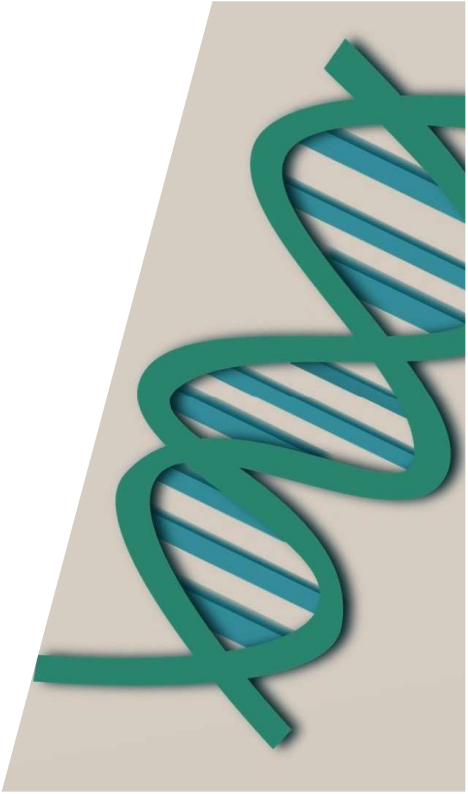
Chapitre 2

Algorithme Génétique et Renforcement

Ressource R4.04 - Méthodes d'Optimisation

Tiphaine Jézéquel

2024-2025

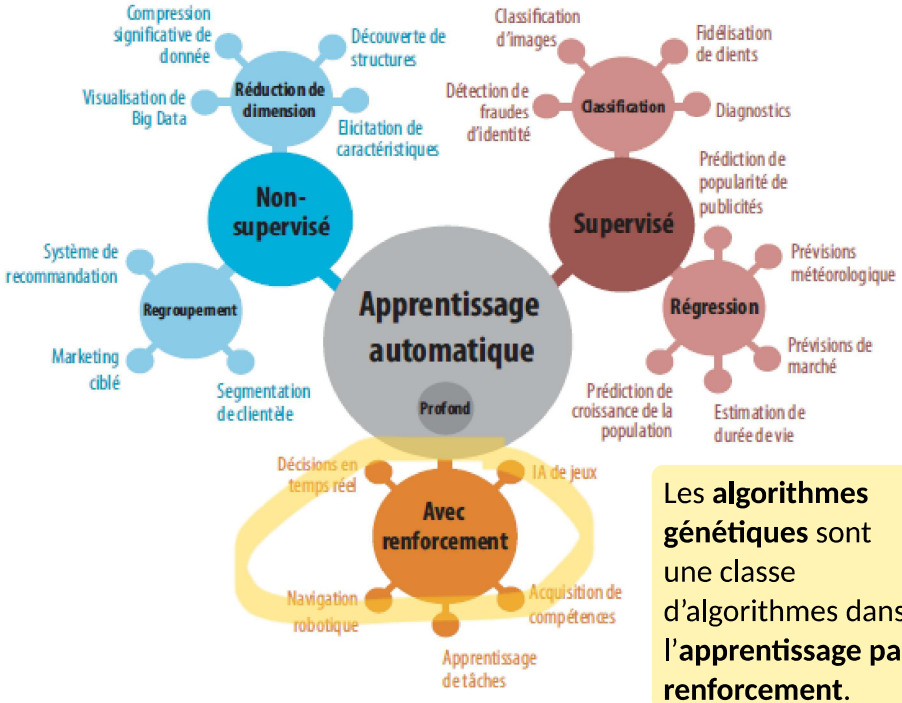


Plan du cours

1. Algorithmes génétiques, utilité
2. Principe des Algorithmes génétiques
3. Etapes d'un Algorithme génétique
4. Algorithme génétique et problème du voyageur de commerce
5. Apprentissage par renforcement et jeu du macareux

Selon les informations disponibles durant la phase d'apprentissage, on le qualifie de différentes manières :

- Si les données sont étiquetées (c'est-à-dire que la réponse à la tâche est connue pour ces données), il s'agit d'un **apprentissage supervisé**.
- Dans le cas sans étiquette, on cherche à déterminer la structure sous-jacente des données (qui peuvent être une densité de probabilité) et il s'agit alors d'**apprentissage non supervisé**.
- Si le modèle est appris de manière incrémentale en fonction d'une récompense reçue par le programme pour chacune des actions entreprises, on parle d'**apprentissage par renforcement**.



Les **algorithmes génétiques** sont une classe d'algorithmes dans l'**apprentissage par renforcement**.

1. Algorithmes génétiques, utilité

• Repères historiques

- Paternité en général attribuée à John Henry Holland avec un livre entier sur ce sujet publié en **1975** : *Adaptation in natural and artificial systems*

Comme souvent, ce n'est pas l'affaire d'un seul homme, mais l'aboutissement de travaux d'équipes :

- **1950** : la "learning" machine de Alan Turing s'inspire des principes de l'évolution
- **1954** : le biologiste Nils Aall Barricelli simule l'évolution par ordinateur
- **1957** : le généticien australien Fraser travaille sur la sélection artificielle
- **années 60-70** : devient une méthode d'optimisation reconnue pour des problèmes d'ingénierie (équipe allemande de Rochenberg et Schwefel, équipe américaine de Fogel, équipe de Holland).

• Qualités / défauts

- Les algorithmes génétiques sont "tout-terrain", ils marchent quasiment pour tout type de problème d'optimisation.
- On n'est jamais sûr qu'ils convergent vers la solution optimale… mais permettent en général de trouver "une bonne solution", si c'est ce que l'on cherche.
- La vitesse de convergence peut laisser à désirer : pour des problèmes particuliers (par exemple si la fonction-objectif est dérivable, linéaire, convexe, etc.), on a en général des algorithmes plus efficaces.
  Par exemple : pour une fonction-objectif convexe et dérivable, l'algorithme du gradient sera + efficace.

- **Utilisation actuelle**

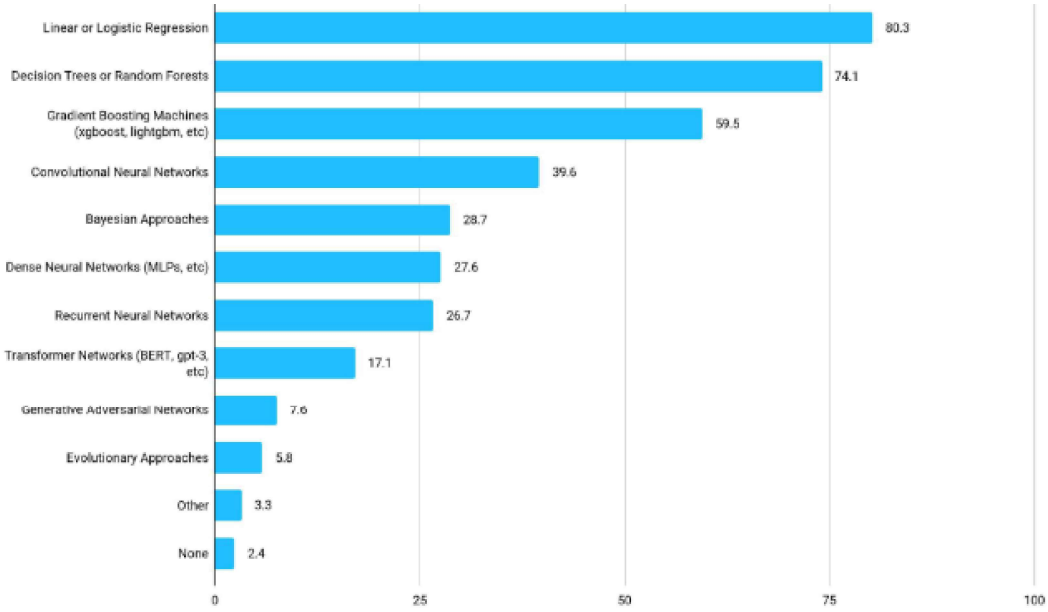
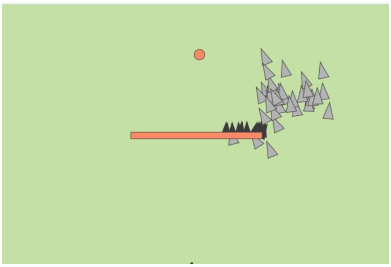
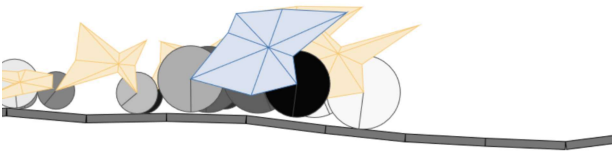


Figure: Methods and Algorithms Usage (Kaggle - State of ML & Data Science 2021)

- **Exemples**
  - **2017** Dylan Audic, étudiant à l'IUT de Lannion :

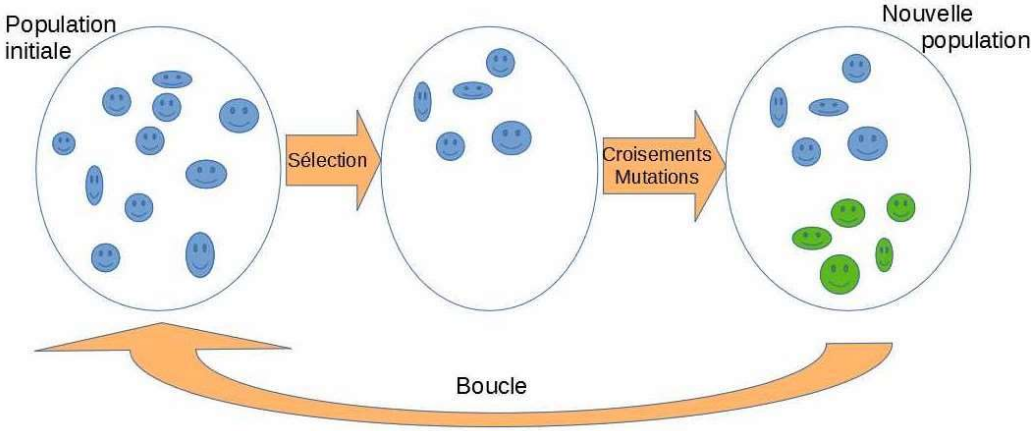


  - Genetic Cars



Lien vers l'exemple : `https://rednuht.org/genetic_cars_2/`

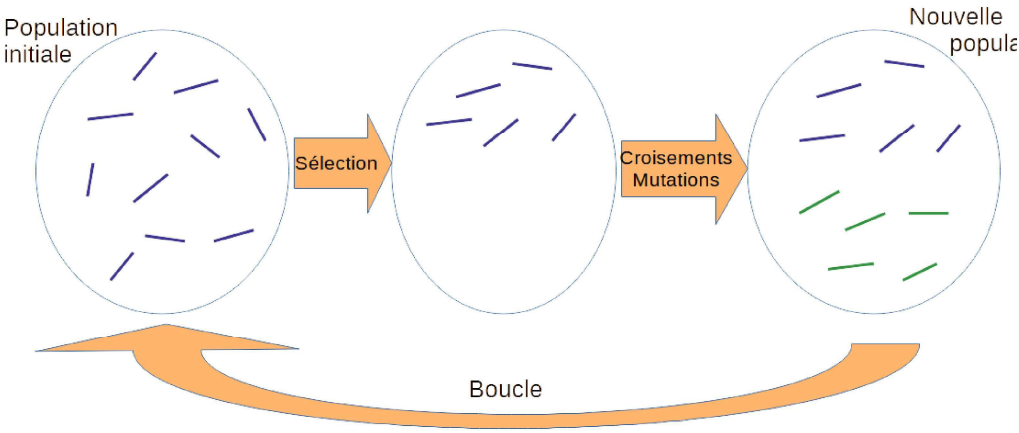
• Principes des Algorithmes génétiques



On traduit notre problème d'optimisation avec du "vocabulaire génétique" :

Environnement	Les données connues sur le problème d'optimisation (fonction à optimiser, contraintes...)
Individu	Une solution admissible du problème.
Population	Groupe d'individus = groupe de solutions admissibles.
Sélection	Elimination des individus/solutions les "moins adaptés à l'environnement" = pour lesquels la fonction-coût est la plus élevée
Croisement	Création d'un nouvel individu/solution (un "enfant") à partir de 2 individus/solutions (les "parents").
Mutation	Modification aléatoire d'un individu/ solution.

- **Application à la régression linéaire**



Environnement	Les données connues sur le problème d'optimisation (fonction à optimiser, contraintes...)
Individu	Une solution admissible du problème.
Population	Groupe d'individus = groupe de solutions admissibles.
Sélection	Elimination des individus/solutions les "moins adaptés à l'environnement" = pour lesquels la fonction-coût est la plus élevée
Croisement	Création d'un nouvel individu/solution (un "enfant") à partir de 2 individus/solutions (les "parents")
Mutation	Modification aléatoire d'un individu/ solution.

3. Etapes d'un Algorithme génétique

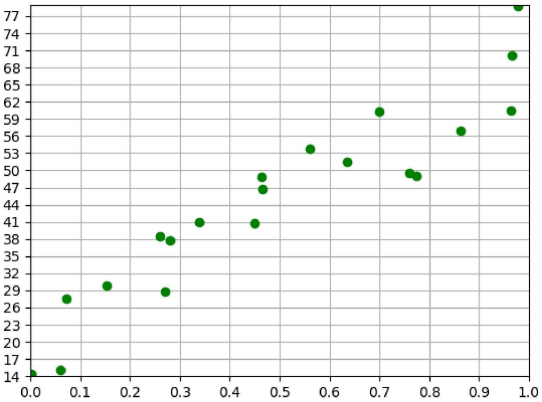
• La "création" / l'environnement

L'algorithme s'applique sur un jeu de données :

- pour un "vrai problème" il est donné
- pour faire des tests, on peut créer des données aléatoirement

Exemple de la régression linéaire :
On peut créer des nuages de points aléatoirement pour tester notre algorithme. Chaque nuage de points est un environnement de départ différent.

Ci-contre, données aléatoires du type du problème de l'âge des adultes en fonction de la proportion de cheveux blancs.

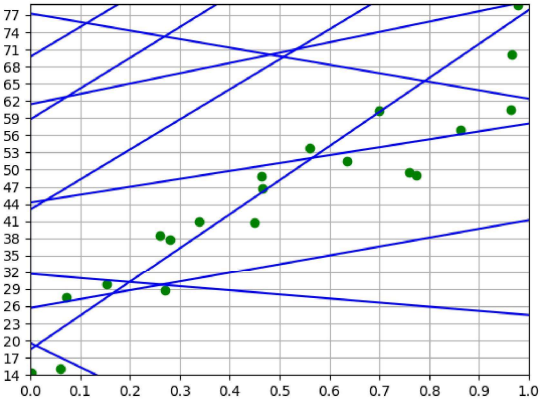


• La "génèse" / initialisation aléatoire

On initialise l'algorithme avec un ensemble de N solutions admissibles créées aléatoirement.
Dit avec le vocabulaire génétique : on crée une population de départ constituée de N individus aléatoires.

Exemple de la régression linéaire :
On initialise l'algorithme en créant un ensemble de droites aléatoires.

Ci-contre, population de 10 droites créées aléatoirement.

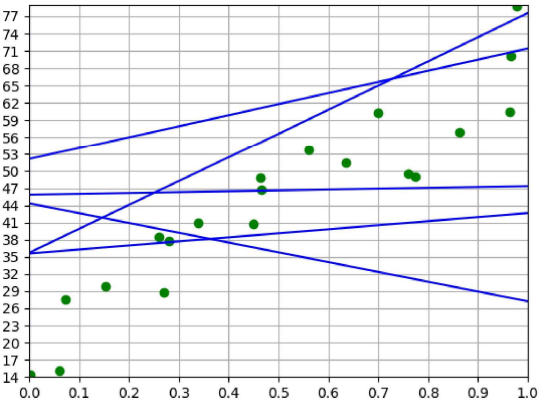


- **La "sélection naturelle" / sélection des meilleures solutions**

Parmis la population de départ, on sélectionne la "meilleure moitié". C'est-à-dire qu'on calcule la valeur de la fonction-coût pour chacun des N individus de la population, et on garde les $N/2$ individus pour lesquelles la valeur est plus petite.

Exemple de la régression linéaire :
Parmis les 10 droites créées lors de l'étape précédente, on sélectionne les 5 droites pour lesquelles l'erreur quadratique est la plus petite.

Ci-contre, les 5 droites sélectionnées.



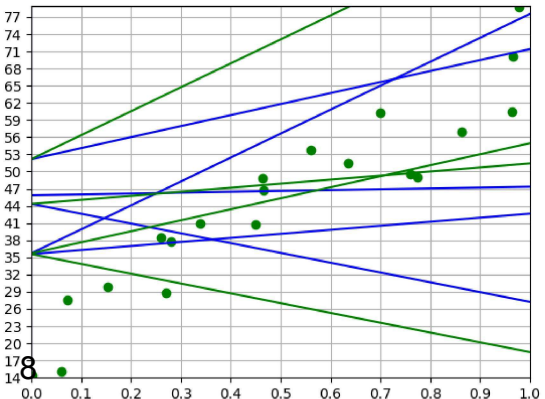
- **Les croisements**

A partir des $N/2$ individus sélectionnés à l'étape précédente, on crée une "descendance" de $N/2$ nouveaux individus.
Pour cela, avec des "couples" d'individus sélectionnés (les "parents"), on crée un nouvel individu (l'"enfant") en mélangeant les caractéristiques des 2 "parents". On crée ainsi $N/2$ "enfants".

Il n'y a pas de méthode générale pour réaliser ces croisements, on peut faire des choix très différents... plus ou moins efficaces.

Exemple de la régression linéaire :
On utilise les $N/2$ droites sélectionnées lors de l'étape précédente pour créer $N/2$ nouvelles droites.

Ci-contre, les 5 droites sélectionnées (en bleu), et les 5 nouvelles droites créées (en vert).

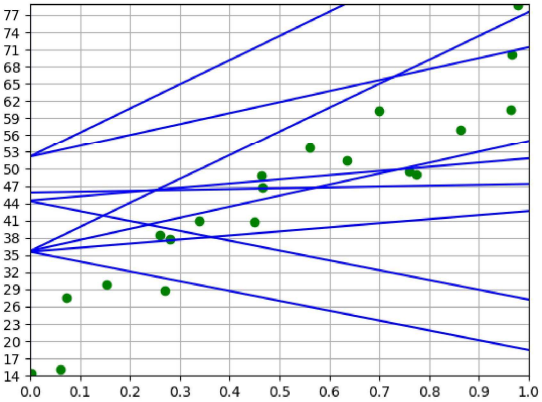


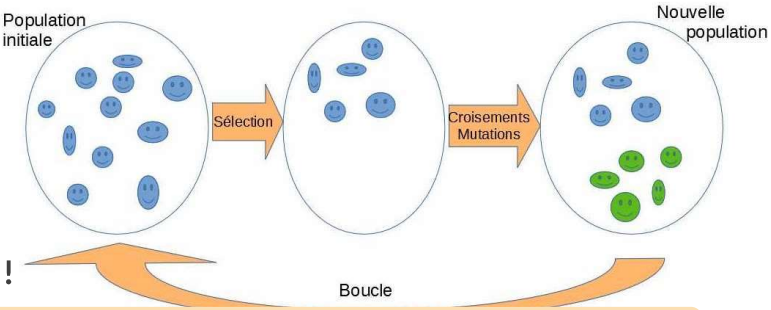
- **Les mutations**

On fait des ”mutations” sur une certaine proportion de la population. C’est-à-dire qu’on choisit aléatoirement un certain pourcentage de la population, et qu’on applique de petites modifications aléatoires à ces individus.

Exemple de la régression linéaire :
On choisit de faire des mutations pour 30% des 10 droites obtenues à l’étape précédente (individus sélectionnés + enfants).

Ci-contre, les 10 droites après mutations aléatoires.



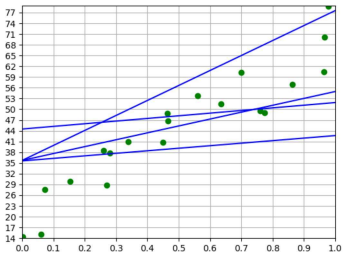


- **... et on boucle !**

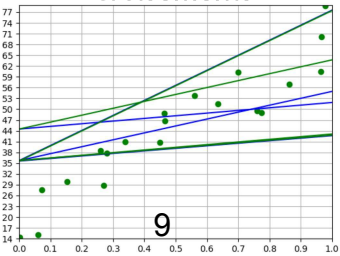
On a à nouveau une population de N individus : les $N/2$ individus sélectionnés + les $N/2$ ”enfants” créés.
On peut repartir à l’étape ”sélection”, on a ainsi un processus itératif.

Exemple de la régression linéaire. Itération suivante :

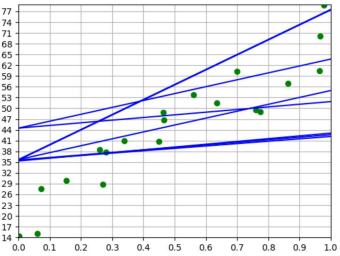
- Sélection



- Croisements



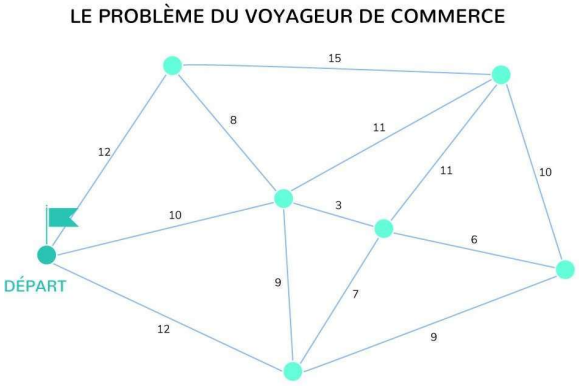
- Mutations



4. Algorithme génétique et problème du voyageur de commerce

Problème du voyageur de commerce

Étant donnés n points (des villes) et les distances séparant chaque point, trouver un chemin de longueur totale minimale qui passe exactement une fois par chaque point et revienne au point de départ.



Un “problème classique” de mathématiques, étudié depuis le XIXe siècle (Hamilton, Kirkman).
Un problème qui a l’air simple... mais ne l’est pas.

Nombre de villes n	Nombre de chemins candidats $\frac{1}{2}(n-1)!$
3	1
4	3
5	12
6	60
7	360
8	2 520
9	20 160
10	181 440
15	43 589 145 600
20	$6{,}082 \times 10^{16}$
71	$5{,}989 \times 10^{99}$

Aujourd’hui, les meilleurs calculateurs font environ 10 milliards de milliards d’opérations par secondes. A 25 villes cela donne 8 heures de temps de traitement. Mais si on passe à 30 villes on obtient 4.10^{11} secondes ou 14 000 siècles.

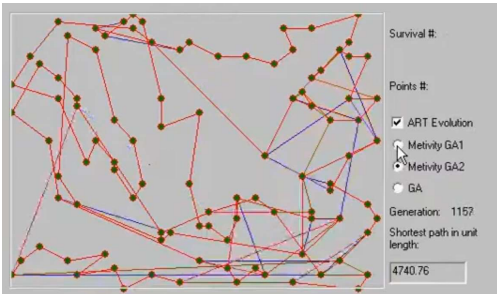
$\longrightarrow$ on va devoir trouver un algorithme qui donne des solutions approchées du problème.

• Algorithme génétique avec le Voyageur de Commerce

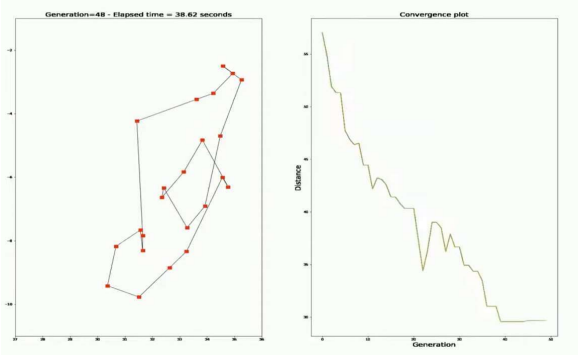
On traduit le problème avec du ”vocabulaire génétique” :

Environnement	Les données du problème.
	.....
Individu	Une solution du problème.
	.....
Population	Groupe d’individus = de solutions du problème.
Sélection	Elimination des individus, des solutions les moins adaptés.
	.....
Croisement	Croisement de deux individus, deux solutions, pour en produire de nouveaux.
Mutation	Modfication aléatoire d’un individu, d’une solution.

Par le biais d’algorithmes génétiques, il est possible de trouver des chemins relativement corrects. De plus, ce type de problèmes est assez facile à coder sous forme d’algorithme génétique.



www.youtube.com/watch?v=94p5NUogClM



www.youtube.com/watch?v=-DytTeQ6mWg

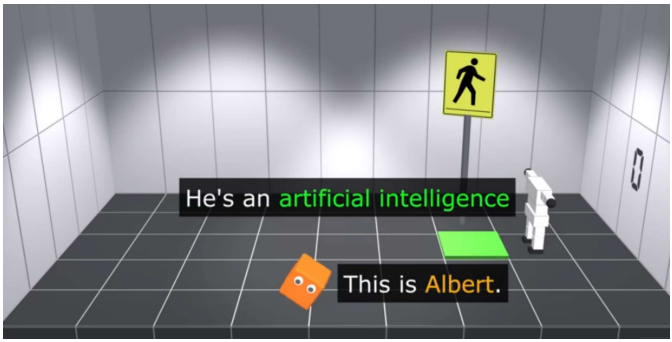
- **Repères historiques**
  - L'apprentissage par renforcement est né des processus de décision markoviens **1957** où : un agent prend des décisions et les résultats de ses actions sont aléatoires.
  - **1989** : Le Q-learning est historiquement l'un des premiers algorithmes d'apprentissage par renforcement connus. Il est mis au point par Chris Watkins durant sa thèse.

Q-learning

Le Q-learning est un algorithme qui ne nécessite pas de modèle initial de l'environnement. La qualité (d'où le "Q") des actions de l'agent est évaluée à un instant t du système.

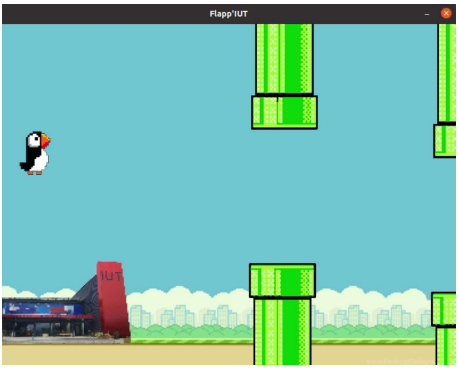
Definition

L'**apprentissage par renforcement** consiste, pour un agent autonome (ex. : robot, agent conversationnel, personnage dans un jeu vidéo, etc.), à apprendre les actions à prendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps.



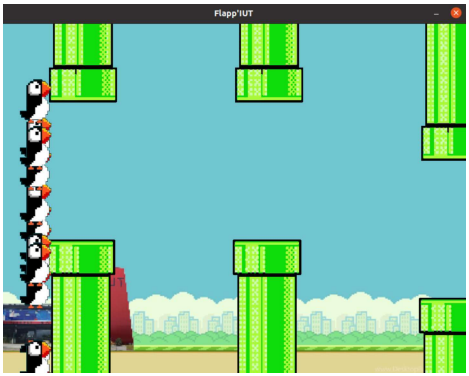
www.youtube.com/watch?v=L_4BPjLBF4E

• Apprentissage par renforcement sur le jeu du Macareux



Pour l'algorithme d'apprentissage par renforcement que nous allons réaliser, l'agent est un Macareux qui doit éviter une série (infinie) de tuyaux. Son seul mouvement possible est le saut.

⟶ La récompense quantitative à optimiser sera ici la distance parcourue par le Macareux.



Nous allons appliquer un algorithme génétique : plusieurs Macareux seront générés et dévieront légèrement de trajectoires (aléatoirement). Seuls les Macareux les plus performants resteront en vie, et l'opération pourra être répétée.

On rajoutera une tactique qui est de l'ordre du renforcement mais n'apparait pas dans un algorithme génétique : on regardera si les macareux perdants ont heurté un tuyau en bas ou en haut, pour modifier la trajectoire en fonction.