ToREAD: <https://www.v7labs.com/blog/reinforcement-learning-applications>

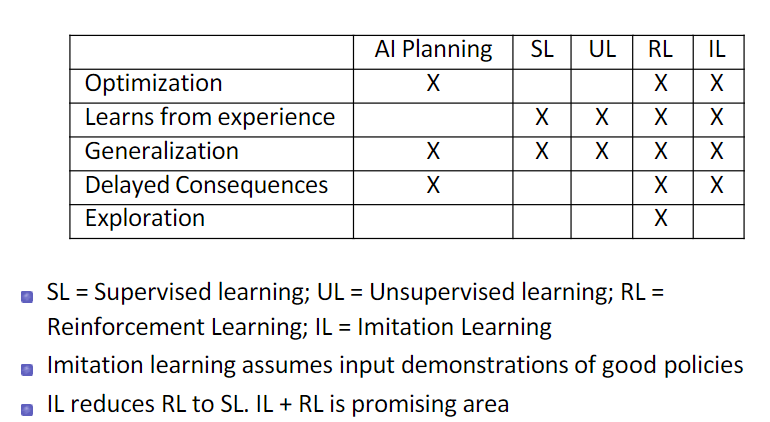
RL se bazeaza pe luarea de decizii, incearca sa optimizeze rezultatele, sa gaseasca o sol care sa generalizeze, sa fie cat mai optima

Aplicatii:

* in self driving cars
* sisteme de recomandare de ex google
* jocuri (go, dota?)
* linii de asamblare
* planificare (optimizarea timpului)

Concepte cheie:

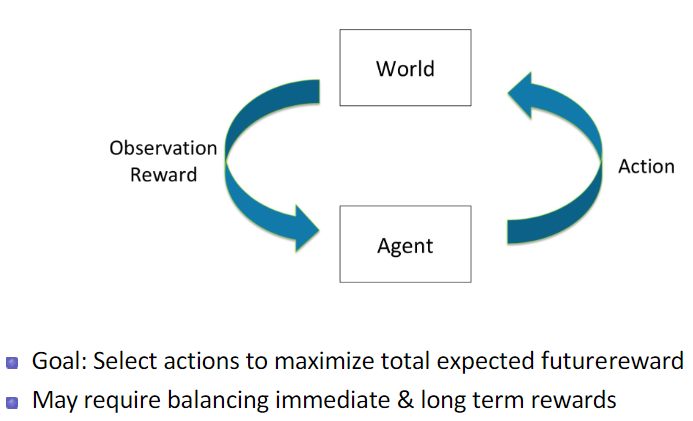
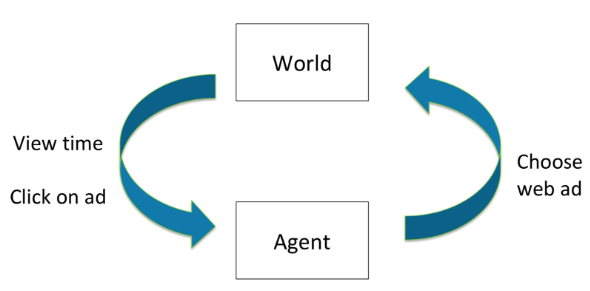
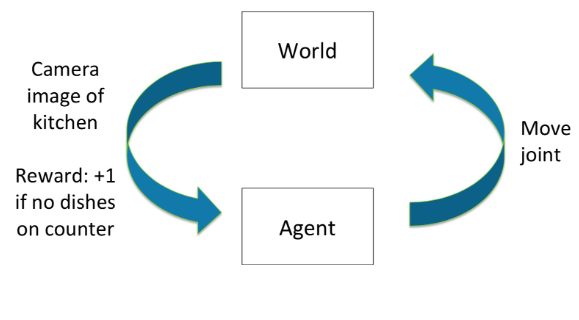
* optimizarea
* consecinte intarziate (de ex la sah, o mutare ti se pare cea mai buna pt urm stare dar pt viitor e cea mai buna?)
* explorare, generalizare (vrem o sol (=set de decizii=**politica**=pi=mapping/asociere al experientelor trecute (stari) catre actiuni) care se potriveste pt acelasi tip de pb cu variatii)



* model
* functia valoare = recompensele pe care le putem obt in viit, dintr-o anumita stare, cu/fara o anumita actiune
* pt a vedea ce actiune ar fi fost mai buna pt o anumita stare/situatie
* acumuleaza recompense, uneori si probabilitati

Plan:

* Avem un mediu (ex: tabla de joc) explorat de agentul care ia decizii pe parcurs
* avem un reward (label) la fiecare pas: decizie buna=>reward pozitiv, decizie proasta=>r.negativ
* alg isi imbunatateste deciziile de la pas la pas in f de ce reward observa ca primeste

ex: web advertising, robot care descarca vase din masina de spalat, blood pressure control

* putem reprezenta ca un graf (nu f bn definit) starile adica deciziile (deci graful se poate imbunatati pe parcurs)
* la fiecare pas de decizie avem temporalitate
* avem episoade (asem cu epocile de la IA) care contin mai multe stari/decizii
* cel mai naiv posibil agent este cel random

Types of RL Agents

* Model-based
* Explicit: Model
* May or may not have policy and/or valuefunction
* Model-free
* Explicit: Value function and/or policy function
* No model

Lab 1

* partea de evaluare
* partea de implementare / imbunatatire a politicilor

Curs 2

* recompensele pe termen scurt sunt mai imp decat cele pe termen lung? FALS
* un factor discount mare => pe termen lung rewardul conteaza mai putin <=> reduce importanta rewardurilor
* un factor discount 0 => decizia (rewardul) imediata conteaza cel mai mult

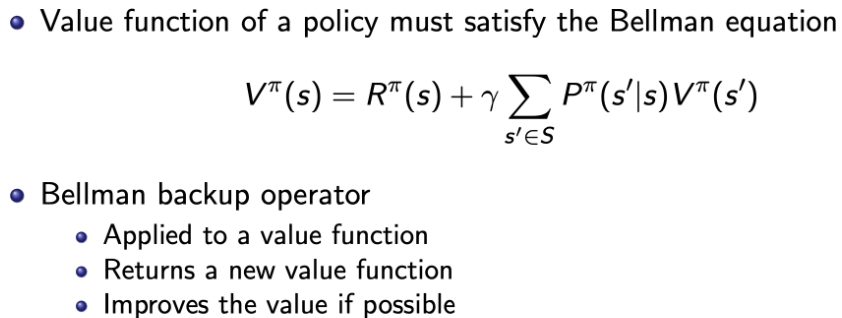
**K-order Markov Process**

* vedem problema ca pe una fara solutie exacta ci trb sa gasim cea mai optima solutie
* spune cat de mult ne avantajeaza sa ajungem intr-o stare s
* exemple: mana-robot, managementul inventarului (cum sa mentinem stocurile)
* **obs: toate rez din viit sunt det de mom curent, nu de cel anterior**
* orizont = ce poate prezice agentul si vedea in fata (va fi cu un pas inainte), in teorie trb sa fie infinit sa ne putem imagina ce se intampla in viitor, dar practic trb sa i punem o limita

Markov Decision Process (MDP):

* avem un set finit de stari si unul de actiuni astfel incat stim ca trebuie sa optimizam profitul dar nu stim exact ce cuatum trebuie sa crestem/descrestem (de ex la un magazin, nu stii din prima ce si cata marfa sa cumperi)
* avem dinamica/ tranzitiile actiunilor
* avem functia de recompense
* si factorul de discount -> variaza mereu, impreuna cu nr de stari
* putem crea o matrice de tranzitii

Obs: If policy doesn’t change with time, it will never change again



Lab 2 - Alg Monte-Carlo

* trial and error
* gasim o medie a ce primim ca recompense = sol optima facand iteratii

Curs 3 – Monte Carlo

* MC methods learn directly from episodes of experience (nu are un nr de pasi determinat)
* MC ismodel-free: no knowledge of MDP transitions / rewards (doar stim regulile jocului si determinam pe parcurs tranzitiile si recompensele)
* MC learns from complete episodes: no bootstrapping
* can only apply MC to episodic MDPs => All episodes must terminate

Curs 4 – MDP

**Politici MDP**

**Politica** = set de acțiuni pe care le putem lua în fiecare stare în parte.​

* se poate invata pe parcurs
* poate fi determinista sau sochastica (cu probabilitati)
* se itereaza: ne ajuta sa ajungem la val optima

Politica optima: nu e neaparat unica.

Există o singură funcție optimă de tip value function, dar multiple politici cu aceeași valoare optimă!​

Politica optimă pentru MDP cu orizont infinit este deterministă si staționară;​ se cauta cu brute-force.

Policy Iteration ne ajută să ajungem atât la valori optime, cât și la politici asociate!​

**State Action Value** = Q(s,a) = in starea s, executam actiunea a, apoi urmam politica pi