Quelques pistes pour enseigner l'Intelligence Artificielle au lycée

LABOMATH LPLA

Jérôme Fellus

17 JUIN 2019

L'Intelligence Artificielle : définition & périmètre

Acte de naissance : 1956

John Mc Carthy : “the science and engineering of **making intelligent machines**”.

- 1963 : Création de laboratoires d'IA au MIT (MAC) et à l'Université de Standford (SAIL)
- Conjecture : "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it"
- **Comprendre/Modéliser l'intelligence + Imiter/Reproduire les comportements intelligents**

Marvin Minky : "Construction de programmes informatiques qui s’adonnent à des tâches qui sont, pour l’instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que l’apprentissage perceptuel, l’organisation de la mémoire et le raisonnement critique.”

Alan Turing : "Les machines peuvent-elles penser ?"

- **Test de Turing** : une machine est intelligente si, lors d'une **conversation en langage naturel** avec un juge humain, celui-ci ne peut distinguer si il s'agit d'une machine ou d'un autre humain
- **Jeu d'échecs** : "La drosophile de l'IA" *(J.Mc Carthy)*
- **Voiture sans conducteur** => robotique autonome

Deux philosophies

IA forte - position idéologique

Postulat : Les émotions, la consience de soi et l'ensembles des manifestations de l'intelligence sont des **processus biochimiques**. Produire une forme d'intelligence totalement artificielle est donc **possible**, Le défi réside dans la traduction de ces processus sur un substrat non-biologique.

- **Machine universelle de Turing** : La seule limite d'un ordinateur est la **calculabilité**
- L'intelligence est-elle calculable ?
- Créer des machines intelligentes nous révèle la mécanique de notre propre intelligence
- L'IA comme **science cognitive experimentale**

IA faible - position pragmatique

Il n'est pas necessaire de **définir** l'intelligence ou d'en **reproduire** le fonctionnement exact.
Il suffit que la machine "**semble intelligente**" à un observateur exterieur (=> test de Turing)

Exemple : Psychologue ELIZA. "Les programmes semblent intelligents, mais ils ne le sont pas. En réalité, ils ne **comprennent rien**." *(J. Weisenbaum)*

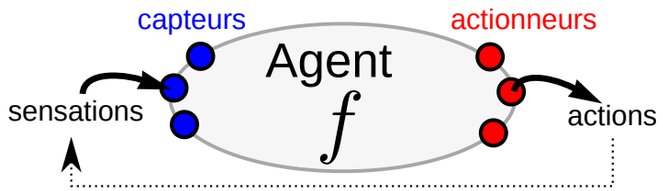
- Égaler/depasser les humains sur des **tâches spécifiques** réputées "difficiles"
- Gagner en **autonomie**, s'abstraire de la supervision du programmeur
- Agentisation : Immersion dans un **environnement réel** inconnu et dynamique

Raisonnement | **Interaction** | **Apprentissage**

Agent rationel

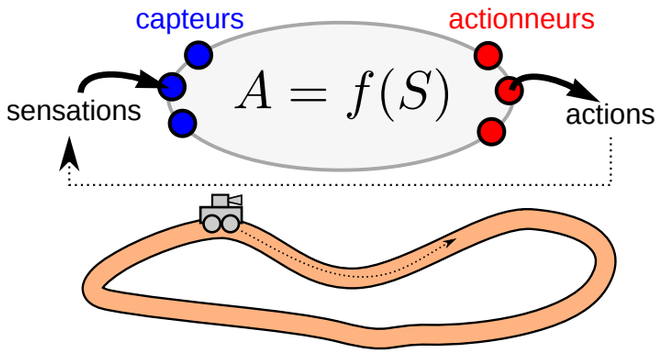
- Trier 1000 **nombres** du plus petit au plus grand n'est pas une tâche d'IA
- Trier 1000 **cailloux** du plus petit au plus grand est une tâche d'IA.
  Pourquoi ?
  - **Interaction** avec un **environnement réel** -> Necessité de **percevoir** et **agir** -> agent
  - **Agent rationel** : Poursuit un but dont les moyens de satisfaction sont *a priori* inconnus
  - **Rationalité limitée** : l'agent a une connaissance du monde réduite et des capacités limitées

Monde



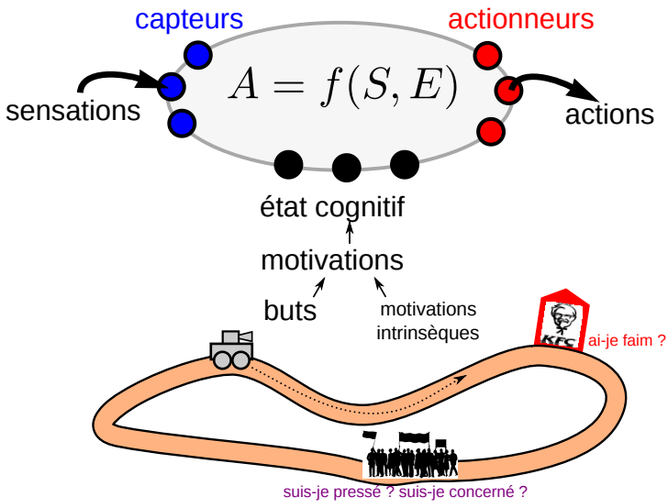
boucle d'interaction => système dynamique

AGENT REACTIF



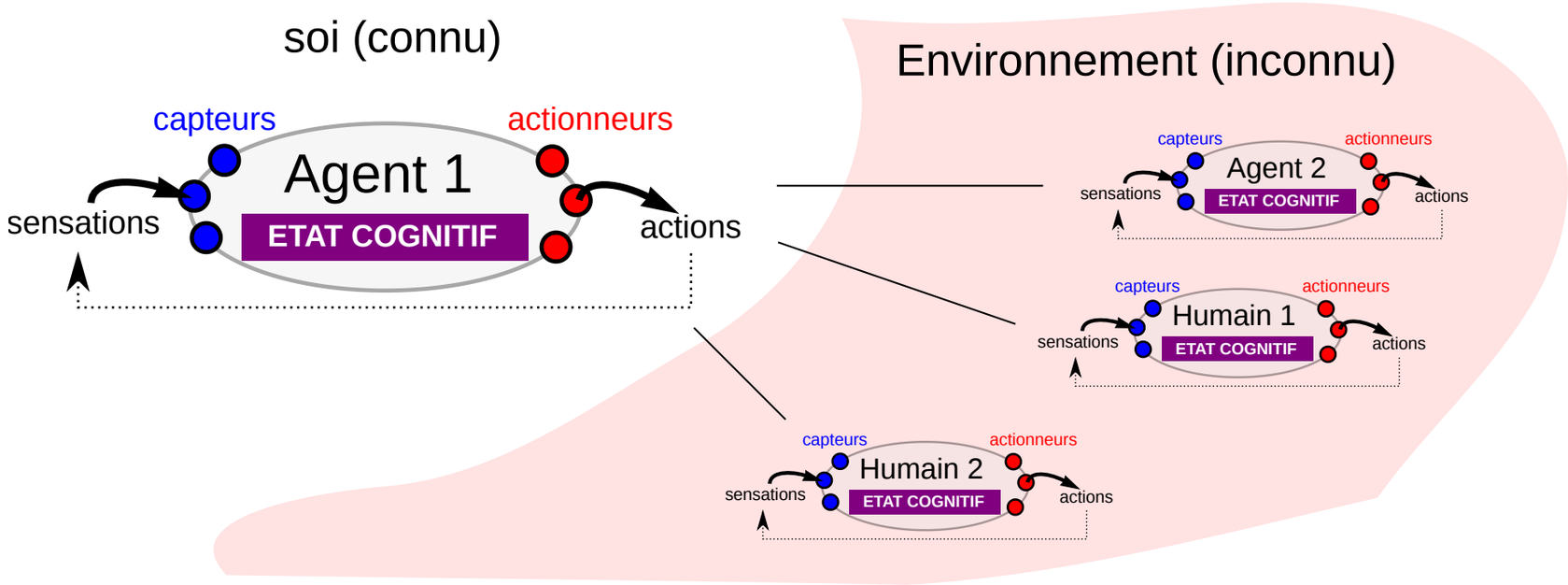
Ne réalise qu'un seul comportement

AGENT COGNITIF



Adapte son comportement au contexte

Interaction avec l'environnement & cognition située



Connaissance incomplète => Apprentissage et Inférence en présence d'incertitude

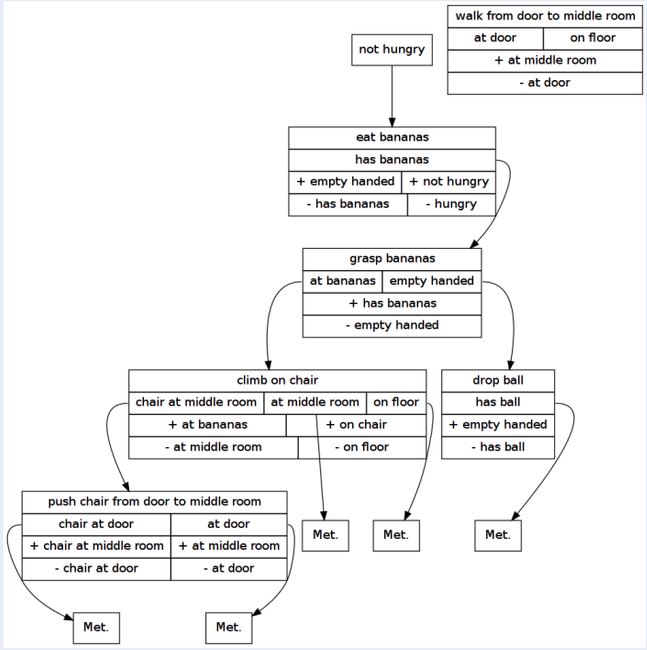
- **Point de vue logique** : enrichir la connaissance par collecte de faits + **raisonnement inférenciel**
  - **Déductif** : ("la nuit, tous les chats sont gris" +  + 0h00) => ce chat est gris
  - **Inductif** : (0h00 +  , 4h00 + ) => "la nuit, tous les chats sont gris"
- **Point de vue statistique et théorie de la décision** :
  - Estimer la **vraisemblance** de diverses **hypothèses** à partir **d'observations**
  - Adopter l'hypothèse qui **minimise le risque** d'erreur
  - Evaluation du comportement via une **fonction d'objectif**

IA Symbolique vs. IA Connexionniste

IA Symbolique

Raisonnement formel, **logique** - Recherche dans des structures combinatoires (**graphes**)

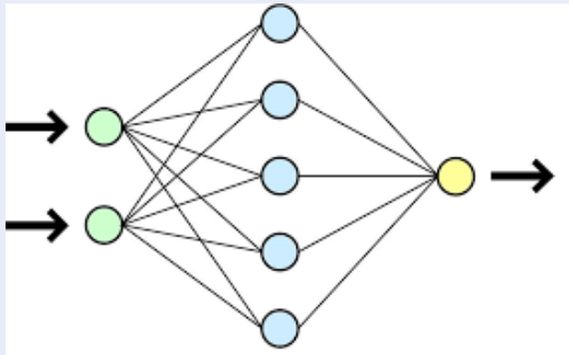
- **General Problem Solver** *(Newell & Simon 1959)* - logique des predicats, geométrie euclidienne, tours de Hanoi, *etc*
- Langage **Prolog** (Colmerauer & Roussel 1972)
  resolution par calcul des prédicats du 1er ordre
- Graph orienté de clauses de Horn $(a_1, a_2, \ldots, a_n) \Rightarrow b$
  clauses "*question*" <- clauses "*implication*" <- clauses "*fait*".
- (+) Trouve automatique le chemin de résolution
  (-) representation symbolique => **explosion combinatoriale**
  e.g., incapable de conduire une voiture...
  Comment convertir les signaux sensoriels brut en "faits" ?



IA Connexionniste

Postulat : Les processus mentaux peuvent être simulés par un **réseau d'opérateurs élémentaires**

- **Réseaux de neurones** : inspiration bio-mimetique de l'architecture du cerveau
- La connaissance comme processus **emergent** :
  (+) pas besoin de convertir les signaux en faits.
  (-) les faits ne sont pas précisément localisés en mémoire -> **opacité** algorithmique. **Capacité d'abstraction ?**



Se ramener à un problème d'optimisation

Fonction de coût

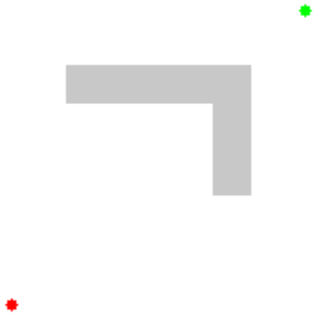
$\mathcal{L} : S \times X \mapsto \mathbb{R}$ évalue la qualité d'un état S du système vis-à-vis des données X

But : trouver l'état du système $S^\star \in \mathcal{H}$ qui minimise le coût $\mathcal{L}(S, X)$ sur le jeu de données X

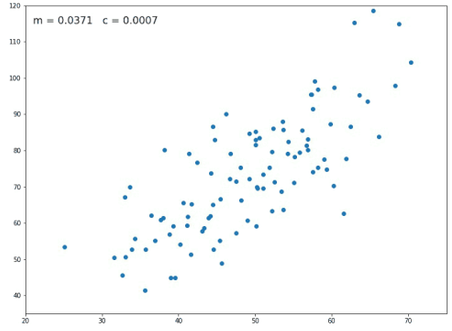
Espace de recherche / Classe d'hypothèses

- $\mathcal{H}$ espace de recherche **discret** fini -> **optimisation combinatoire** (S = "stratégie")
  - Parcours efficient du graphe d'états
  - Accélération de la recherche par des **heuristiques**
- $\mathcal{H}$ espace vectoriel **continu** de dimension finie -> **optimisation numérique** (S = "modèle")
  - **Solution analytique** accessible
  - **Résolution itérative** par approximations successives

Exemple d'optimisation combinatoire :
A-star



Exemple d'optimisation numérique :
Descente de Gradient



Apprentissage Statistique (Machine Learning)

Fonction de coût

Modèle : $f_\theta : X \mapsto Y$ **Paramètres** : $\theta \in \Theta$ **Jeu d'entraînement** : $D = (d_1, \ldots, d_n)$

Objectif : $\theta^\star = \arg\min_{\theta\in\Theta} \mathcal{L}(\theta, D) = \sum_i^n \ell(\theta, d_i)$

Apprentissage supervisé : Jeu de couples (entrée,sortie) $D = ((x_i, y_i))_i^n$

coût = erreur de prédiction, *e.g.*, erreur quadratique: $\sum_i (y_i - f(x_i))^2$

Regression : $Y = \mathbb{R}^m$

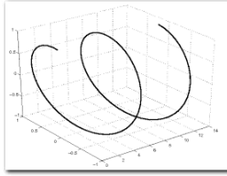
Classification : $Y = [1..m]$

Apprentissage non-supervisé : Jeu d'entrées $D = ((x_i))_i^n$, coût = fidélité aux données

Estimation de densité
modèle = distribution de proba
coût = vraissemblance $-\ln f_\theta(x_i)$

Réduction de dimension: $X = \mathbb{R}^q, Y = \mathbb{R}^p,\ p \ll q,$
coût = erreur de reconstruction, preservation de distance,
e.g., $(\|x_i - x_j\|_q^2 - \|f(x_i) - f(x_j)\|_p^2)^2$

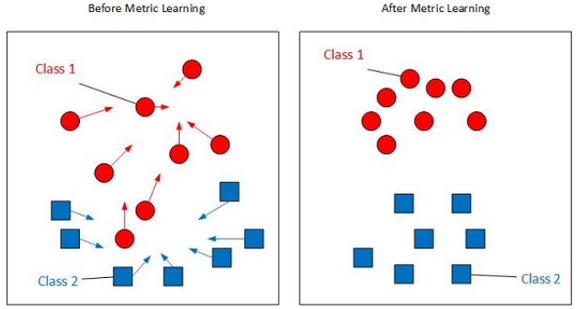
Clustering (categorisation): $Y = [1..m]$ (catégories),
coût = integrité de chaque catégorie,
e.g., $\sum_i \min_k^m (x_i - \mu_k)^2$ (variance intra-catégorie)



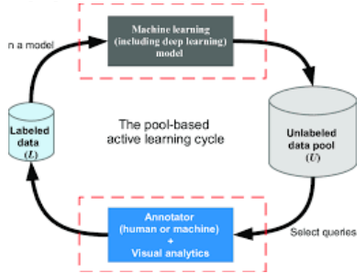
Extraction de variété riemannienne

Apprentissage semi-supervisé

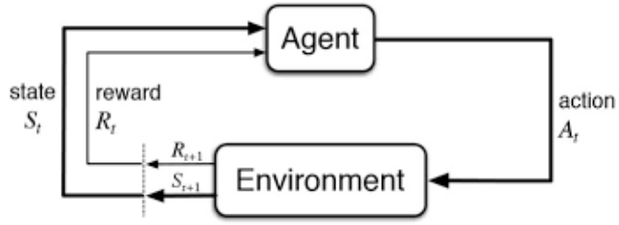
Apprentissage de noyau/metrique



Apprentissage actif



Apprentissage par renforcement



Sciences cognitives, intelligence du corps et apprentissage développemental

L'intelligence animale est un processus développemental

- Un bébé ne resoud que des tâches "simples". La manipulation, le langage, la motricité, s'acquièrent à des stades particuliers de son **développement épigénétique**
- **Découverte de soi**: mon corps émerge de l'environnement par ma capacité à le contrôler
- **Paradigme perception-action**: ma perception des entités de l'environnement se limite aux effets des mes actions, que je découvre au fil de l'experience -> **affordance** *(Gibson 1979)*
- **Schéma corporel**: l'utilisation d'outil comme une extension du corps accessible au contrôle
- **Plasticité cérébrale**: La structure du réseau de neurones est elle aussi adaptative

Intelligence du corps

- Le système musculo-squelétique animal est **prédisposé** à certaines tâches de coordination
- **Exemple de la marche** : Central Pattern Generators dans la moelle épinière
- Soft robotics

Motivations intrinsèques

Les tâches réalisées par le système sont le resultat indirect d'une "envie d'apprendre", "d'être surpris" et non des buts en tant que tel -> **curiosité artificielle** *(Kaplan & Oudeyer 2007)*

Applications et risques associés

Historiques... vieilles de 60 ans mais toujours un challenge

- **Voiture sans conducteur** : robotique autonome (*e.g.,* DARPA Grand Challenge)
- **Perception visuelle** (ILSVRC ImageNet, Sharp Eyes)
- **Traitement du langage naturel** (traduction/comprehension/conversation)
- **Découverte scientifique** (*e.g.*, identification d'exoplanètes inconnues)
- **Jeux** (echecs, go, jeux vidéos)

Plus recentes

- **Cybersécurité**: detection d'intrusions, de malwares, de SPAMs, reversing de programmes
- **Recommandation et Publicité**: profilage comportemental des usagers, (*e.g.*, réseaux sociaux)
- **Finance**: Hyper-trading
- **Médecine personnalisée**, pharmacogénétique (*e.g.*, Warfarin)
- **Criminalité**: prédiction des risques de récidive (COMPAS)

Nouveaux risques

- **Vie privée** : de toute méthode statistique qui révèle un fait à l'échelle d'une population fuit une certaine quantité d'information sur les participants au jeu d'entraînement
- **Opacité algorithmique** : un prédicteur n'a pas à expliquer sa décision
- **Vulnérabilité aux attaques** par pollution/évasion de modèles

De l'IA dans le programme de NSI

Classification K-plus proches voisins

Fonctionnement : Affecter à une entrée de classe inconnue la classe majoritaire parmi ses K plus proches voisins dans l'espace des entrée

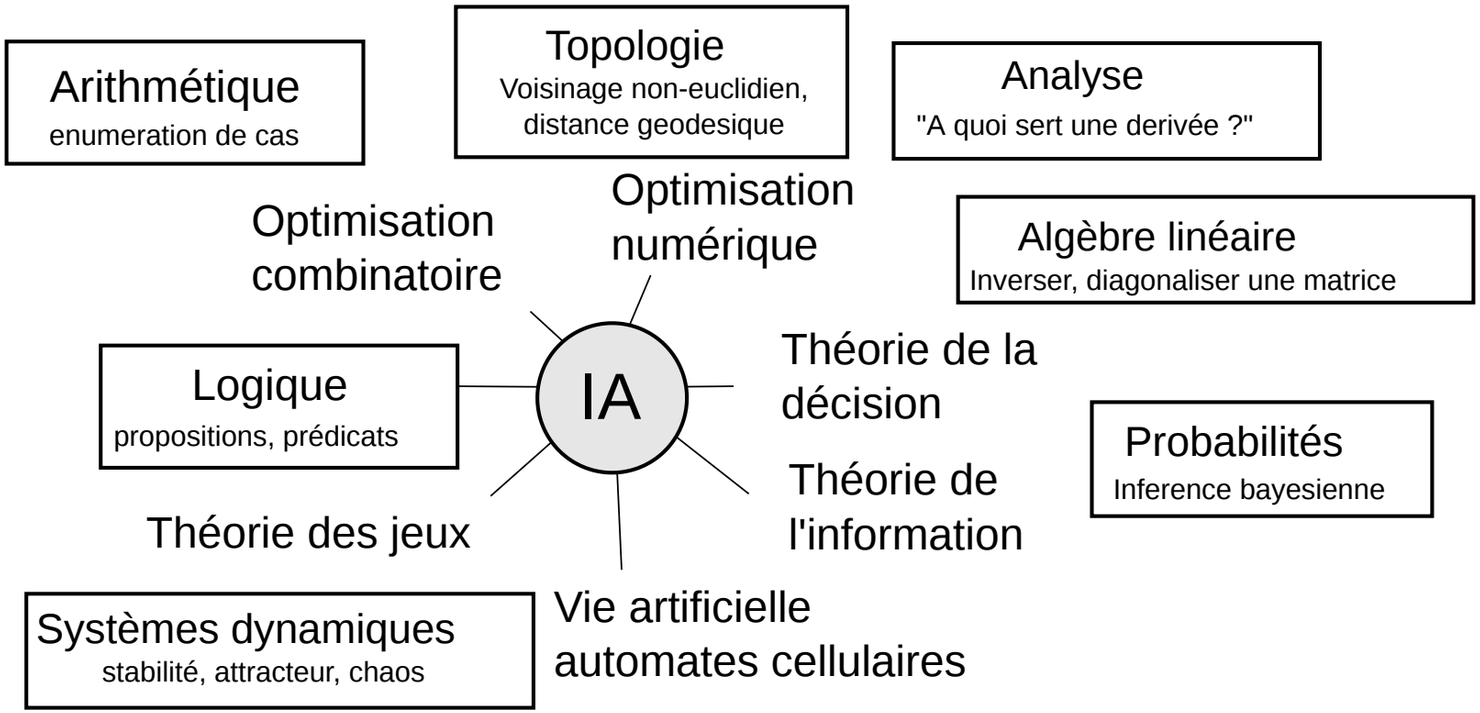
- Decision
- Métrique et EV
- Application : Système de Recommandation (collaborative filtering)

Parcours de graphe

- Parcours en profondeur
- Parcours en largeur
- Exploration des solutions d'un problème combinatoire
- Heuristiques de parcours -> algorithme A* (cf slide suivante)
- Detection de communauté, composantes connexes, distance géodésique

Pourquoi enseigner l'IA au lycée ?

- Susciter des vocations
  - **Nouveaux métiers** centrés sur l'apprentissage automatique
  - **Domaine de recherche** en pleine explosion, largement financé par l'état et les entreprises
- Initier des ponts avec les autres disciplines théoriques et experimentales
  - **Philosophie** : Phénoménologie de la perception (Merleau-Ponty), Enaction (Varela), Raisonnement (Aristote, Descartes), controverse Chomsky-Norvig sur la nature de la science et de l'intelligence (predire *vs* comprendre)
  - **Biologie** : Du neurone biologique au neurone formel, Modèle évolutionniste, developpement phylogénétique.
- Introduire/approfondir des **concepts mathématiques** de manière ludique pour réaffirmer le rôle des mathématiques comme **socle indispensable à toute science de la modélisation**.



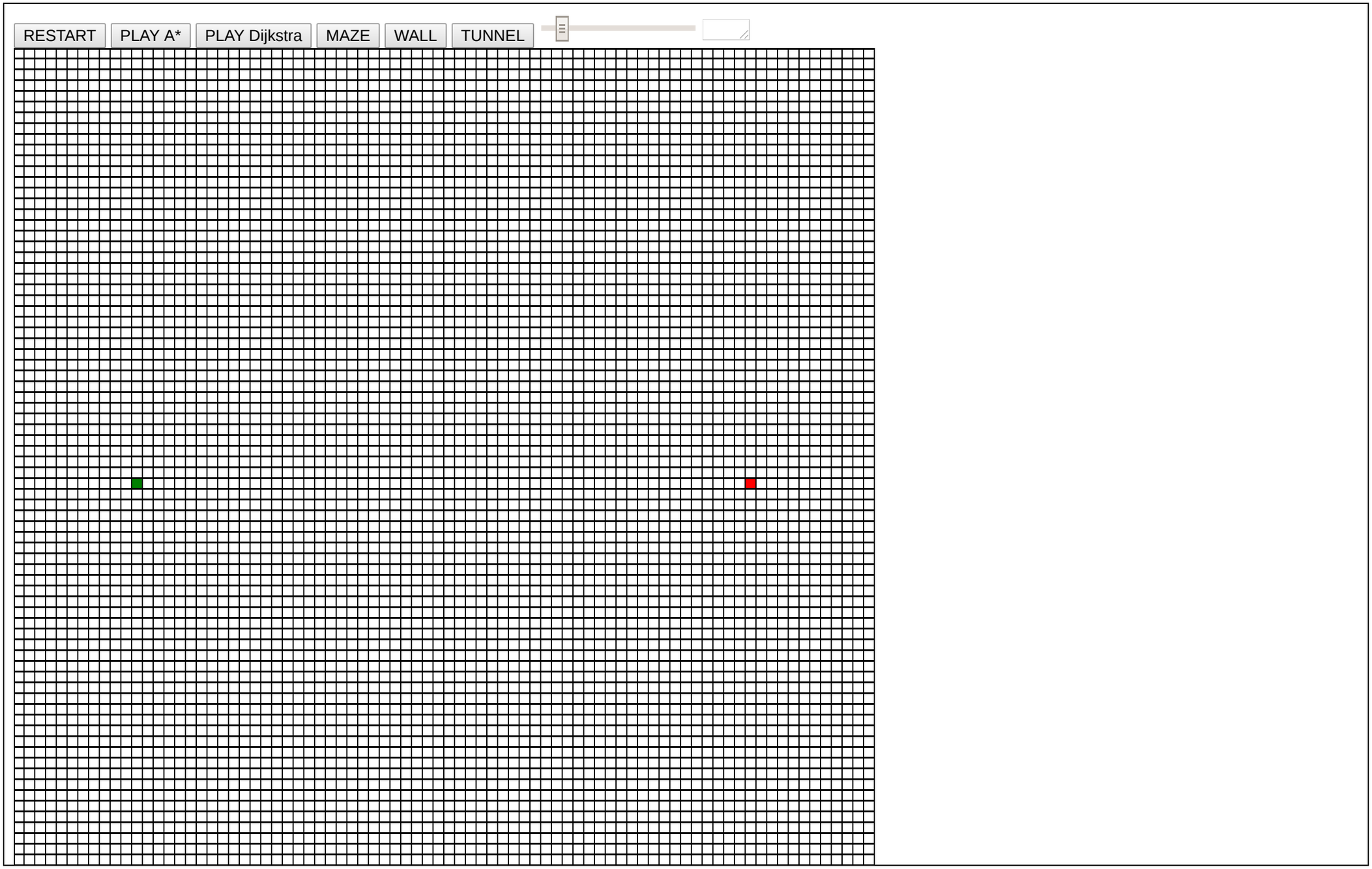
A*

Recherche du plus court chemin entre deux noeuds d'un graphe

- **Parcours en largeur** (DFS, cf programme NSI) -> utilisation d'une structure de type **file**
- **Algorithme de Dijkstra** (1959) -> arêtes pondérées -> **file de priorité**
- **A*** ("A star") -> estimation de la distance restant à parcourir via une **heuristique** $h(x, y)$

Algorithme A*

A* : Idée d'atelier à faire en classe



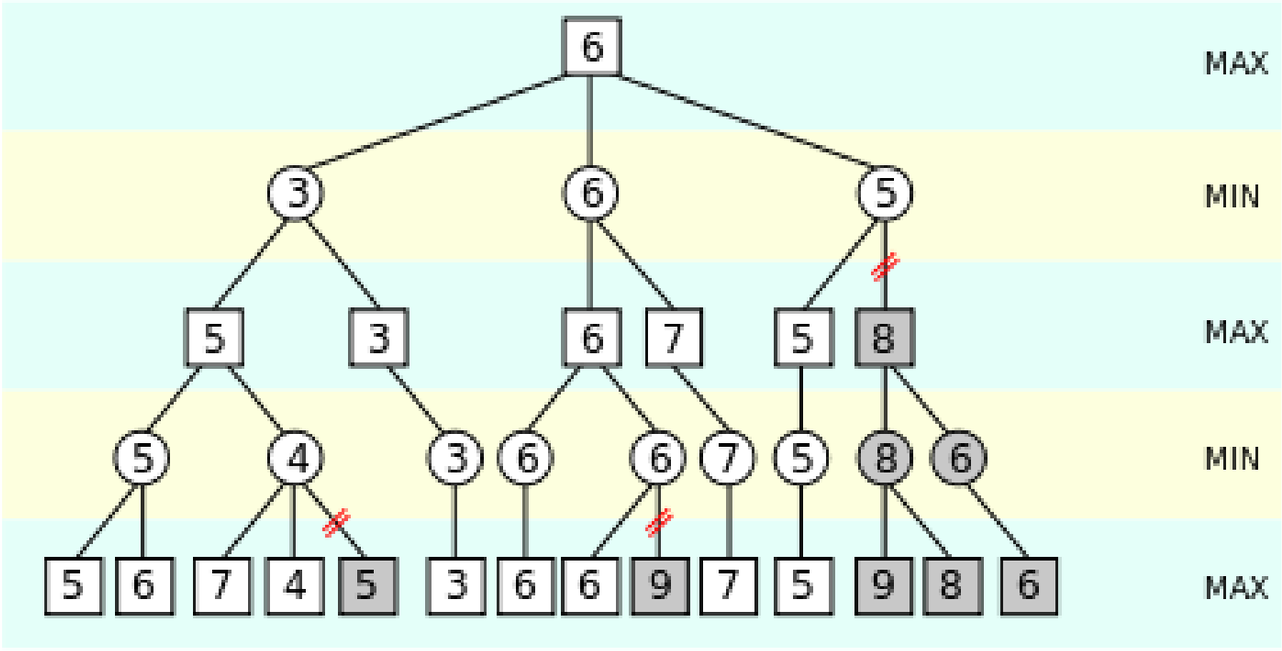
Mini-max

Théorie des jeux à somme nulle à 2 joueurs

- **Hypothèse :** l'adversaire cherche à maximiser ma perte (hypothèse du pire cas)
- **Stratégie MiniMax** = **minimiser la perte maximale**
- **Équilibre de Nash** *(1950)* : aucun joueur n'a interêt à modifier sa stratégie
- **Algorithme du Minimax**

1. Exploration en profondeur du graphe état-actions alterné entre les 2 joueurs (prof <= d)
2. Évaluation du gain des états feuilles
3. On remonte en maximisant quand c'est mon tour / minimisant quand c'est son tour

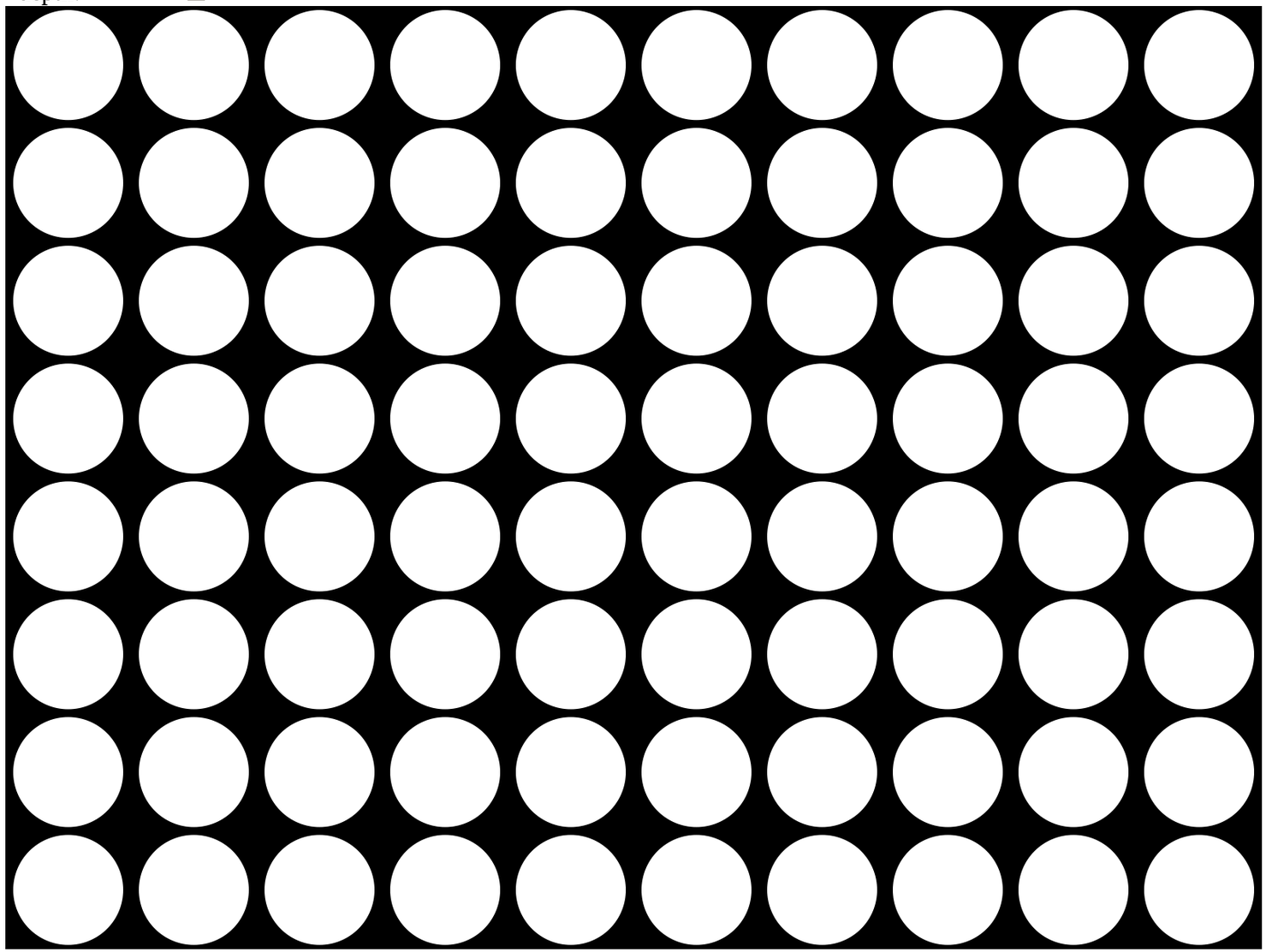
-> Extension : Elagage alpha-beta



Minimax

RESTART
PLAY ALPHA-BETA
PLAY MINIMAX

depth:

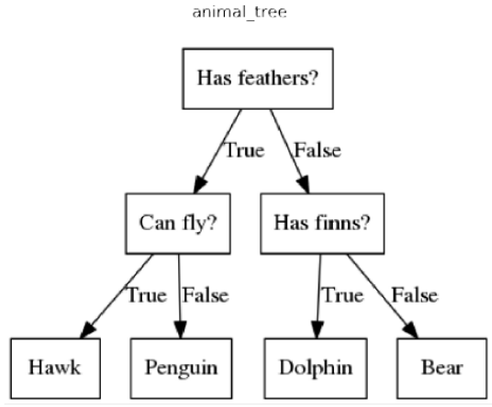


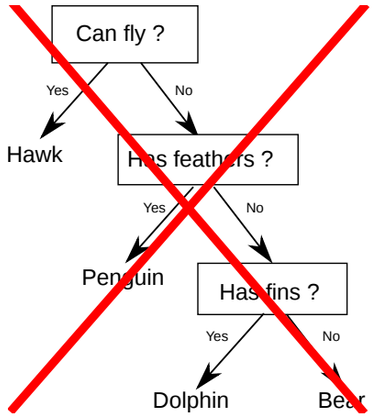
Arbres de décision

Jeu de données

attributs — classe

Feathers	Fly	Fins	Species
YES	YES	NO	Hawk
YES	NO	NO	Penguin
NO	NO	YES	Dolphin
NO	NO	NO	Bear





ID3 (Iterative Dichotomiser 3)

- Exemple typique de **raisonnement inductif**
- **Partitionnement recursif** du jeu de données
- A chaque étape on partitionne selon l'attribut qui apporte le **gain d'information maximal**

$$IG(S_1, \ldots, S_n) = H(\cup_i S_i) - \frac{1}{|\cup_i S_i|} \sum_i |S_i| H(S_i),$$

$H(S) = -\sum_x pS(x) \log_2 pS(x)$ H(S) est l'entropie de l'ensemble S

- On s'arrête quand chaque feuille ne contient plus qu'une seule classe ou qu'on ne peut plus obtenir de gain.

Idée d'activité à faire en classe

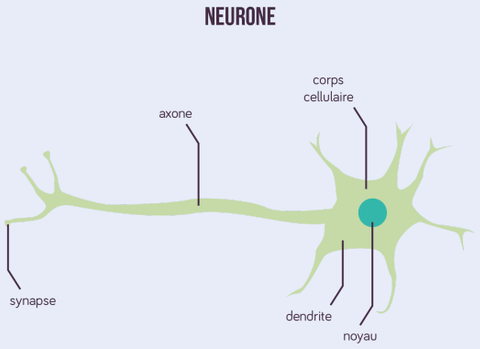
Former une base de données collaborative (anonymisée !) d'attributs des élèves (genre, couleur preferée, matière préférée, sport préféré, etc) et deviner le chanteur préféré d'un élève en posant le moins de questions possible <- Intro **décision** + **classif** + **théorie de l'information**

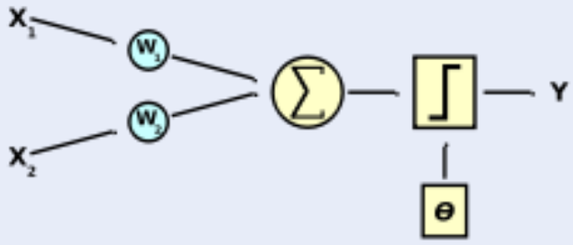
Réseaux de neurones
Optimisation par Descente de Gradient

Neurone formel *(McCulloch & Pitts 1943)*

- Inspiré des neurones biologiques
- K entrées, 1 sortie
- Poids synaptiques réglables (paramètres)
- Discriminant linéaire

$$f(\mathbf{w}, x) = \phi[\sum_i^K w_i \cdot x_i]$$





- ϕ : fonction d'activation $\mathbb{R} \mapsto \mathbb{R}$ (non-linéarité)
  Exemples : Heaviside (0 si x<0 et 1 si x>=0), Sigmoïde $\phi(x) = 1/(1+e^{-x})$, ...

Règle d'apprentissage : La Descente de Gradient

- **Fonction de coût** : exemple erreur quadratique $\mathcal{L}(\mathbf{w}, X, Y) = \sum_i^n (y_i - f(\mathbf{w}, x_i))^2$

1. On calcule le **gradient** de $\mathcal{L}$ par rapport aux **poids** : $\nabla\mathcal{L} = (\frac{\partial \mathcal{L}}{\partial w_1}, \ldots, \frac{\partial \mathcal{L}}{\partial w_K})$
2. On **déplace les poids** dans la direction négative du gradient: $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta\nabla\mathcal{L}$

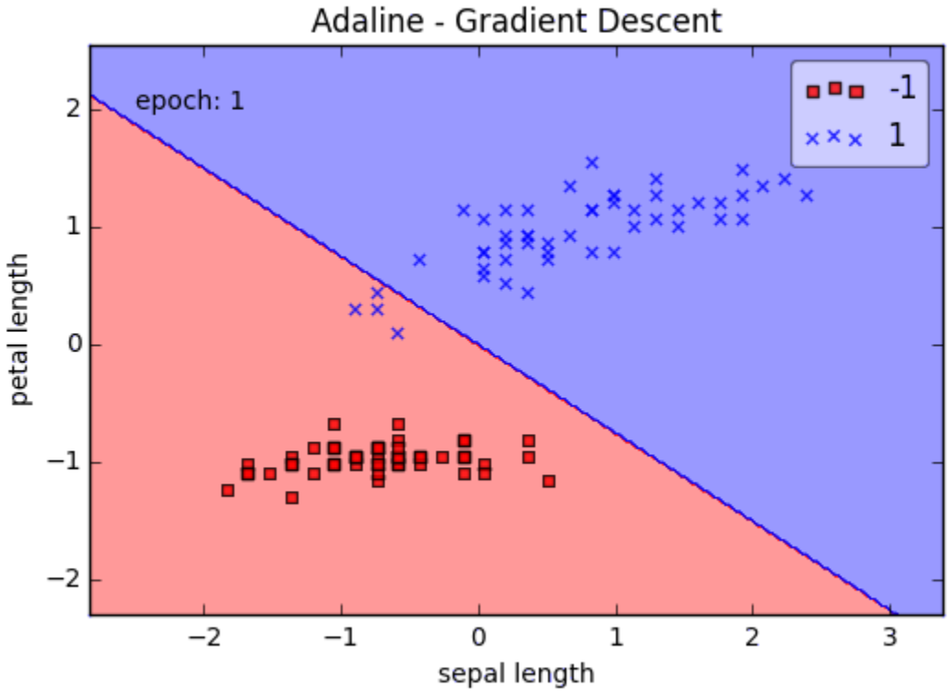
(∇ = pas d'apprentissage) -> puis on recommence.

- **Descente de Gradient Stochastique** (SGD) : on calcule le gradient sur un seul exemple tiré au hasard (ou un "mini-batch" de quelques exemples) -> étapes plus rapides, mais pb stabilité

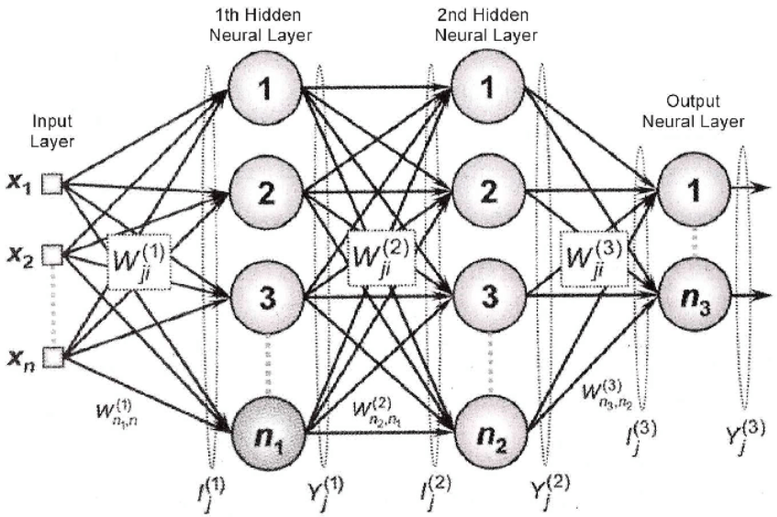
Réseaux de neurones
Optimisation par Descente de Gradient

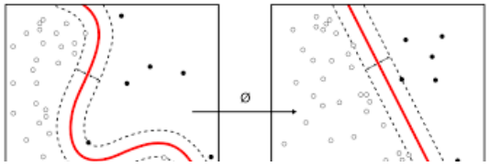
Perceptron *(Rosenblatt 1958)*

- Discriminant linéaire $f(\mathbf{w}, x) = H[\sum_i^K w_i \cdot x_i + b]$
- Calcul du gradient très simple : $\mathcal{L}(\mathbf{w}, X, Y) = \sum_i^n (y_i - H[\sum_i^K w_i \cdot x_i + b])$
  => $\nabla\mathcal{L} = \sum_i^n [(y_i - f(\mathbf{w}, x_i)) \cdot x_i]$
- On ne met à jour qu'en cas d'erreur de prédiction -> on ajoute une portion de l'entrée au poids pour réduire l'erreur
- Extension : **Perceptron multi-couche** (MLP) -> permet la classification non-linéaire
  Règle de chainage des dérivées pour les fonctions composées => **Retropropagation du gradient**



Perceptron Multi-couche

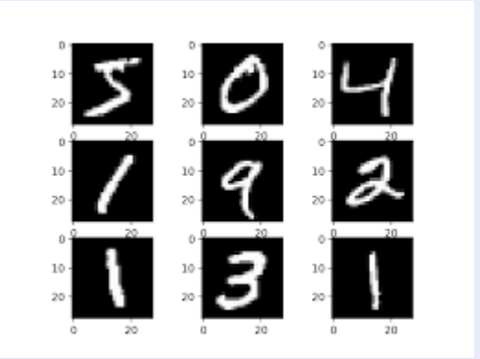




Classification d'images par réseaux profonds

MNIST *(LeCun 1998)*

70000 images de chiffres manuscrits, (60000 entraînement, 10000 test)
28x28px, noir et blanc => $x \in \mathbb{R}^{784}$, $y \in [0..9]$



But : optimiser (entraîner) un modèle sur les 60000 exemples, puis classifier les 10000 exemples par chiffre représenté
Stratégie possible : apprendre un perceptron par chiffre (one-vs-all)
Score : précision moyenne $\frac{1}{N}\sum_i \mathbf{1}[y_i = f(x_i)]$ par exemple

Idée d'activité à faire en classe
Comparer les performances / capacités de généralisation de diverses architectures

- Classificateur KPPV (cf programme NSI)
- Perceptron avant apprentissage (poids aléatoires)
- Perceptron entraîné par Descente de Gradient
- Perceptron multicouche (voire un réseau profond, Deep Network)
  => gradient et couches déjà implémentées dans le toolkit python Keras -> clé en main
  https://keras.io/examples/mnist_cnn/
- Tâche plutôt simple et peu coûteuse, pas besoin de GPU

Prolog

http://www.epi.asso.fr/revue/articles/a1812b.htm

http://www.swi-prolog.org/pldoc/man?section=clpfd-sudoku

Exemple : resolution du sudoku en quelques lignes de Prolog

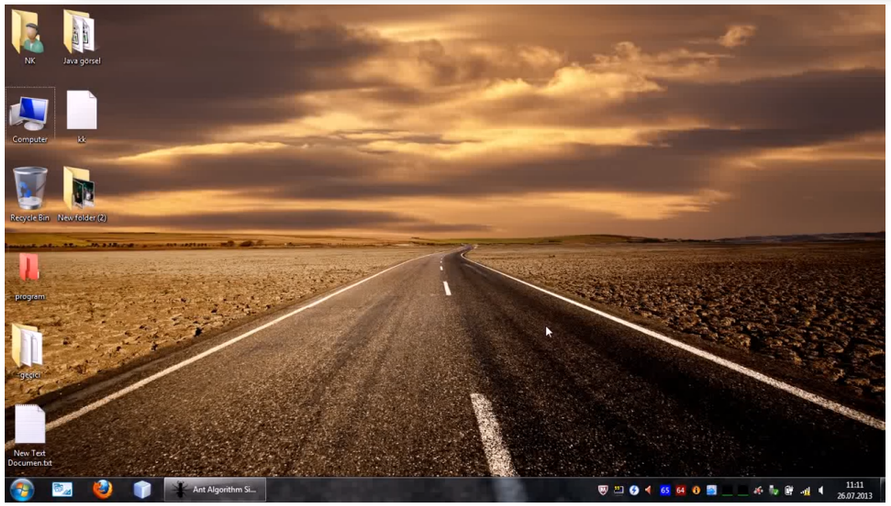
```
sudoku(Rows) :-
        length(Rows, 9), maplist(same_length(Rows), Rows),
        append(Rows, Vs), Vs ins 1..9,
        maplist(all_distinct, Rows),
        transpose(Rows, Columns),
        maplist(all_distinct, Columns),
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
        blocks(As, Bs, Cs), blocks(Ds, Es, Fs), blocks(Gs, Hs, Is).

blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
        all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
        blocks(Ns1, Ns2, Ns3).

problem(1, [[_,_,_,_,_,_,_,_,_],
            [_,_,_,_,_,3,_,8,5],
            [_,_,1,_,2,_,_,_,_],
            [_,_,_,5,_,7,_,_,_],
            [_,_,4,_,_,_,1,_,_],
            [_,9,_,_,_,_,_,_,_],
            [5,_,_,_,_,_,_,7,3],
            [_,_,2,_,1,_,_,_,_],
            [_,_,_,_,4,_,_,_,9]]).
```

Intelligence collective : Optimisation par Colonie de Fourmies (ACO)

Stigmergie *(Grassé 1959)*

Mécanisme de **coordination indirecte** entre agents (inspiré des **insectes eusociaux**, e.g., fourmis)

- Un agent qui reçoit une **stimulation** (recompense) dépose des **phéromones** sur son chemin
- Les autres agents sont **attirés** par les traces de phéromones en direction de la **récompense**
- Les phéromones sont **volatiles** -> décroissance exponentielle ~ distance au but
- La colonie **s'auto-organise** sans supervision pour optimiser l'accès à la récompense de manière **dynamique**
- **Principe du tableau noir** : l'environnement joue le rôle de **mémoire** et de **médiateur** de la récompense



Apprentissage par renforcement : Q-learning

Principe

- Un **agent**, un **environnement**. A chaque instant, les **actions** de l'agent modifient **l'état** de l'environnement
- Pour chaque état, chaque action produit une **récompense** (positive, négative, ou nulle)
- La fonction de récompense est **non-dérivable** -> descente de gradient impossible !
- **Algorithme Q-learning**. Par exploration aléatoire, on apprend à predire l'action qui donne la meilleure **recompense future (Q-value)** pour chaque couple *(etat,action)*, en propagant la valeur maximale le long du graphe état-action

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

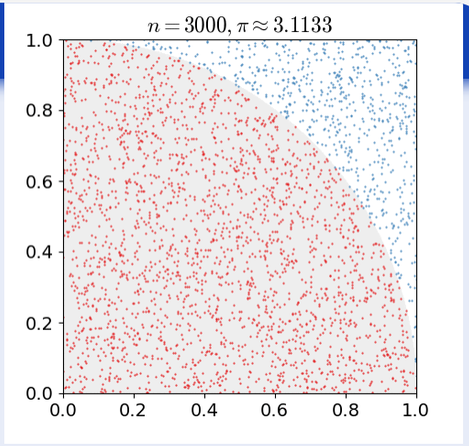
Algorithmes évolutionnaires / génétiques

Cas d'usage

Problème de **recherche d'optimum** lorsque la fonction de coût est **non-linéaire** et **non-dérivable**

Méthode Monte-Carlo

- On instancie n entrées aléatoires
- On évalue le coût de ces n entrées
- On agrège le résultat (max, moyenne ...)



Algorithme génétique

1. On instancie une population initiale de n entrées aléatoires
2. On évalue le coût (fitness) de ces n entrées
3. On "tue" les n entrées puis on recrée une nouvelle "génération" par
   - Combinaison des p meilleurs individus de la génération précédente (-> **croisement**)
   - Légères modifications aléatoires (-> **mutation**)
   - Les $n - p$ individus de la génération précédente disparaissent (**selection naturelle**)
4. On retourne à l'étape 2

Bibliographie

Blogs :

- Perceptron et réseaux de neurones
  : https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html
- Cours video en ligne de Hugo Larochelle :
  https://www.youtube.com/playlist?list=PL6Xpj9I5qXYGhsvMWM53ZLfwUInzvYWsm
  https://www.youtube.com/playlist?list=PL6Xpj9I5qXYFD_rc1tttugXLfE2TcKyiO

Livres :

- Introduction to Machine Learning with Python
- Pattern Recognition and Machine Learning

Presentations :

- Metric learning : http://researchers.lille.inria.fr/abellet/talks/metric_learning_tutorial_CIL.pdf
- ID3 & Arbres de décision : https://www.slideshare.net/abood85/id3c45-algorithim

et surtout Wikipedia, excellente ressource pour toutes les techniques d'IA et d'apprentissage !