'aprenentatge automàtic | Data: | dimarts, 22 d'abril 2025, 21:28 |
| Llibre: | Aprenentatge per reforçament | | |

# Taula de continguts

# 1. Introducció

L'**aprenentatge per reforçament** s'ha convertit en una àrea prometedora dins de l'aprenentatge automàtic per abordar problemes seqüencials de presa de decisions que normalment es troben sota incertesa. Alguns exemples d'això inclouen la gestió d'inventaris amb diversos nivells i diversos proveïdors amb terminis de lliurament sota incertesa de la demanda; problemes de control com operacions de fabricació autònoma o control del pla de producció; i problemes d'assignació de recursos en finances o operacions.

L'aprenentatge per reforçament és un paradigma d'aprenentatge que aprèn a optimitzar les decisions seqüencials, que són decisions que es prenen de manera recurrent a través de passos de temps, per exemple, les decisions diàries de reposició d'estocs que es prenen en el control d'inventari. A un alt nivell, l'aprenentatge de reforç imita com aprenem nosaltres, com a humans. Els humans tenim la capacitat d'aprendre estratègies que ens ajuden a dominar tasques complexes com la natació, la gimnàstica o fer una prova. L'aprenentatge de reforçament cerca, en general, la inspiració d'aquestes habilitats humanes per aprendre a actuar. Però més específicament als casos d'ús pràctic de l'aprenentatge per reforçament, cerca adquirir la millor estratègia per prendre decisions seqüencials repetides al llarg del temps en un sistema dinàmic sota incertesa. Ho fa interactuant amb un simulador del sistema dinàmic estocàstic d'interès, també anomenat **entorn**, per aprendre aquestes estratègies guanyadores. Una estratègia per prendre decisions seqüencials repetides al llarg del temps en un sistema dinàmic també s'anomena **política**. L'aprenentatge per reforçament intenta aprendre la política guanyadora, és a dir, una recepta guanyadora de com prendre **accions** en diferents **estats** d'un sistema dinàmic.

## 1.1. Estats, accions i recompenses

L'aprenentatge per reforçament funciona en un marc matemàtic que consta dels elements següents:

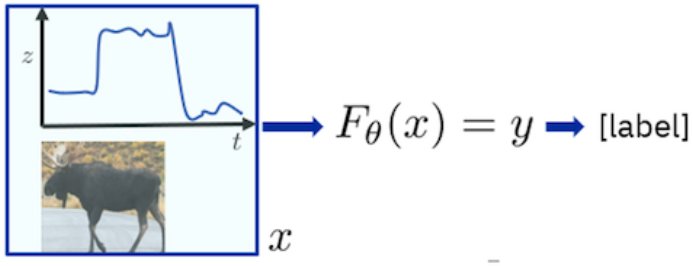
- Un **espai d'estats** (o espai d'observació): tota la informació disponible i les característiques del problema que són útils per prendre una decisió. Això inclou variables totalment conegudes o mesurades (per exemple, els nivells actuals d'estoc disponible en el control d'inventari), així com variables no mesurades de les quals només podeu creure o estimar (per exemple, previsió de la demanda per al dia o la setmana futurs). ).
- Un **espai d'accions**: decisions que pots prendre en cada estat del sistema.
- Un **senyal de recompensa**: un senyal escalar que proporciona la retroalimentació necessària sobre el rendiment i, per tant, l'oportunitat d'aprendre quines accions són beneficioses en un estat determinat. L'aprenentatge és tant local en la seva naturalesa per aprendre guanys immediats com guanys a llarg termini perquè les accions que es fan en qualsevol estat condueixen a estats futurs on es pren una altra acció, etc. El senyal de recompensa acumulada amb descompte és l'objectiu d'optimització per a l'aprenentatge de reforç, de manera que se centra en una estratègia a llarg termini que produeixi la millor recompensa acumulada.

La majoria dels problemes d'optimització dinàmics, així com alguns problemes d'optimització discrets deterministes (combinatoris) es poden expressar de manera natural en un marc **estat-acció-recompensa**. Un sistema dinàmic experimenta transicions (incertes) a l'espai d'estats quan es fan accions en qualsevol estat per recollir una recompensa local i impulsar el sistema endavant en el temps. Per exemple, un model de procés de decisió de Markov (MDP) formalitza la presa de decisions seqüencial en sistemes dinàmics sota transicions i recompenses incertes, i pren la forma d'un model estat-acció-recompensa.

L'aprenentatge mitjançant el reforç en sistemes dinàmics sota transicions incertes i recompenses incertes combina dues idees que es reforcen mútuament: explorar nous estats i noves combinacions estat-acció, i utilitzar l'experiència resultant per millorar la presa de decisions. Aquestes són les dues idees fonamentals en l'aprenentatge de reforç, és a dir, l'**exploració** i l'**explotació**. Amb el temps suficient (és a dir, la recopilació suficient d'experiència), l'aprenentatge de reforç pot conduir a una estratègia guanyadora (o una política) que podeu utilitzar per a la presa de decisions a llarg termini en problemes repetits de presa de decisions.

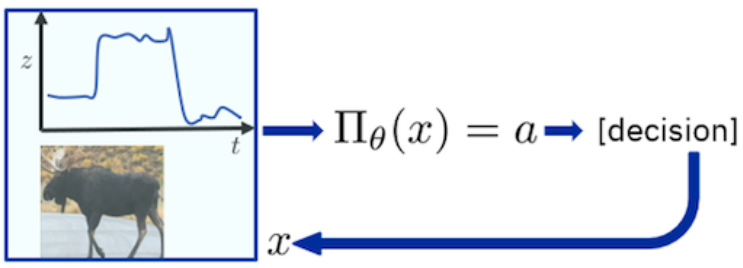
## 2. Reforçament vs. supervisió

Per entendre l'aprenentatge per reforçament, és útil contrastar-lo amb l'aprenentatge automàtic supervisat clàssic . En l'aprenentatge automàtic supervisat, l'objectiu és predir allò que no se sap. Funciona millor sota els supòsits estadístics de dades d'entrada independents i distribuïdes de manera idèntica (dades d'entrada i.i.d). En termes simples, això significa que se suposa que cada registre de dades d'entrada és independent de tots els altres registres, i que cada registre de dades d'entrada és una realització d'algun model de distribució de dades d'entrada únic i comú. Quan s'utilitza com a model de presa de decisions, el model de predicció supervisada és una decisió o predicció d'un sol cop que se centra exclusivament en un registre determinat de dades d'entrada i es desacobla de tots els altres registres de dades d'entrada. L'aprenentatge del model de predicció es produeix mitjançant la supervisió utilitzant la maximització de la precisió de la predicció com a objectiu d'aprenentatge. Tenim accés a la predicció de la veritat de referència (*ground truth*) corresponent a cada registre d'entrada de dades que s'utilitza per a l'entrenament. La figura següent recull aquest paradigma del que es pot anomenar "aprendre a predir".




En canvi, l'aprenentatge de reforç és un marc per a la presa de decisions basada en l'aprenentatge, on ja no tenim una taula de dades i.i.d amb etiquetes de veritat bàsica. En canvi, tenim trajectòries de tuples en forma de combinacions "estat actual-acció-estat següent-recompensa" que són interdependents en sèrie, és a dir, les dades ja no són un conjunt de dades tabulars estàtiques a diferència de l'aprenentatge automàtic supervisat. L'objectiu de l'aprenentatge és produir una **política**, és a dir, un mapeig o una estratègia que calculi la següent millor acció a prendre, entenent que qualsevol acció que es fa influeix en les entrades futures en el mapeig de polítiques per calcular la millor acció següent i així successivament. Això fa que l'aprenentatge ja no se centra exclusivament en l'estat actual, sinó també en les conseqüències a llarg termini sobre els estats futurs que es produeixen com a conseqüència de l'acció actual en qüestió.

Aquest aspecte és diferent de l'enfocament d'un sol tir (*single-shot*) d'utilitzar un model predictiu que només utilitza l'entrada actual. Finalment, a diferència de l'aprenentatge automàtic supervisat, no hi ha cap veritat bàsica ni supervisió disponible. En canvi, l'aprenentatge es realitza a partir de l'experiència en forma de seqüència dinàmica acoblada en sèrie de tuples "estat-acció-estat següent-recompensa", és a dir, experiència en forma de trajectòries dinàmiques controlades juntament amb recompensa a l'espai d'estats i destil·lant aquesta experiència per aprendre a actuar de manera òptima. La figura següent il·lustra aquest paradigma del que es pot anomenar "aprendre a actuar".



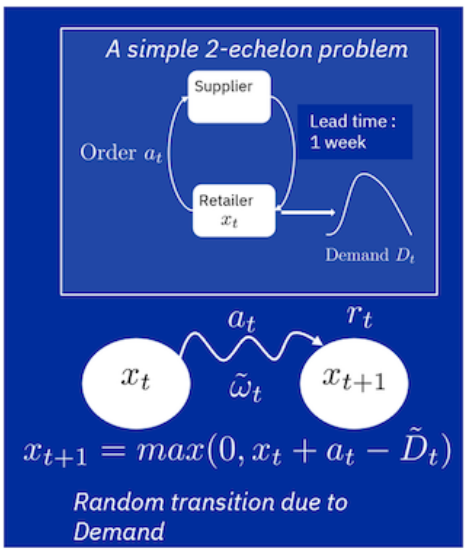
# 3. Modelització

Com s'ha descrit anteriorment, l'aprenentatge de reforç permet una plantilla de modelització molt general per a problemes seqüencials. Aquests són problemes que impliquen un sistema (per exemple, un sistema dinàmic físic) que passa d'un estat a un altre en prendre accions sota l'aleatorietat del sistema. Com a desenvolupador de solucions, podeu modelitzar el vostre problema triant-hi els elements següents:

- Espai d'estats: tota la informació disponible i les característiques del problema que són útils per prendre una decisió
- Espai d'acció: decisions que podeu prendre
- Recompensa: mesura de rendiment clau de l'interès que us interessa, per exemple, els ingressos
- Restriccions: podeu augmentar la recompensa amb una penalització negativa per la violació de cada restricció d'interès
- Incertesa: aleatorietat del sistema fora del vostre control

Per prendre un exemple instructiu, considereu el control d'inventaris on un minorista ha de prendre decisions setmanals de reposició d'un proveïdor amb incertesa en el temps de lliurament de la demanda i del subministrament. L'espai d'estat podria incloure cadascun dels següents: estoc actual al final de cada setmana, demanda observada durant les últimes setmanes per intentar conèixer els patrons de demanda, accions de reposició anteriors realitzades durant les últimes setmanes per tenir en compte l'aleatorietat del temps de lliurament del proveïdor i previsió de demanda estimada per a la setmana vinent. Això cobreix tant variables directament mesurables com conegudes, així com variables estimades de creences d'interès (com la previsió).

L'espai d'acció podria incloure l'ordre de reposició col·locada al final de cada setmana per rebre amb un temps de lliurament. Finalment, la incertesa del sistema es refereix a la incertesa de la demanda i la incertesa del temps de lliurament del proveïdor. Podríeu tenir restriccions que penalitzin la superació o la baixa dels nivells d'estoc, que evolucionen sota la incertesa en aquest exemple. Per simplificar la il·lustració, la figura següent mostra les transicions en una versió simplificada d'aquest problema de 2 escalons, on l'espai d'estat escollit (x) fa un seguiment del nivell d'inventari sota la incertesa de la demanda (ω o D), en prendre accions de reposició (a) i rebre una recompensa local (r), en el supòsit d'un termini de lliurament fix i conegut d'una setmana.

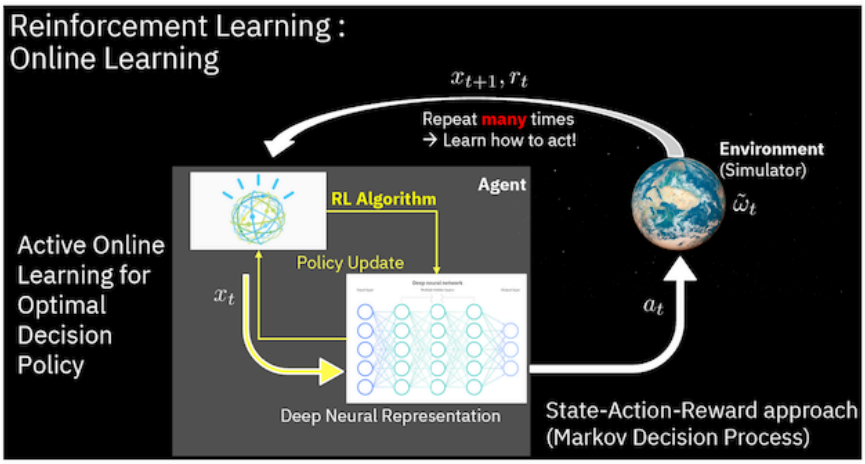


# 4. RL online i offline

Les tècniques d'aprenentatge de reforç es poden organitzar de manera àmplia i útil en termes de tècniques en línia versus tècniques fora de línia. En l'aprenentatge de reforç en línia, teniu accés a un sistema en directe o simulat que representa (o codifica mitjançant la simulació) la dinàmica del sistema, és a dir, com evoluciona el sistema d'estat a estat en prendre accions permeses i, a més, normalment sota incertesa. A més, també proporciona un senyal de recompensa en cada pas de temps. Aquest senyal és possiblement buit o nul i, de manera més general, possiblement és multivariant, és a dir, més d'una mesura de rendiment clau d'interès. Els algorismes d'aprenentatge de reforç en línia (o agents) aprenen interactuant amb un sistema en línia tan sensible. En canvi, l'aprenentatge de reforç fora de línia (o aprenentatge de reforç per lots) és un entorn complementari on no teniu accés a un sistema tan sensible i interactiu. En canvi, l'aprenentatge de reforç fora de línia només té accés a un registre històric d'accions passades, transicions d'estats anteriors i recompenses corresponents, i aprèn una política a partir d'aquestes dades històriques.

# 5. La interfície OpenAI Gym.Env

En l'aprenentatge de reforç en línia, un algorisme d'aprenentatge per reforçament (o **agent**) interactua amb un **simulador** del sistema. Adquireix experiència en forma de trajectòries tal com es descriu en els apartats anteriors combinant **exploració** i **explotació**. En l'aprenentatge per reforçament profund (*Deep Reinforcement Learning*, **DRL**), l'agent manté una representació de la política basada en la xarxa neuronal, que perfecciona i millora a través d'interaccions que ofereixen a l'agent una experiència d'aprenentatge valuosa. La figura següent recull l'aprenentatge interactiu i experiencial que es produeix en l'aprenentatge per reforçament en línia. La promesa de l'aprenentatge de reforç és que pot utilitzar aquest aprenentatge interactiu per descobrir noves heurístiques fins i tot en problemes antics ben estudiats com el control d'inventari en forma de política de decisió, per exemple, més enllà de les heurístiques clàssiques i analítiques que es poden provar fàcilment basant-se en resultats ja coneguts. Cal tenir en compte que els resultats clàssics fan diverses suposicions per a la tractabilitat analítica, mentre que l'aprenentatge en l'aprenentatge de reforç es pot fer relaxant tots aquests supòsits ideals en un simulador realista.




A la pràctica, el simulador del sistema s'implementa segons l'especificació de la interfície d'OpenAI Gym. L'especificació de la interfície OpenAI Gym està disponible a https://gymlibrary.dev. Aquesta és una especificació d'interfície de programari que permet la flexibilitat per capturar i modelar la majoria de sistemes dinàmics d'interès on és possible que vulgueu desenvolupar solucions basades en l'aprenentatge de reforç. Permet espais d'estat i acció flexibles en termes de variables finites o contínues amb límits, encara que siguin grans en la mida del nombre de variables d'estat i d'acció. L'espai d'estat s'anomena **observation_space**, que és un atribut de l'entorn a la interfície Gym.Env. L'espai d'acció s'anomena **action_space** i també és un atribut de l'entorn a la interfície Gym.Env. Podeu trobar més detalls sobre els atributs importants que formen un entorn, així com els seus possibles tipus al lloc https://gymlibrary.dev.

L'entorn proporcionat per l'usuari s'ha d'implementar a Python com una classe que deriva de la classe OpenAI Gym.Env. La documentació sobre com fer aquesta subclasse està disponible al lloc https://gymlibrary.dev. Cada entorn conté un mètode **__init__** que construeix i inicialitza l'entorn utilitzant els atributs dels espais d'acció i observació, juntament amb la inicialització de l'estat inicial del sistema. L'entorn també necessita una funció d'interfície anomenada **step** per capturar la dinàmica d'un sol pas a l'espai d'estat en resposta a qualsevol acció d'entrada en l'estat actual, així com la recompensa local d'un sol pas en fer aquesta acció. La funció de pas **step** és una funció important perquè indirectament (a través de l'experiència) ofereix l'oportunitat d'aprenentatge perquè l'agent d'aprenentatge de reforç que interactua adquireixi una representació adequada de la dinàmica i incerteses del simulador (és a dir, del sistema). Podeu trobar més detalls a la documentació de Gym.

Podeu utilitzar qualsevol model intern per incorporar incerteses i capturar la dinàmica del sistema al simulador, és a dir, aprofitar el poder expressiu del modelatge de simulació. Tingueu en compte que com millor sigui el simulador, més útil serà la política d'aprenentatge de reforç guanyador apresa quan la implementeu a la vida real. Més enllà de **__init__** i **step**, l'entorn ha de tenir una funció de reinici **reset**. Aquesta és una funció important que restableix l'estat del sistema a un estat inicial i retorna l'estat corresponent (o "observació") a la funció que l'ha invocat. Les dues últimes funcions de la interfície per renderitzar i tancar (**render** i **close**) són opcionals. La primera és útil per visualitzar mitjançant la representació de l'estat actual del sistema en entorns de joc, per tant opcional, i no serveix per a res més enllà d'alguna visualització. De la mateixa manera, **close** simplement neteja en

sortir de la simulació i, per tant, és opcional. Qualsevol agent que adquireixi experiència de manera interactiva per aprendre en un entorn només necessita un únic script de Python que pugui contenir diverses classes. La figura següent mostra un exemple d'un entorn compatible amb Gym amb les funcions principals de la interfície.

```python
import gym
from gym import error, spaces, utils
from gym.utils import seeding

class FooEnv(gym.Env):
    metadata = {'render.modes': ['human']}

    def __init__(self):
        # TODO
        ...
    def step(self, action):
        # TODO
        ...
    def reset(self):
        # TODO
        ...
    def render(self, mode='human'):
        # TODO
        ...
    def close(self):
        # TODO
        ...
```

Per a aquest sistema, heu de definir un entorn compatible amb Gym i embolicar-lo dins d'una funció, tal com es mostra a la figura següent. En aquest exemple, heu de definir un entorn compatible amb Gym anomenat BasicEnv. Aquesta classe segueix l'API Gym esmentada anteriorment. Podeu definir classes de Python addicionals al mateix fitxer, i aquestes classes es poden utilitzar com a classes de suport per a BasicEnv, la classe principal. També podeu importar paquets PyPI al mateix fitxer, i aquests paquets PyPI seran instal·lats pel nostre sistema. Perquè el codi funcioni, heu d'inserir dues línies al principi i al final del fitxer (per exemple, la línia 1 i la línia 22 de la figura següent). El primer és per def instantiate_env(*args, **kwargs): i el segon és return BasicEnv(). Amb aquestes dues línies, l'analitzador pot analitzar i carregar correctament l'entorn definit per l'usuari.

```python
def instantiate_env(*args, **kwargs):
    import gym

    class BasicEnv(gym.Env):
        def __init__(self):
            self.action_space = gym.spaces.Discrete(5)
            self.observation_space = gym.spaces.Discrete(2)

        def step(self, action):
            state = 1
            if action > 0:
                reward = 200
            else:
                reward = -100
            done = True
            return state, reward, done, {}

        def reset(self):
            state = 0
            return state

    return BasicEnv()
```

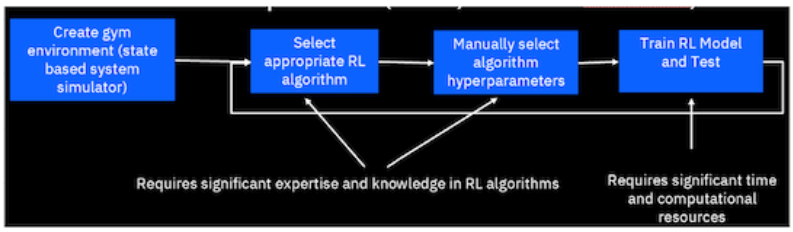
# 6. Reptes pràctics

Tot i que l'aprenentatge per reforçament és prometedor com a enfocament per incorporar la IA automatitzada per a la presa de decisions, a la pràctica està ple de reptes a l'hora d'aplicar-lo a problemes industrials realistes. En primer lloc, la literatura està dominada per problemes acadèmics de referència sobre els quals els investigadors estan impulsant activament l'estat de l'art en algorismes efectius. Això us deixa sense molts exemples de referència per buscar inspiració de modelització i confiança que és un enfocament viable per als problemes industrials d'interès.

En segon lloc, hi ha nombrosos algorismes com les tècniques basades en funcions de valor (com l'aprenentatge Q, DQN), tècniques basades en mapes de polítiques (com l'aprenentatge de polítiques i PPO) i les seves combinacions (com els mètodes crítics d'actor, A2C i SAC). Adquirir una bona comprensió del funcionament intern d'aquests algorismes requereix invertir en alguns esforços matemàtics i formació en temes com els processos de decisió de Markov, les cadenes de Markov, les tècniques de Monte-Carlo, la programació dinàmica i la recursivitat de Bellman, que pot ser que no sigui pràctic en el temps. molts científics de dades. A més, hi ha diversos marcs de codi obert i les seves respectives implementacions d'aquests diversos algorismes, com ara stable-baselines, RLLib, d3rlpy, Coach i Keras RL. Familiaritzar-se amb aquestes implementacions i com utilitzar-les de manera productiva també requereix una corba d'aprenentatge pronunciada que podria ser limitant a la pràctica per als desenvolupadors de solucions.

A continuació, cadascun d'aquests algorismes i les seves implementacions admet diversos hiperparàmetres ajustables i inclou alguns valors predeterminats, però se sap que l'aprenentatge de reforç és hipersensible en el seu rendiment a l'elecció inicial, així com a la reajustament dinàmic d'aquestes opcions a mesura que avança l'entrenament i l'aprenentatge. Aquesta triple maledicció de la dimensionalitat (nombre de marcs, nombre d'agents, nombre d'hiperparàmetres) requereix una experimentació a gran escala, distribuïda i intensiva en càlcul per investigar i cercar els millors algorismes i la política guanyadora que podria produir. Tot i que hi ha marcs distribuïts disponibles en codi obert com Ray i Ray Tune, també requereixen experiència especial per configurar un clúster informàtic i utilitzar-los de manera eficaç.

Finalment, l'aprenentatge de reforç profund utilitza representacions de xarxes neuronals profundes per capturar la funció de valor i el mapatge de polítiques, i això proporciona més opcions que heu de fer per a l'arquitectura neuronal. En general, l'amplitud i la profunditat de les àrees temàtiques d'expertesa i fluïdesa necessàries per fer que l'aprenentatge de reforç funcioni a la pràctica és un limitant seriós en la seva adopció per a problemes de la vida real. Això és encara més problemàtic a la pràctica perquè l'aprenentatge de reforç, malgrat la seva promesa, és un esforç arriscat sense garanties formals que tot l'esforç i la inversió que un equip pugui fer per adquirir aquestes habilitats donaran solucions útils. La figura següent mostra el flux de treball laboriós i que consumeix temps en l'aprenentatge de reforç avui per als professionals. Les API automatitzades d'IA per a la presa de decisions ofereixen una solució que automatitza aquest flux de treball de manera significativa i democratiza l'abast de l'aprenentatge de reforç per als científics de dades i els desenvolupadors de solucions amb una àmplia gamma d'experiència.



# 7. Deep Reinforcement Learning

L'aprenentatge de reforç profund (deep RL) és un subcamp de l'aprenentatge automàtic que combina l'aprenentatge de reforç (RL) i l'aprenentatge profund. RL considera el problema d'un agent computacional que aprengui a prendre decisions per assaig i error. Deep RL incorpora l'aprenentatge profund a la solució, permetent als agents prendre decisions a partir de dades d'entrada no estructurades sense enginyeria manual de l'espai d'estat. Els algorismes de RL profund són capaços de rebre entrades molt grans (per exemple, cada píxel representat a la pantalla en un videojoc) i decidir quines accions realitzar per optimitzar un objectiu (per exemple, maximitzar la puntuació del joc). L'aprenentatge de reforç profund s'ha utilitzat per a un conjunt divers d'aplicacions que inclouen, entre d'altres, la robòtica, els videojocs, el processament del llenguatge natural, la visió per ordinador, l'educació, el transport, les finances i la salut.

# 7.1. DL + RL = DRL

## Aprenentatge profund

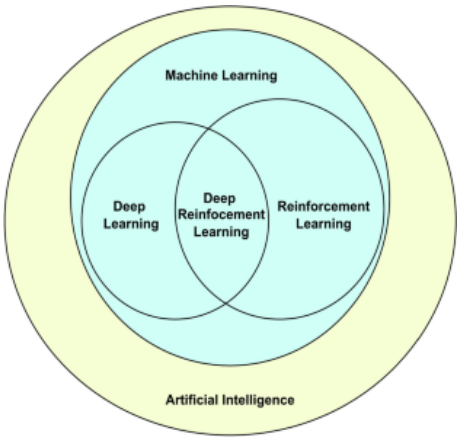
L'aprenentatge profund és una forma d'aprenentatge automàtic que utilitza una xarxa neuronal per transformar un conjunt d'entrades en un conjunt de sortides mitjançant una xarxa neuronal artificial. S'ha demostrat que els mètodes d'aprenentatge profund, que sovint utilitzen l'aprenentatge supervisat amb conjunts de dades etiquetats, resolen tasques que impliquen el maneig de dades d'entrada en brut complexes i de gran dimensió, com ara imatges, amb menys enginyeria manual de funcions que els mètodes anteriors, permetent un progrés significatiu en diversos camps, inclòs l'ordinador. visió i processament del llenguatge natural. En la darrera dècada, Deep RL ha aconseguit resultats notables en una sèrie de problemes, des de jocs per a un sol i multijugador com GO, Atari Games i Dota 2, fins a la robòtica

## Aprenentatge per reforçament

L'aprenentatge per reforç és un procés en el qual un agent aprèn a prendre decisions mitjançant assaig i error. Aquest problema sovint es modela matemàticament com un procés de decisió de Markov (MDP), on un agent en cada pas de temps es troba en un estat s, pren accions a, rep una recompensa escalar i passa al següent estat s' segons la dinàmica de l'entorn. L'agent intenta aprendre una política o un mapa des de les observacions fins a les accions, per tal de maximitzar els seus rendiments (suma esperada de recompenses). En l'aprenentatge de reforç (a diferència del control òptim) l'algoritme només té accés a la dinàmica mitjançant el mostreig.

## Aprenentatge per reforçament profund

En molts problemes pràctics de presa de decisions, els estats s de l'MDP són d'alta dimensió (per exemple, imatges d'una càmera o el flux de sensor en brut d'un robot) i no es poden resoldre amb algorismes RL tradicionals. Els algorismes d'aprenentatge de reforç profund incorporen aprenentatge profund per resoldre aquests MDP, sovint representant la política o altres funcions apreses com a xarxa neuronal i desenvolupant algorismes especialitzats que funcionen bé en aquest entorn.



## 7.2. Evolució

Juntament amb el creixent interès per les xarxes neuronals a partir de mitjans dels anys vuitanta, va créixer l'interès per l'aprenentatge per reforçament profund, on una xarxa neuronal s'utilitza en l'aprenentatge per reforçament per representar polítiques o funcions de valor. Com que en aquest sistema, tot el procés de presa de decisions, des dels sensors fins als motors d'un robot o agent, implica una única xarxa neuronal, de vegades també s'anomena aprenentatge de reforç d'extrem a extrem. Una de les primeres aplicacions reeixides d'aprenentatge per reforç amb xarxes neuronals va ser TD-Gammon, un programa informàtic desenvolupat l'any 1992 per jugar al backgammon. Es van utilitzar quatre entrades per al nombre de peces d'un color determinat en una ubicació determinada del tauler, amb un total de 198 senyals d'entrada. Amb el coneixement zero incorporat, la xarxa va aprendre a jugar el joc a un nivell intermedi mitjançant el joc propi i el mètode de diferència temporal.

A partir del 2012, l'anomenada revolució de l'aprenentatge profund va provocar un interès creixent en l'ús de xarxes neuronals profundes com a aproximadors de funcions en diversos dominis. Això va provocar un renovat interès en els investigadors que utilitzen xarxes neuronals profundes per aprendre la política, el valor i/o les funcions Q presents als algorismes d'aprenentatge de reforç existents.

A partir de l'any 2013, DeepMind va mostrar resultats d'aprenentatge impressionants amb RL profund per jugar als videojocs d'Atari. El jugador d'ordinador que va entrenar una xarxa neuronal utilitzant un algorisme RL profund, una versió profunda de l'aprenentatge Q que van anomenar xarxes Q profundes (DQN), amb la puntuació del joc com a recompensa. Van utilitzar una xarxa neuronal convolucional profunda per processar 4 fotogrames de píxels RGB (84x84) com a entrades. Els 49 jocs es van aprendre utilitzant la mateixa arquitectura de xarxa i amb coneixements previs mínims, superant els mètodes de la competència en gairebé tots els jocs i amb un nivell comparable o superior a un provador de jocs humà professional.

L'aprenentatge de reforç profund va assolir una altra fita el 2015 quan AlphaGo, un programa informàtic entrenat amb RL profund per jugar a Go, es va convertir en el primer programa informàtic Go que va vèncer un jugador professional de Go humà sense handicap en un tauler de 19×19 de mida completa. En un projecte posterior el 2017, AlphaZero va millorar el rendiment a Go alhora que va demostrar que podien utilitzar el mateix algorisme per aprendre a jugar als escacs i el shogi a un nivell competitiu o superior als programes informàtics existents per a aquests jocs, i va tornar a millorar el 2019 amb MuZero. Per separat, els investigadors de la Carnegie Mellon University van assolir una altra fita el 2019 desenvolupant Pluribus, un programa informàtic per jugar al pòquer que va ser el primer a vèncer als professionals en jocs multijugador de Texas Hold 'em sense límit. OpenAI Five, un programa per jugar cinc contra cinc Dota 2 va vèncer als anteriors campions del món en un partit de demostració el 2019.

L'aprenentatge de reforç profund també s'ha aplicat a molts dominis més enllà dels jocs. En robòtica, s'ha utilitzat per permetre que els robots facin tasques domèstiques senzilles i resoldre un cub de Rubik amb una mà de robot. Deep RL també ha trobat aplicacions de sostenibilitat, utilitzades per reduir el consum d'energia als centres de dades. Deep RL per a la conducció autònoma és una àrea activa de recerca a l'acadèmia i la indústria. Loon va explorar RL profund per navegar de manera autònoma pels seus globus d'altitud.

## 7.3. Algorismes

Existeixen diverses tècniques per formar polítiques per resoldre tasques amb algorismes d'aprenentatge de reforç profund, cadascuna amb els seus propis beneficis. Al nivell més alt, hi ha una distinció entre l'aprenentatge de reforç basat en models i l'aprenentatge lliure de models, que fa referència a si l'algoritme intenta aprendre un model avançat de la dinàmica de l'entorn.

En algorismes d'aprenentatge de reforç profund basats en models, s'estima un model avançat de la dinàmica de l'entorn, normalment mitjançant un aprenentatge supervisat mitjançant una xarxa neuronal. A continuació, s'obtenen accions utilitzant el control predictiu del model mitjançant el model après. Atès que la veritable dinàmica de l'entorn normalment divergirà de la dinàmica apresa, l'agent torna a planificar sovint quan realitza accions en l'entorn. Les accions seleccionades es poden optimitzar mitjançant mètodes de Montecarlo com el mètode d'entropia creuada, o una combinació d'aprenentatge de models amb mètodes lliures de models.

En els algorismes d'aprenentatge de reforç profund sense models, s'aprèn una política sense modelar explícitament la dinàmica avançada. Es pot optimitzar una política per maximitzar els rendiments estimant directament el gradient de la política, però pateix una gran variància, cosa que fa que no sigui pràctic per utilitzar-la amb l'aproximació de funcions en RL profund. Els algorismes posteriors s'han desenvolupat per a un aprenentatge més estable i s'han aplicat àmpliament. Una altra classe d'algoritmes d'aprenentatge de reforç profund sense models es basa en la programació dinàmica, inspirada en l'aprenentatge de diferències temporals i l'aprenentatge Q. En espais d'acció discrets, aquests algorismes solen aprenen una funció Q de xarxa neuronal Q(s,a) que estima els rendiments futurs fent acció a des de l'estat s. En espais continus, aquests algorismes sovint aprenen tant una estimació de valor com una política.

# 7.4. Recerca

L'aprenentatge de reforç profund és una àrea activa de recerca, amb diverses línies d'investigació.

## Exploració

Un agent de RL ha d'equilibrar el compromís d'exploració/explotació: el problema de decidir si perseguir accions que ja se sap que donen altes recompenses o explorar altres accions per descobrir recompenses més altes. Els agents RL solen recopilar dades amb algun tipus de política estocàstica, com ara una distribució de Boltzmann en espais d'acció discrets o una distribució gaussiana en espais d'acció contínua, induint un comportament bàsic d'exploració. La idea que hi ha darrere de l'exploració basada en la novetat o impulsada per la curiositat és donar a l'agent un motiu per explorar resultats desconeguts per trobar les millors solucions. Això es fa "modificant la funció de pèrdua (o fins i tot l'arquitectura de xarxa) afegint termes per incentivar l'exploració". També es pot ajudar a un agent en l'exploració mitjançant la utilització de demostracions de trajectòries reeixides o la creació de recompenses, donant a un agent recompenses intermèdies personalitzades per adaptar-se a la tasca que està intentant completar.

## Aprenentatge de reforç fora de la política

Una distinció important a RL és la diferència entre els algorismes integrats en la política que requereixen avaluar o millorar la política que recull dades i els algorismes fora de la política que poden aprendre una política a partir de les dades generades per una política arbitrària. En general, els mètodes basats en funcions de valor, com ara l'aprenentatge Q, són més adequats per a l'aprenentatge fora de la política i tenen una millor eficiència de la mostra: la quantitat de dades necessàries per aprendre una tasca es redueix perquè les dades es reutilitzen per a l'aprenentatge. A l'extrem, RL fora de línia (o "per lots") considera aprendre una política a partir d'un conjunt de dades fix sense interacció addicional amb l'entorn.

## Aprenentatge de reforç invers

La RL inversa fa referència a inferir la funció de recompensa d'un agent donat el comportament de l'agent. L'aprenentatge de reforç invers es pot utilitzar per aprendre a partir de demostracions (o aprenentatge d'aprenentatge) inferint la recompensa del demostrador i optimitzant una política per maximitzar els rendiments amb RL. Els enfocaments d'aprenentatge profund s'han utilitzat per a diverses formes d'aprenentatge d'imitació i RL invers.

## Aprenentatge de reforç condicionat per objectius

Una altra àrea activa de recerca és l'aprenentatge de polítiques condicionades a objectius, també anomenades polítiques contextuals o universals pi (a|s,g) que prenen un objectiu addicional g com a input per comunicar un objectiu desitjat a l'agent. La repetició de l'experiència retrospectiva és un mètode per a la RL condicionada a un objectiu que implica emmagatzemar i aprendre dels intents anteriors fallits de completar una tasca. Tot i que un intent fallit pot no haver assolit l'objectiu previst, pot servir com a lliçó sobre com aconseguir el resultat no desitjat mitjançant un reetiquetatge posterior.

## Aprenentatge de reforç multiagent

Moltes aplicacions de l'aprenentatge per reforç no impliquen només un agent únic, sinó una col·lecció d'agents que aprenen junts i s'adapten conjuntament. Aquests agents poden ser competitius, com en molts jocs, o cooperatius com en molts sistemes multiagent del món real. L'aprenentatge de reforç multiagent estudia els problemes introduïts en aquest entorn.

## Generalització

La promesa d'utilitzar eines d'aprenentatge profund en l'aprenentatge de reforç és la generalització: la capacitat d'operar correctament amb inputs no vists anteriorment. Per exemple, les xarxes neuronals entrenades per al reconeixement d'imatges poden reconèixer que una imatge conté un ocell encara que mai no hagi vist aquesta imatge en particular ni tan sols aquest ocell en particular. Com que el RL profund permet dades en brut (per exemple, píxels) com a entrada, hi ha una necessitat reduïda de predefinir l'entorn, permetent que el model es generalitzi a múltiples aplicacions. Amb aquesta capa d'abstracció, es poden dissenyar algorismes d'aprenentatge

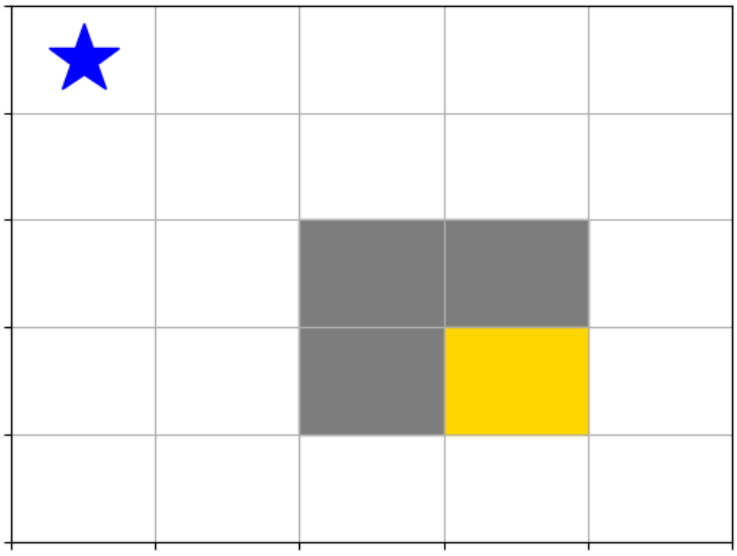
de reforç profund de manera que siguin generals i es pugui utilitzar el mateix model per a diferents tasques. Un mètode per augmentar la capacitat de generalització de polítiques entrenades amb polítiques de RL profundes és incorporar l'aprenentatge de representació.

# 8. Cas pràctic: GridWorld

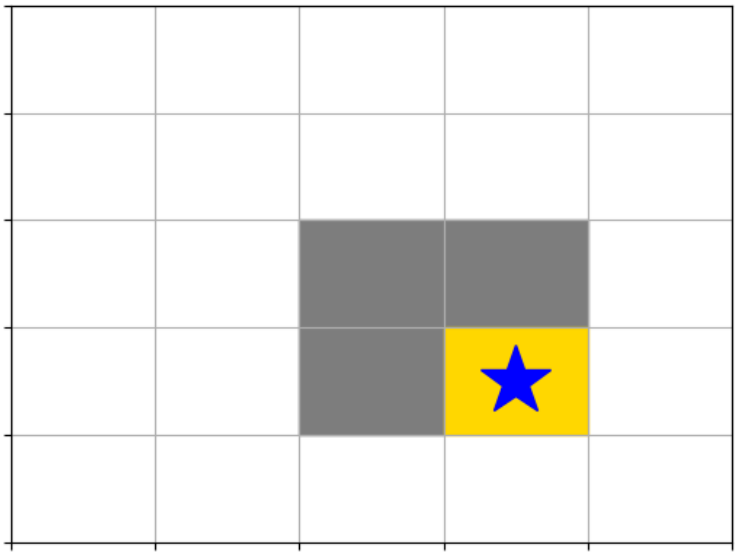
Com a demostració de l'aprenentatge per reforç, veurem com un agent en un món-graella és capaç d'aprendre a arribar a un objectiu esquivant obstacles.

L'agent és l'estrella blava, l'objectiu és la graella daurada i els obstacles són els blocs grisos.

La situació inicial és la següent.



Quan l'agent ha assolit l'objectiu, és a la cel·la daurada.



A continuació veurem els diferents blocs de codi que formen l'exemple.

El codi complet és a https://colab.research.google.com/drive/1Gjb2tuw9V99J3Co3PS8e1OoSdsLSCj7s?usp=sharing

## 8.1. Inicialització

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML
import random

# Grid setup
GRID_SIZE = 5
GOLD = (3, 3)
TRAPS = [(2, 3), (3, 2), (2, 2)]
# Accions possibles: amunt, avall , esquerra, dreta
ACTIONS = [(-1,0), (1,0), (0,-1), (0,1)]
```

Importam les llibreries que necessitarem i definim un món quadrat de 5x5 quadres, amb una sola cel·la daurada, objectiu, i tres obstacles o trampes. Les accions possibles són moviments unitaris horitzontals o verticals, a cel·les adjacents.

La posició (0,0) de la graella és a dalt a l'esquerra.

## 8.2. Recompenses

```python
# Recompenses
def get_reward(state):
    if state == GOLD:
        return 1
    elif state in TRAPS:
        return -1
    else:
        return -0.01  # petita penalització per afavorir camins curts

def valid_state(state):
    return 0 <= state[0] < GRID_SIZE and 0 <= state[1] < GRID_SIZE

def step(state, action):
    new_state = (state[0] + action[0], state[1] + action[1])
    if valid_state(new_state):
        return new_state
    return state
```

La recompensa, definida a la funció **get_reward**, és positiva per a la cel·la daurada i negativa per a les trampes o obstacles.

També hi ha una petita recompensa negativa per a les cel·les buides, que ajudarà a preferir els camins curts.

La funció **valid_state** comprova que la posició és dins la graella, i la funció **step** calcula el següent estat a partir de l'estat actual i l'acció (moviment) realitzat.

## 8.3. Paràmetres

```
# Q-learning parameters
Q = {}
alpha = 0.1
gamma = 0.9
epsilon = 0.2
episodes = 500
```

Definim els valors dels paràmetres per a l'entrenament.

## 8.4. Entrenament

```python
# Entrenament
for _ in range(episodes):
    state = (0, 0)
    while state != GOLD and state not in TRAPS:
        if state not in Q:
            Q[state] = np.zeros(len(ACTIONS))
        if random.random() < epsilon:
            action_idx = random.randint(0, len(ACTIONS) - 1)
        else:
            action_idx = np.argmax(Q[state])
        action = ACTIONS[action_idx]
        new_state = step(state, action)
        reward = get_reward(new_state)
        if new_state not in Q:
            Q[new_state] = np.zeros(len(ACTIONS))
        Q[state][action_idx] += alpha * (reward + gamma * np.max(Q[new_state]) - Q[state][action_idx])
        state = new_state
```

El bucle d'entrenament actualitza la taula Q amb l'equació de Bellman, mentre l'agent no ha arribat a l'objectiu, tantes vegades com hàgim definit.

## 8.5. Camí òptim

```
# Millor trajectòria
def get_best_trajectory(start=(0, 0)):
    path = [start]
    state = start
    while state != GOLD and state not in TRAPS:
        if state not in Q:
            break
        action = ACTIONS[np.argmax(Q[state])]
        next_state = step(state, action)
        if next_state == state or next_state in path:
            break  # avoid loops
        path.append(next_state)
        state = next_state
    return path

trajectory = get_best_trajectory()
```

Anam reconstruint el camí òptim seguint els màxims de la taula Q, començant a la posició inicial (0,0) fins que arribam a la cel·la daurada.

## 8.6. Visualització

```
# Animació

fig, ax = plt.subplots()

def draw_grid(agent_pos):
    ax.clear()
    ax.set_xticks(np.arange(-0.5, GRID_SIZE, 1))
    ax.set_yticks(np.arange(-0.5, GRID_SIZE, 1))
    ax.set_xticklabels([])
    ax.set_yticklabels([])
    ax.grid(True)
    ax.set_xlim(-0.5, GRID_SIZE - 0.5)
    ax.set_ylim(-0.5, GRID_SIZE - 0.5)

    # Cel.les
    for i in range(GRID_SIZE):
        for j in range(GRID_SIZE):
            y = GRID_SIZE - 1 - i
            if (i, j) == GOLD:
                ax.add_patch(plt.Rectangle((j - 0.5, y - 0.5), 1, 1, color='gold'))
            elif (i, j) in TRAPS:
                ax.add_patch(plt.Rectangle((j - 0.5, y - 0.5), 1, 1, color='gray'))

    # Agent
    y = GRID_SIZE - 1 - agent_pos[0]
    x = agent_pos[1]
    ax.plot(x, y, 'b*', markersize=36)

def update(frame):
    draw_grid(trajectory[frame])

anim = animation.FuncAnimation(fig, update, frames=len(trajectory), interval=700)
HTML(anim.to_jshtml())
```

Representam la graella, amb les cel·les de colors i la trajectòria que va seguint l'agent.

# 9. Recursos

Per completar el cas pràctic vist aquí, donarem referències a més exemples amb codi i dos recursos on aprofundir en aprenentatge per reforç.

- Exemples a Keras
- Reinforcement Learning a Kaggle
- Reinforcement Learning a Hugging Face

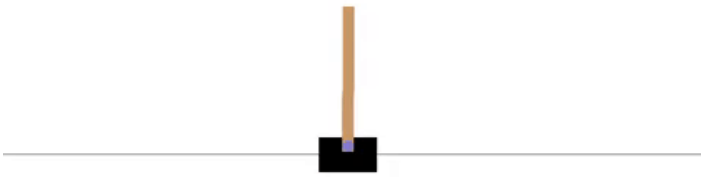
## 9.1. Keras

Keras ofereix quatre exemples interessants d'aprenentatge per reforç

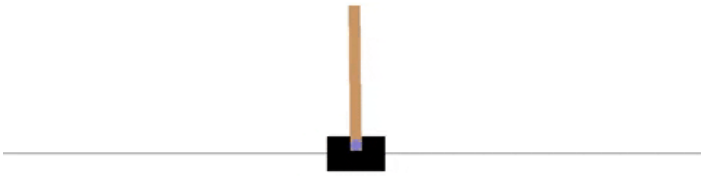
https://keras.io/examples/rl/

Els dos primers resolen l'equilibri d'un bastó vertical damunt un carretó.

Abans de l'entrenament:



Després de l'entrenament:



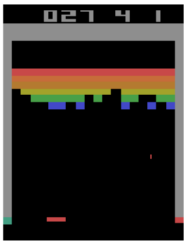
- https://keras.io/examples/rl/actor_critic_cartpole/
- https://keras.io/examples/rl/ppo_cartpole/

El tercer exemple es refereix als videojocs d'Atari.

Abans d'entrenar:

Amb poques iteracions:

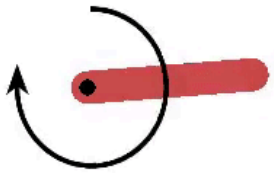

Finalment:




- https://keras.io/examples/rl/deep_q_network_breakout/

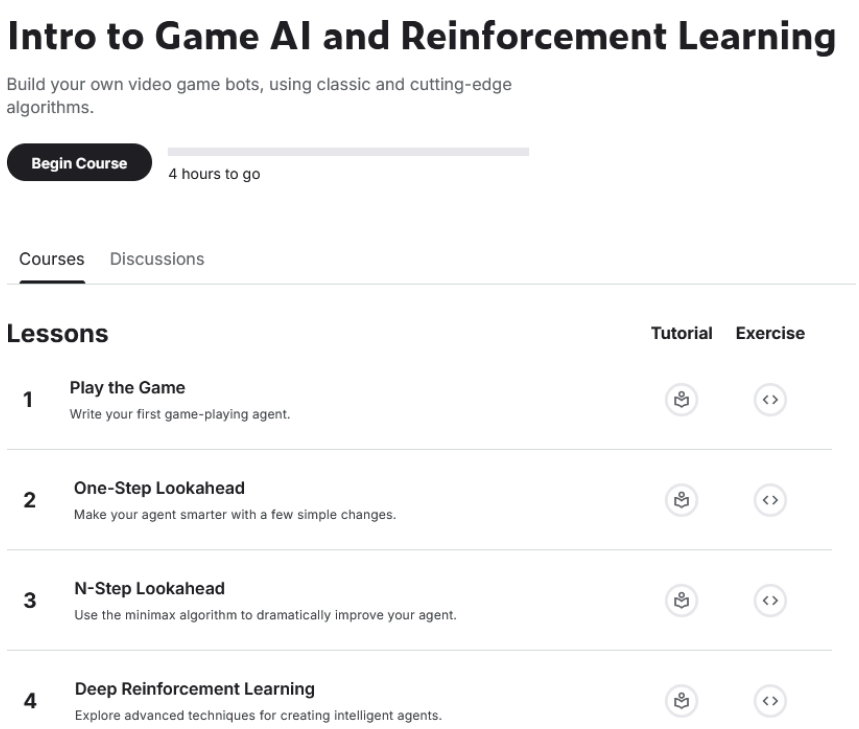
El quart exemple resol l'equilibri d'un pèndol invertit.

Abans d'entrenar:
Després:



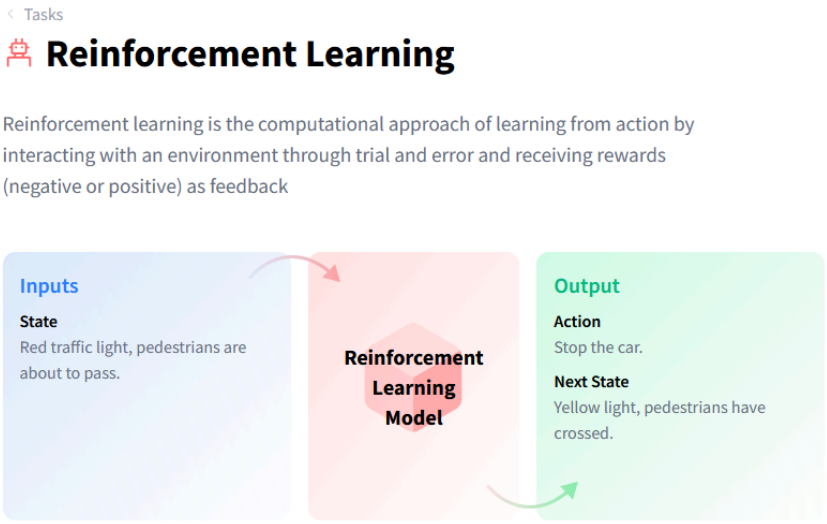
- https://keras.io/examples/rl/ddpg_pendulum/

## 9.2. Kaggle




https://www.kaggle.com/learn/intro-to-game-ai-and-reinforcement-learning

## 9.3. Hugging Face




https://huggingface.co/tasks/reinforcement-learning

# 10. Qui és qui

Acabam el capítol amb una selecció d'investigadors capdavanters en el camp de l'aprenentatge de reforç.

Dos autors fonamentals són **Andrew Barto** i **Richard Sutton**, amb el seu llibre ***Reinforcement Learning***.

**Demis Hassabis** és cofundador i CEO de DeepMind, una de les organitzacions líders en investigació d'aprenentatge per reforç i aprenentatge profund, amb contribucions clau en agents artificials que aconsegueixen rendiment a nivell humà en diversos dominis mitjançant aquest paradigma.

**Peter Dayan** és un investigador destacat en neurociència computacional i aprenentatge per reforç, conegut per treballs que connecten la teoria del reforç amb processos cognitius i biològics, aportant fonaments teòrics importants.

**Volodymir Mnih** va ser un dels primers autors de l'algoritme Deep Q-Network (DQN) a DeepMind, que combina xarxes neuronals profundes amb aprenentatge per reforç per assolir un rendiment humà en jocs Atari, sent un punt d'inflexió en el camp.

**John Schulman** ha desenvolupat importants algorismes d'aprenentatge per reforç com Proximal Policy Optimization (PPO), que milloren l'estabilitat i l'eficiència de l'entrenament d'agents.

**David Silver** és un investigador clau a DeepMind, conegut per liderar l'equip que va crear AlphaGo, el primer sistema d'IA que va derrotar un campió mundial de Go, utilitzant aprenentatge per reforç profund.

**Pieter Abbeel** és un expert en robòtica i aprenentatge per reforç, amb contribucions importants en l'aplicació pràctica de tècniques d'aprenentatge per reforç en entorns reals i simulats.

**Marc Bellemare** ha fet contribucions destacades en l'aprenentatge per reforç, especialment en l'àmbit de l'aprenentatge per reforç distribuït i en la creació de benchmarks com l'Arcade Learning Environment, que ha facilitat la recerca i comparació d'algoritmes.

**Sergey Levine** és reconegut per la seva recerca en aprenentatge per reforç i robòtica, especialment en tècniques d'aprenentatge profund per a control i exploració eficient en entorns complexos.