Université Mohammed Khider-BISKRA

Faculté des Sciences Exactes et la nature et de la vie

Département d'informatique

Cours de Master2
Image et vie artificielle

Module :

Vie Artificielle

Pr. CHERIF Foudil

Les systèmes d'apprentissage

Les systèmes d'Apprentissage permettent à des individus de s'adapter à un environnement par un apprentissage sans en modifier sa structure.

Les systèmes d'apprentissage

- les deux principales techniques développées :

Les ***réseaux de neurones*** et les ***systèmes de classeurs***

Les réseaux de neurones

- Inspiré du fonctionnement du cerveau, c'est en 1943 avec Mc Culloch et Pitts qu'une première recherche est faite pour modéliser mathématiquement ces fonctions.
- Aujourd'hui ces recherches sur les "réseaux de neurones artificiels" ont beaucoup progressées.

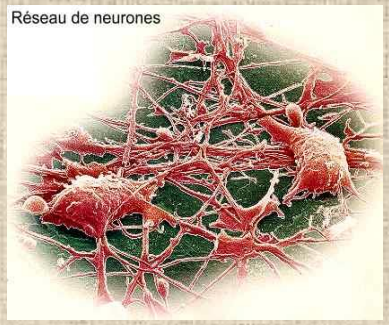
Les systèmes de classifieurs

furent introduits par John Holland en 1978 comme une architecture utilisant les algorithmes génétiques afin d'étudier l'apprentissage des systèmes à bases de règles du type condition/action.

Les réseaux de neurones

Les réseaux de neurones

Réseaux de neurones formels



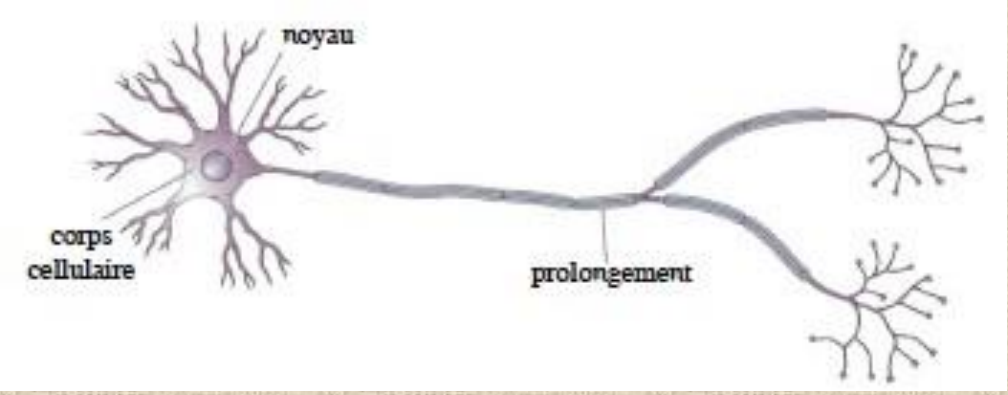
Les **réseaux de neurones formels** sont à l'origine une tentative de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'oeuvre de MM. *Mac Culloch et Pitts*. Ils présentent un modèle assez simple pour les neurones et explorent les possibilités de ce modèle.

L'idée principale des réseaux de neurones "modernes" est la suivante :

On se donne une unité simple, un *neurone*, qui est capable de réaliser quelques calculs élémentaires. On relie ensuite entre elles un nombre important de ces unités et on essaye de déterminer la puissance de calcul du *réseau* ainsi obtenu. Il est important de noter que ces neurones manipulent des données **numériques** et non pas symboliques

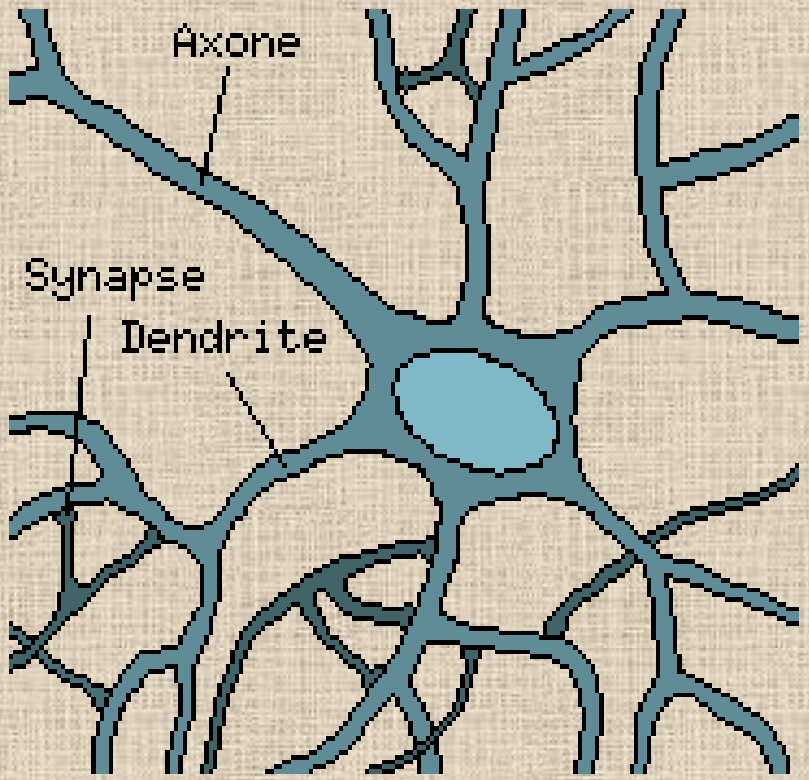
Le cerveau

- L'ensemble du tissu nerveux est composé par les cellules nerveuses, qu'on appelle aussi neurones.
- Les neurones sont considérés comme le support physique de l'intelligence.
- Le neurone est une cellule dédié à la transmission de l'influx nerveux
- Pour cela, la cellule est munie d'un prolongement capable de transporter le signal sur une plus ou moins longue distance.
- Le système nerveux central correspond à ce que l'on appelle cerveau.
- Le système nerveux périphérique est constitué des nerfs qui traversent le corps.



Un modèle biologique

- Dans le cerveau, les neurones sont reliés entre eux par l'intermédiaires **d'axones** et de **dendrites**. En première approche. on peut considérer que ces sortes de filaments sont conducteurs d'électricité et peuvent ainsi véhiculer des messages depuis un neurone vers un autre. Les dendrites représentent les entrées du neurone et son axone sa sortie.

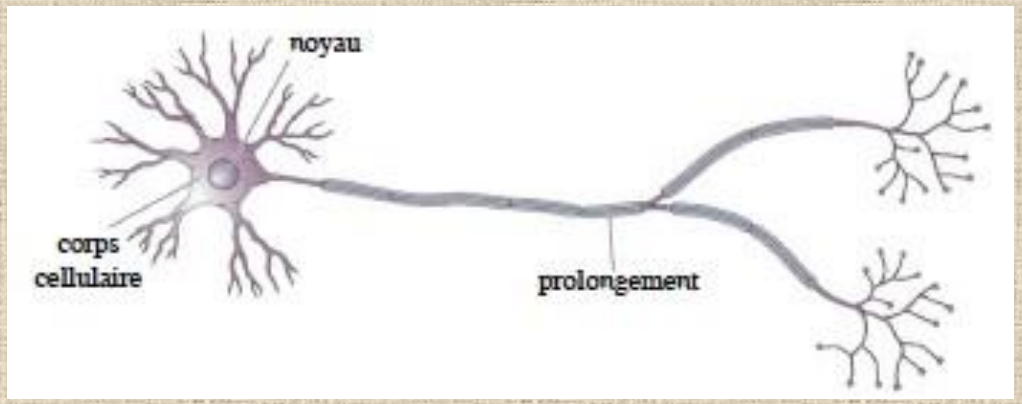


Fondements biologiques

Le système nerveux est composé de 1012 neurones interconnectés. Bien qu'il existe une grande diversité de neurones, ils fonctionnent tous sur le même schéma.

Ils se décomposent en trois régions principales :

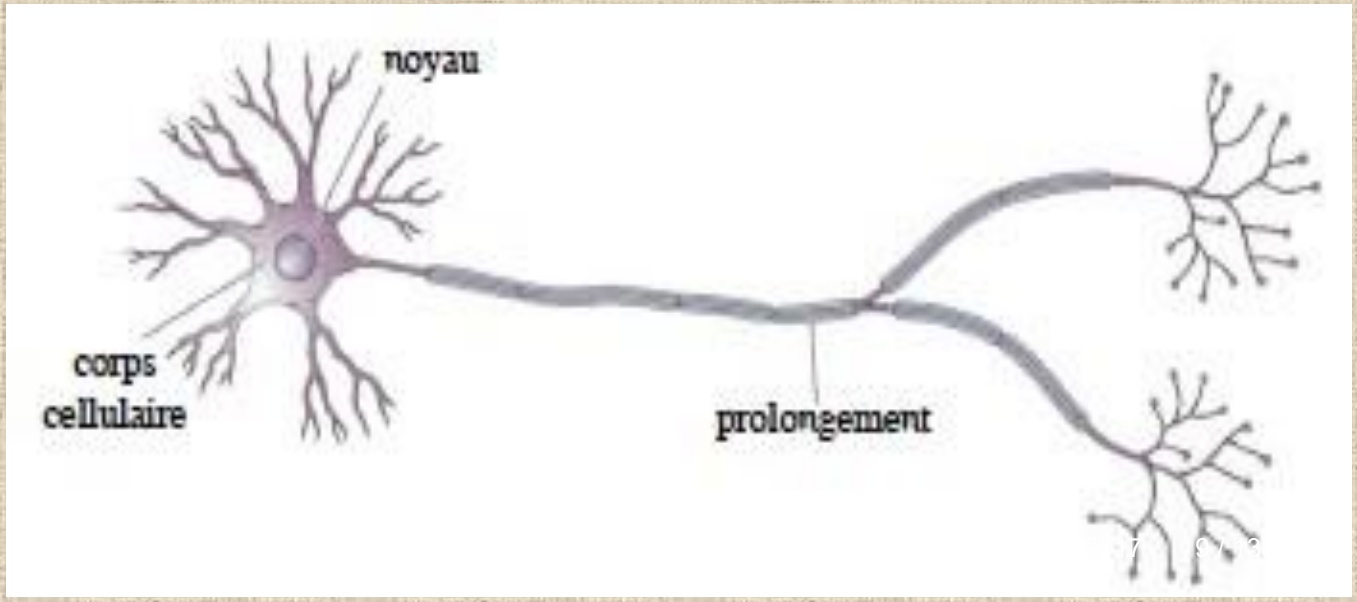
1-Le corps cellulaire 2-Les dendrites 3-L'axone



Fondements biologiques

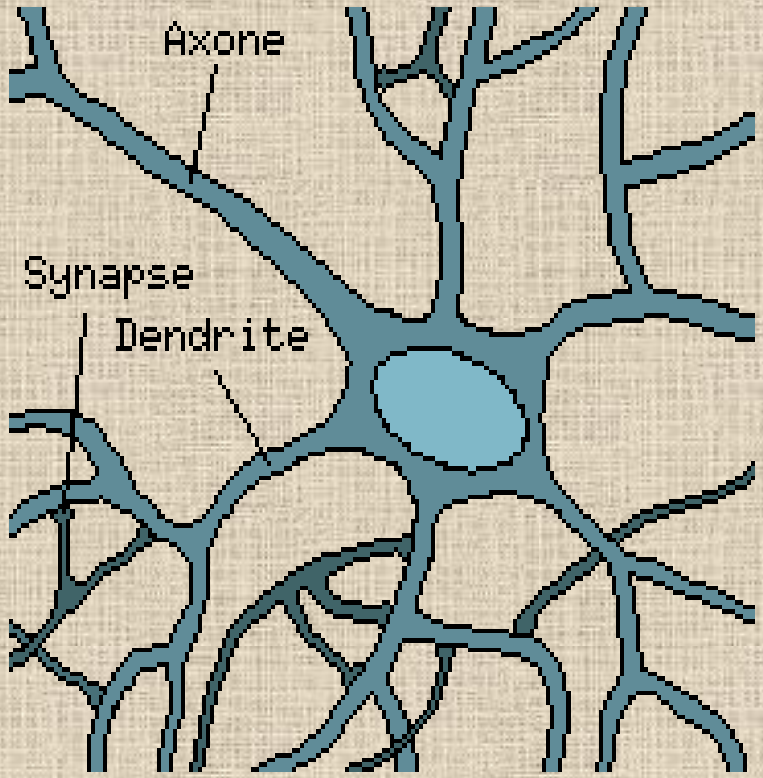
L'influx nerveux est assimilable à un signal électrique se propageant comme suit:

- Les dendrites reçoivent l'influx nerveux d'autres neurones.
- Le neurone évalue l'ensemble de la stimulation reçue.
- Si elle est suffisante, il est exicité : il transmet un signal (0/1) le long de l'axone.
- L'excitation est propagée jusqu'aux autres neurones qui y sont connectés via les synapses.



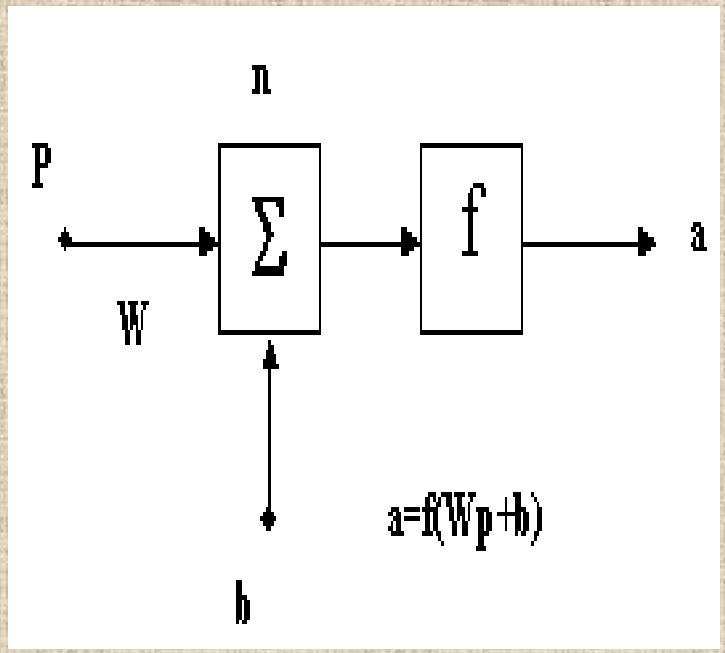
Un modèle biologique

- Un neurone émet un signal en fonction des signaux qui lui proviennent des autres neurones. On observe en fait au niveau d'un neurone, une *intégration* des signaux reçus au cours du temps, c'est à dire une sorte de sommations des signaux. En général, quand la somme dépasse un certain seuil, le neurone émet à son tour un signal électrique



Un neurone formel

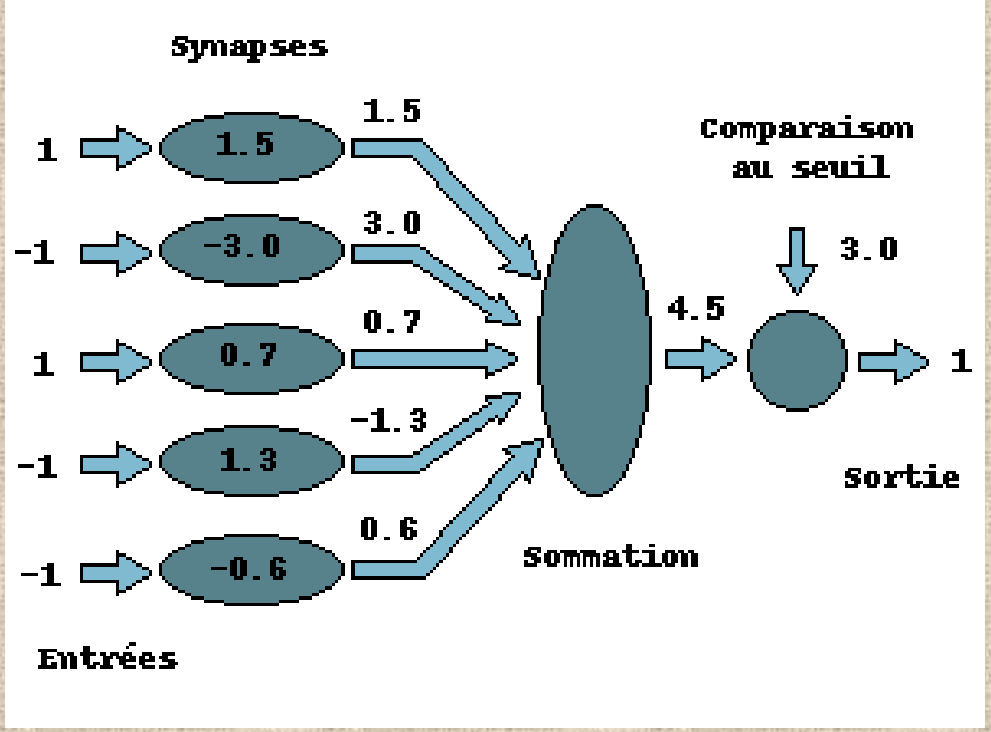
- Le modèle de neurone formel présenté ici, du à Mac Culloch et Pitts, est un modèle mathématique très simple dérivé d'une analyse de la réalité biologique.
- On constate tout d'abord que le modèle biologique fait intervenir une notion temporelle qui est difficile à intégrer dans un modèle simple. On oublie donc cette notion et de ce fait on remplace l'intégration temporelle par une simple sommation des signaux arrivant au neurone (ces signaux sont communément appelés les *entrées* du neurones).



Neurone Formel

Principes :

- Pas de notion temporelle (remplacé par une sommation)
- Coefficient synaptique : coefficient réel
- Sommation des signaux arrivant au neurone
- Sortie obtenue après application d'une fonction de transfert

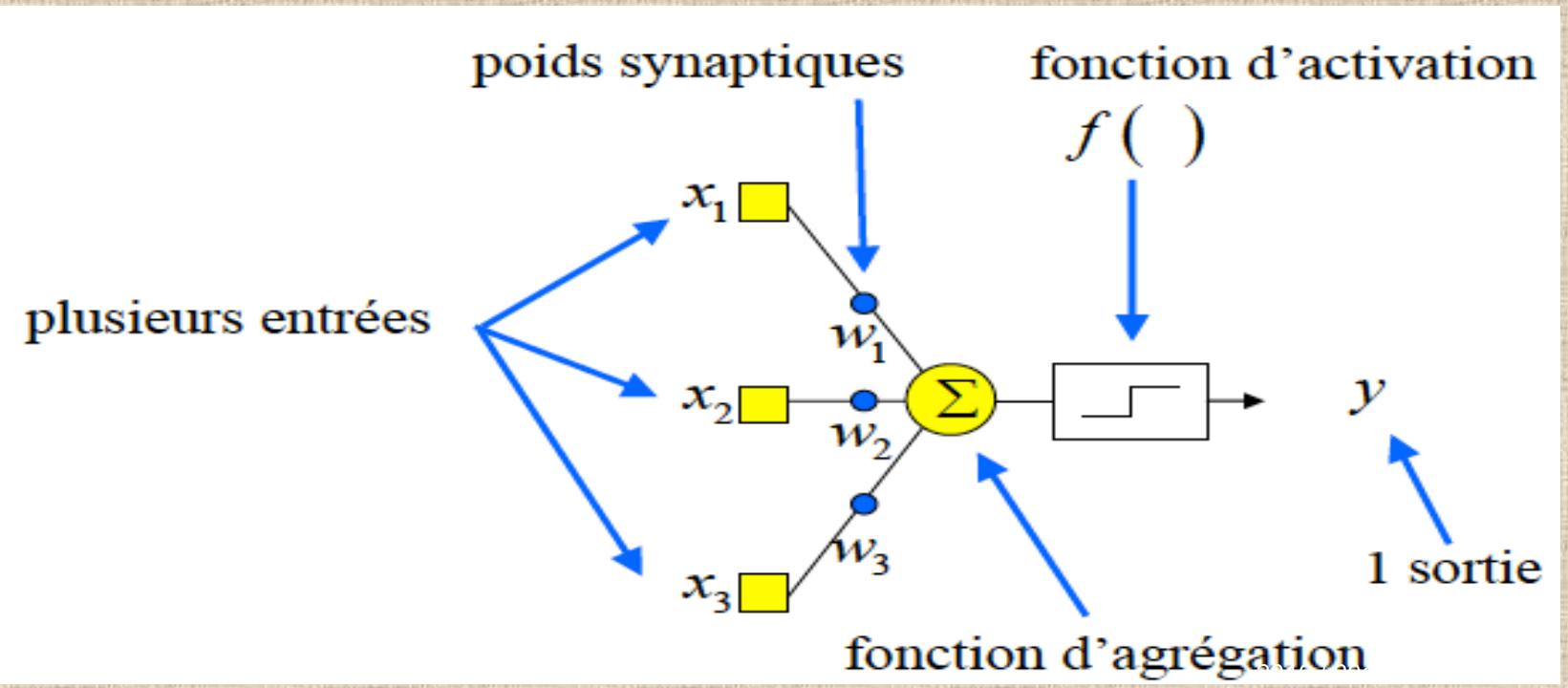


Neurone Formel

Principes :

Le neurone formel : exemple avec 3 entrées

Un neurone est une cellule qui "concentre" des signaux, qui les traite et qui les traduit en une réponse.



Neurone Formel (historique)

1930, Nicolas Rashevski propose une modélisation mathématique du fonctionnement du système nerveux basé sur la minimisation de l'énergie

En 1943, le neurone formel de Warren McCulloch et Walter Pitts fait son apparition, c'est un modèle mathématique très simple dérivé d'une analyse de la réalité biologique.

En **1949**, D. Hebb présente dans son ouvrage "The Organization of Behavior" une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb.

En **1958**, F. Rosenblatt développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche lié à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience

Neurone Formel (historique)

En 1969, M. Minsky et S. Papert publient une critique des propriétés du Perceptron. Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où T. Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.

C'est en **1982** que J. Hopfield présente son étude d'un réseau complétement rebouclé, dont il analyse la dynamique.

Aujourd'hui, les réseaux neuronaux sont utilisés dans de nombreux domaines (entre autres, vie artificielle et intelligence artificielle) à cause de leur propriété en particulier, leur capacité d'apprentissage, et qu'ils soient des systèmes dynamiques.

Ordinateur vs(contre) cerveau

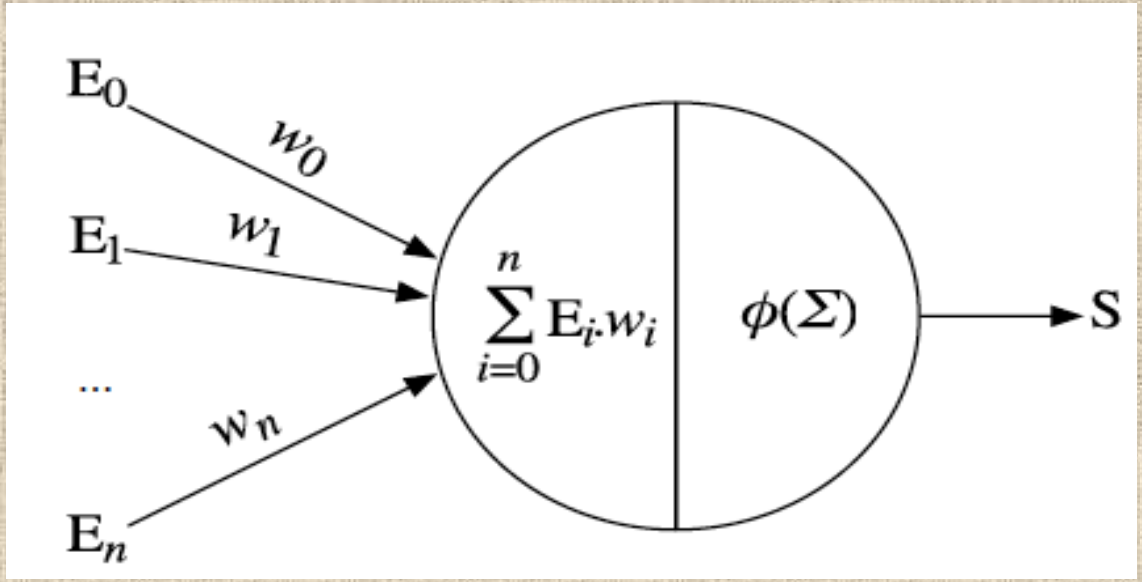
- Ordinateur de Von Neumann's
  - déterministe
  - séquences d'instructions
  - tâches répétitives
  - vitesse élevée
  - programmation
  - unicité des solutions
- ex: produit matriciel

- Cerveau (humain)
  - parallélisme
  - adaptation (apprentissage)
  - vitesse lente
  - comportement flou
  - différentes façons d'aborder un problème
  - différentes solutions
- ex: reconnaissance de visages

Il se révèle extrêmement difficile de construire un ordinateur qui imite réellement un cerveau

Neurone Formel (caracteristiques)

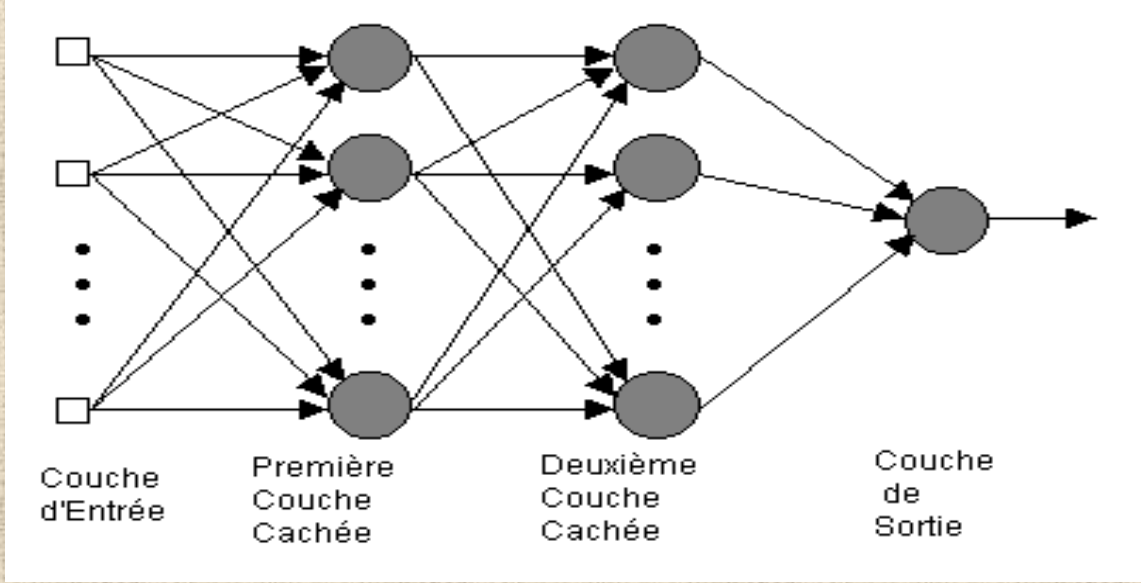
- La capacité d'apprentissage
- La capacité de mémoriser l'information
- La capacité de traiter des informations incomplètes



Réseau de neurones

- L'idée principale est de grouper des neurones dans une *couche*. On place ensuite bout à bout plusieurs couches et on connecte complètement les neurones de deux couches adjacentes. Les entrées des neurones de la deuxième couche sont donc en fait les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée (c'est en fait l'entrée du réseau). Ils calculent alors leur sorties qui sont transmises aux neurones de la deuxième couche, etc. Les sorties des neurones de la dernière couche forment la sortie du réseau.

Neurones en reseau →



Apprentissage

- Une fonction des réseaux de neurones formels, à l'instar du modèle vivant, est d'opérer rapidement des classifications et d'apprendre à les améliorer.
- À l'opposé des méthodes traditionnelles de résolution informatique, on ne doit pas construire un programme pas à pas en fonction de la compréhension du programme.
- Les paramètres importants de ce modèle sont les coefficients synaptiques et le seuil de chaque neurone, et la façon de les ajuster. Ce sont eux qui déterminent l'évolution du réseau en fonction de ses informations d'entrées.

Apprentissage

- Il faut choisir un mécanisme permettant de les calculer et de les faire converger si possible vers une valeur assurant une classification aussi proche que possible de l'optimale. C'est ce qu'on nomme **la phase d'apprentissage du réseau**.
- Dans un modèle de réseaux de neurones formels, apprendre revient donc à déterminer les **coefficients synaptiques** les plus adaptés à classifier les exemples présentés.
- Les réseaux de neurones, en tant que systèmes capables d'apprendre, mettent en œuvre le principe de l'induction, c'est-à-dire l'apprentissage **par l'expérience**.
- Les réseaux de neurones artificiels ont besoin **de cas réels** servant d'exemples pour leur apprentissage (on appelle cela la base d'apprentissage). Ces cas doivent être d'autant **plus nombreux** que le problème est complexe et que sa topologie est peu structurée.

Definition de l'apprentissage

- Déterminer un réseau de neurones = Trouver les coefficients synaptiques.

- On parle de phase **d'apprentissage** : les caractéristiques du réseau sont modifiées jusqu'à ce que le comportement désiré soit obtenu.

- Base d'apprentissage : exemples représentatifs du comportement ou de la fonction à modéliser. Ces exemples sont sous la forme de couples (entrée ; sortie) connus.

- Base d'essai : pour une entrée quelconque (bruitée ou incomplète), calculer la sortie. On peut alors évaluer la performance du réseau.

Définition de l'apprentissage

Apprentissage supervisé : les coefficients synaptiques sont évalués en minimisant l'erreur (entre sortie souhaitée et sortie obtenue) sur une base d'apprentissage.

Apprentissage non-supervisé : on ne dispose pas de base d'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales.

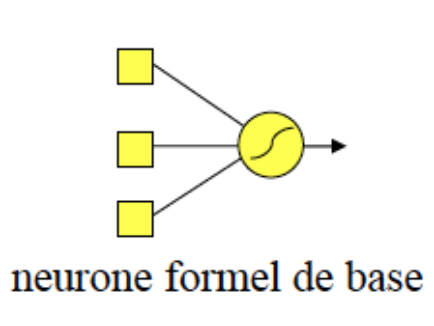
Sur-apprentissage : on minimise l'erreur sur la base d'apprentissage à chaque itération mais on augmente l'erreur sur la base d'essai. Le modèle prend sa capacité de généralisation : c'est **l'apprentissage par cœur**.

- Il arrive souvent que les exemples de la base d'apprentissage comportent des valeurs **approximatives ou bruitées**. Si on oblige le réseau à répondre de façon quasi parfaite relativement à ces exemples, on peut obtenir un réseau qui est **biaisé par des valeurs erronées**.

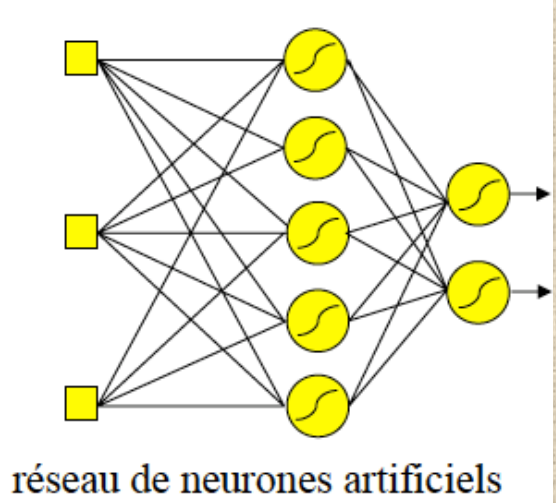
Pour éviter le surapprentissage, il existe une méthode simple : il suffit de partager la base d'exemples en 2 sous-ensembles. Le premier sert à **l'apprentissage** et le second sert à **l'évaluation de l'apprentissage**.

Assembler les neurones

Objectif : créer des méthodes adaptatives d'identification et des lois de commandes intelligentes, basées sur le principe de l'apprentissage, donc sur des réseaux de neurones artificiels. Assembler des neurones c'est multiplier leurs capacités d'apprentissage.



neurone formel de base



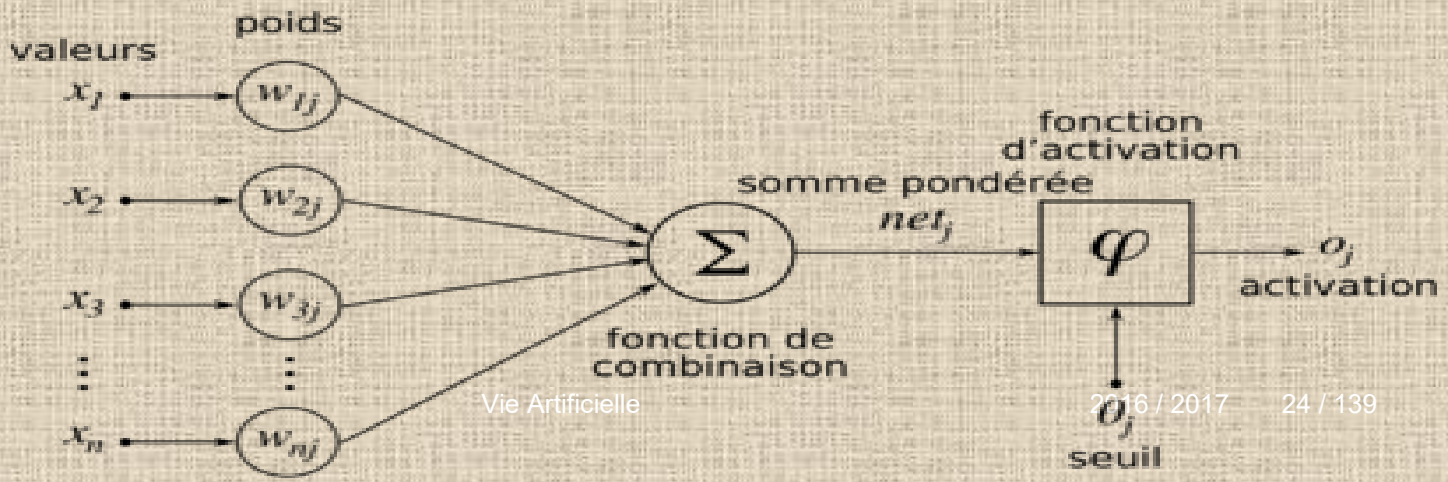
réseau de neurones artificiels

Structure d’un réseau

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de Ni neurones, prenant leurs entrées sur les Ni-1 neurones de la couche précédente.

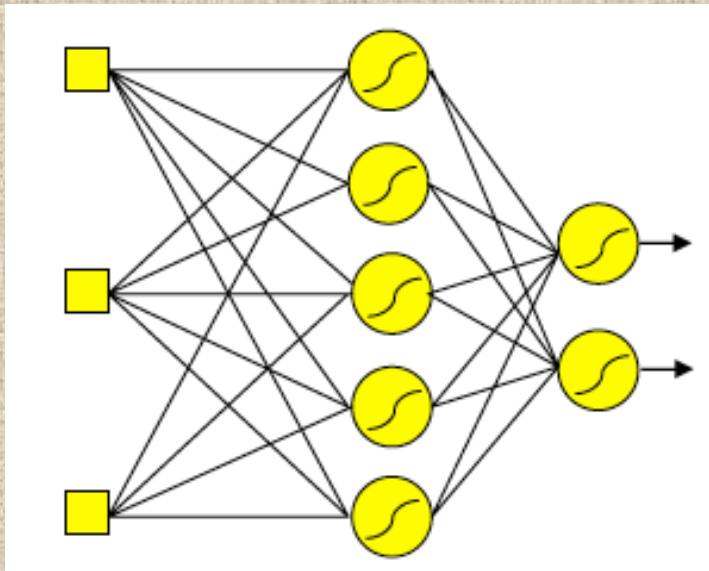
l'importance du choix judicieux d'une bonne fonction de sortie : un réseau de neurones dont les sorties seraient linéaires n'aurait aucun intérêt.

Au-delà de cette structure simple, le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. De la même façon que des boucles peuvent transformer une logique combinatoire en logique séquentielle, les boucles dans un réseau de neurones transforment un simple dispositif de reconnaissance d'entrées en une machine complexe capable de toutes sortes de comportements.



Assembler les neurones

Exemple d'un réseau multicouche complètement connecté



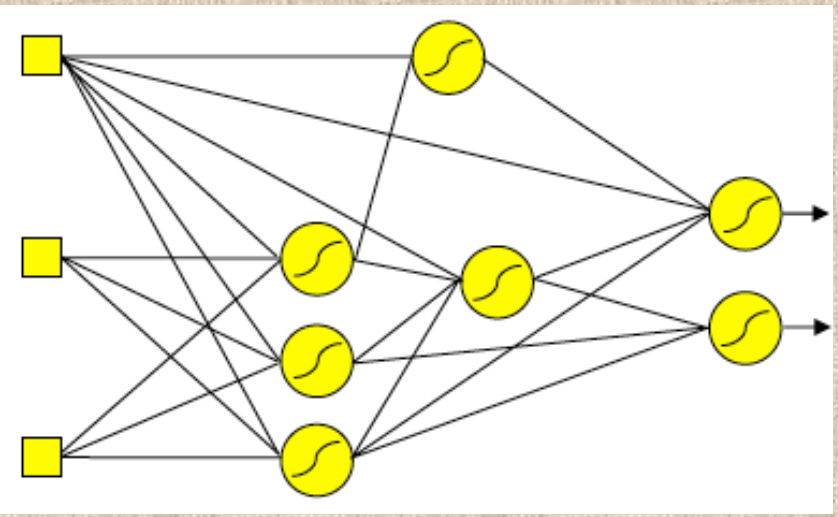
- Le réseau suivant est constitué de 5 neurones (3 entrées).
- Ce réseau est bien ordonné. On y distingue des couches et les neurones sont entièrement connectés entre les couches.

Fonction d'activation

La **fonction d'activation** (ou **fonction de seuillage**, ou encore **fonction de transfert**) sert à introduire une non-linéarité dans le fonctionnement du neurone.
Les fonctions de seuillage présentent généralement trois intervalles :
1.en dessous du seuil, le neurone est non-actif (souvent dans ce cas, sa sortie vaut 0 ou -1) ;
2.aux alentours du seuil, une phase de transition ;
3.au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1).
Des exemples classiques de fonctions d'activation sont :
1.La fonction **sigmoïde**.
2.La fonction tangente hyperbolique.
3.La fonction de Heaviside.

Assembler les neurones

Exemple d'un réseau non ordonné et partiellement connecté



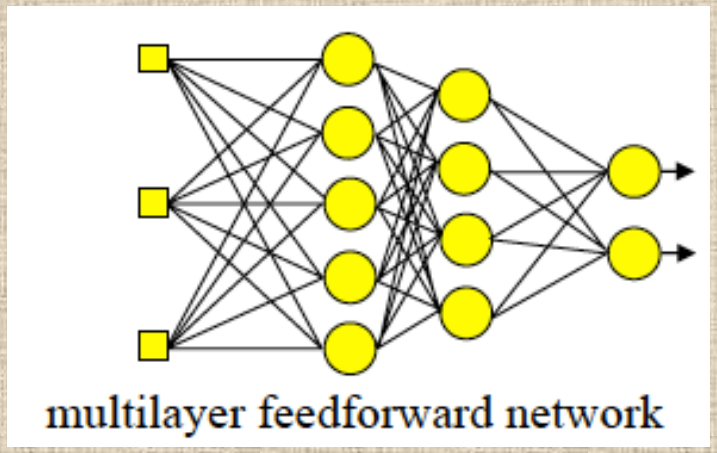
- Il est plus simple de mettre des neurones ensemble, mais il est plus difficile de les faire apprendre.
- Les architectures généralisées ou ordonnées sont plus simple à "manipuler".

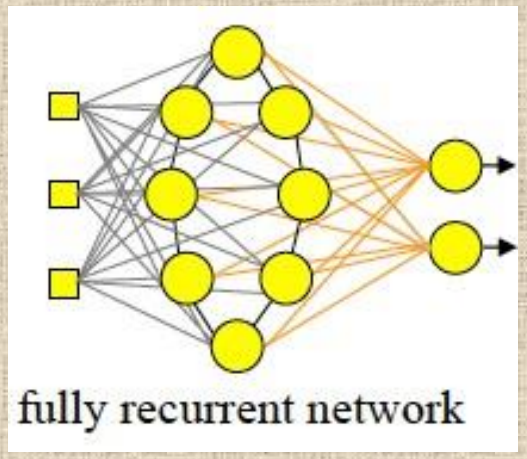
Différentes architectures neuronales

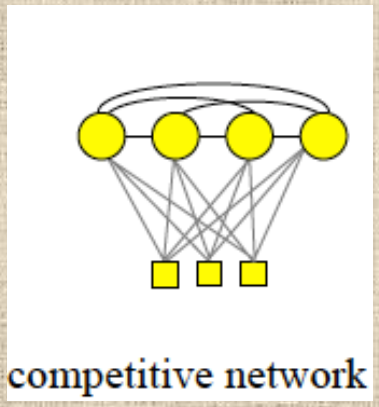
- L'architecture d'un réseau de neurone est l'organisation des neurones entre eux au sein d'un même réseau. Autrement dit, il s'agit de la façon dont ils sont ordonnées et connectés.
- La majorité des réseaux de neurones utilise le même type de neurones. Quelques architectures plus rares se basent sur des neurones dédiés. L'architecture d'un réseau de neurones dépond de la tache à apprendre (problème à résoudre).
- Un réseau de neurones est en général composé de plusieurs couches de neurones, des entrées jusqu'aux sorties

Différentes architectures neuronales

Quelques architectures :

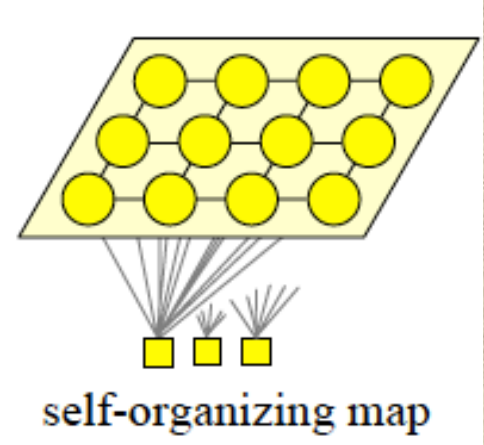


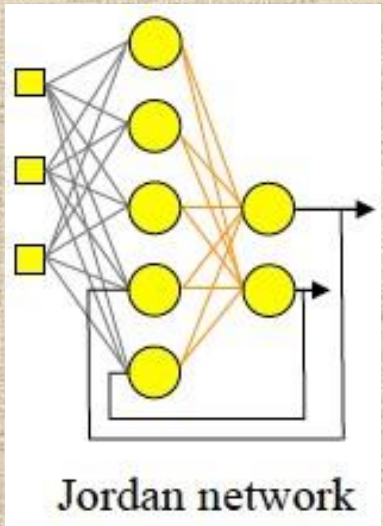


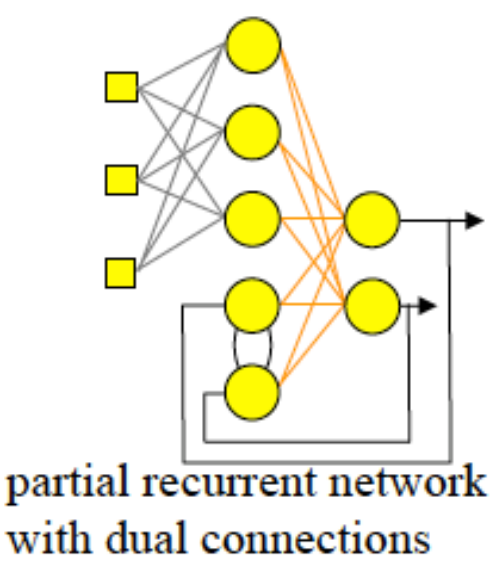


Différentes architectures neuronales

Quelques architectures :

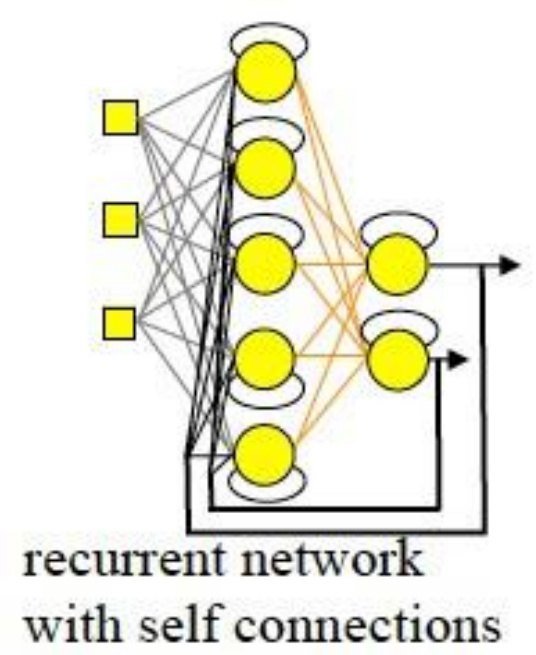






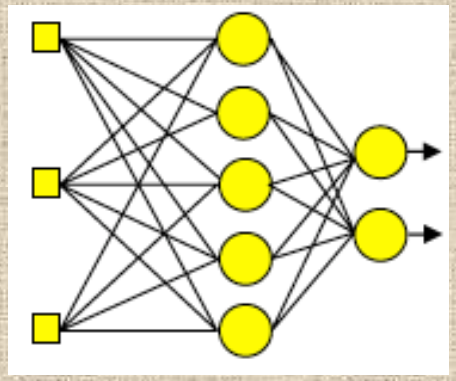
Différentes architectures neuronales

Quelques architectures :

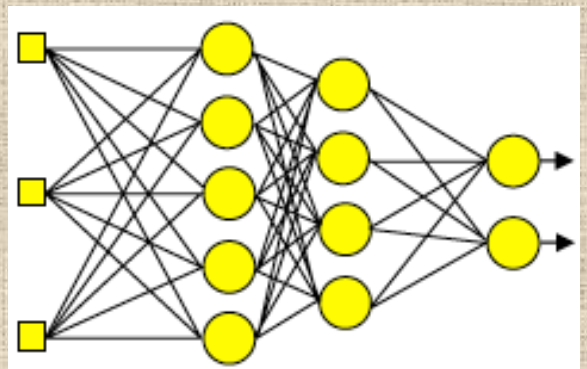


Différentes architectures neuronales

Le réseau de neurones multicouche : 2 exemples
(multilayer feedforward network, ou Multilayer Perceptron= MLP



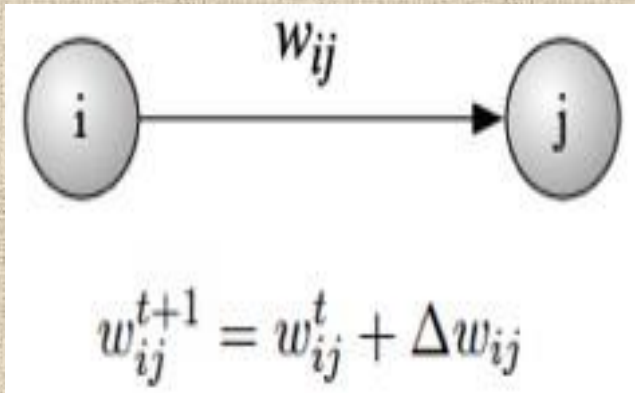
- Ce réseau a :
- 1 couche de 3 entrées
- 1 couche caché (5 neurones)
- 1 couche de sortie (2 neurones)



- Ce réseau a :
- 1couche de 3 entrées
- 2couche cachées (de 5 et 4 neurones)
- 1 couche de sortie (2 neurones)

Apprentissage

- Les connaissances dans un réseau ne sont pas codées dans les neurones mais dans la configuration du réseau (poids des connections)
- Le principe d'apprentissage repose sur la modification de ces poids (d'une façon incrémentale).
- Le scalaire w_{ij} représente le poids associé à la connexion entre les neurones i et j.



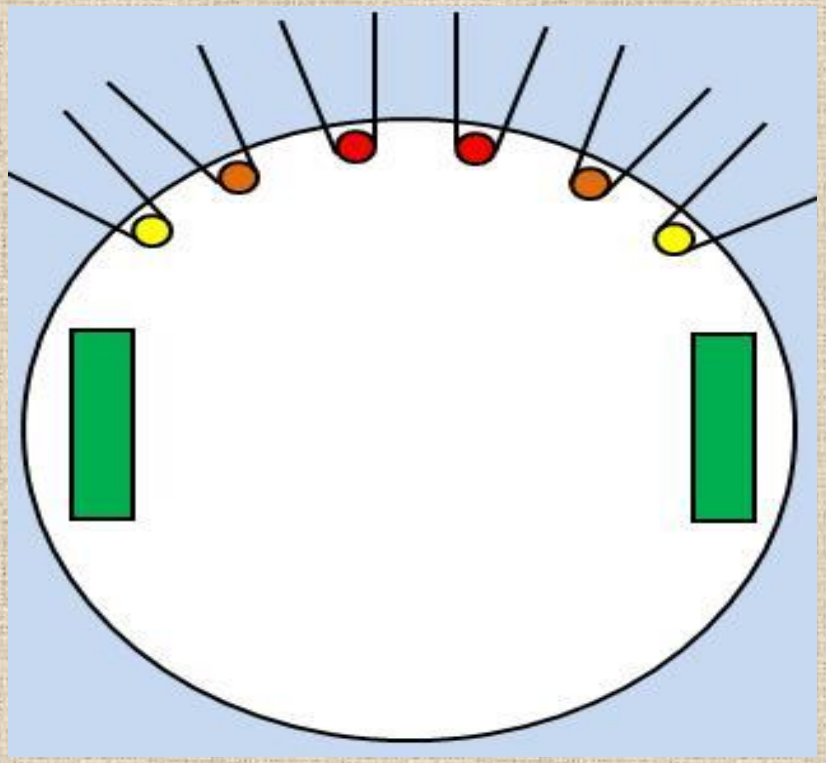
Exemple pédagogique:
Constitution du robot

1-Partie "hardware"

-

les principaux composants du robot

•en vert : 2 moteurs à courant continu
•en jaune, orange et rouge : 6 capteurs de distance
•un micro-rupteur (= interrupteur) qui se déclenche si le bord du robot touche un obstacle(non représenté)
•L'alimentation, l'électronique employée (ne sont pas détaillées)



Exemple pédagogique

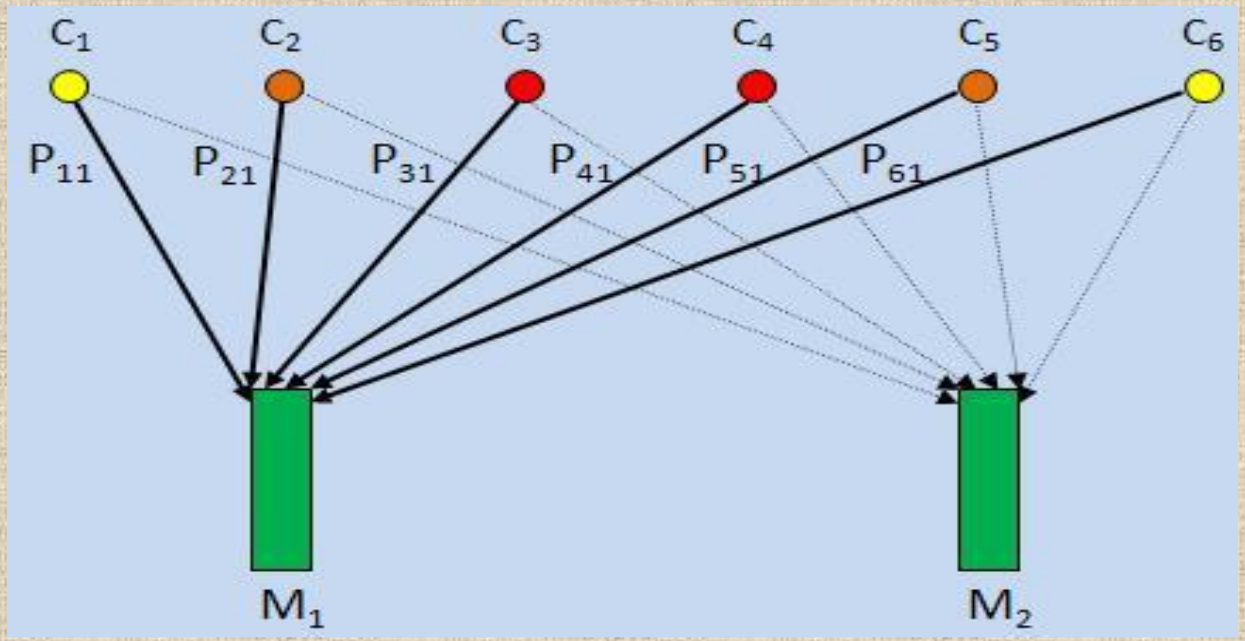
L'objectif de cette présentation est de comprendre le fonctionnement des réseaux de neurones : et elle sera applicable à n'importe quel robot ayant cette configuration et pouvant être programmé à vos souhaits.

2-Partie "software" : le réseau de neurone proprement dit

Un réseau de neurone relie les entrées d'un système à ses sorties.

Les entrées et les sorties sont appelés des nœuds. Le "lien" qui relie deux nœuds est appelé un axiome.

Voici un schéma détaillant les notations utilisées :



On ne va s'intéresser qu'au moteur gauche du robot.

On distingue clairement sur ce schéma - grâce aux traits en gras - que toutes les entrées (les Ci) sont reliées au moteur (M1). Sur chaque lien, on peut voir d'autres notations : P11, P21, P31, P41, P51 et P61 (ou encore les P_{ik} - P indice i indice k - avec i allant de 1 à 6, et k valant 1 puisqu'on n'étudie que le moteur gauche) C'est ce qu'on appelle les poids. Grâce à ces notations, voici l'équation du réseau de neurone que l'on vient de créer - pour le moteur gauche uniquement :

$$M_1 = C_1 \times P_{11} + C_2 \times P_{21} + C_3 \times P_{31} + C_4 \times P_{41} + C_5 \times P_{51} + C_6 \times P_{61}$$

Exemple pédagogique

- Les valeurs des capteurs sont données par les capteurs eux-même, en fonction de la distance qui les séparent d'un obstacle extérieur.
- Les valeurs des moteurs sont contrôlées par le logiciel qui est sur le robot... ici, il sera contrôlé grâce au réseau de neurone, dont on a écrit l'équation juste au-dessus.
- Il nous manque donc plus qu'à connaître les valeurs des poids (Pik)...

c'est justement ces poids que le robot doit apprendre à déterminer, de lui-même !

- Pour comprendre comment lui faire cela, nous allons d'abord essayer de voir quelles valeurs pourraient fonctionner pour nos Pik, pour que le robot puisse éviter les obstacles...

Établissement manuel du réseau de neurone

Comportement désiré:

Pour des raisons de compréhension, on va simplifier le comportement du robot. **Le robot doit toujours avancer**, sauf dans les cas suivants:

- Si un obstacle se présente sur la **gauche**, le robot doit tourner à **droite** (c'est à dire que le moteur gauche doit "**avancer**" - tandis que le moteur droit doit "**reculer**")
- Si un obstacle se présente sur la droite, le robot doit tourner à gauche (c'est à dire que le moteur gauche doit "reculer" - tandis que le moteur droit doit "avancer")

- Et si un obstacle se présente au centre ? Ceci est un faux problème car les capteurs ne donneront jamais constamment la même valeur à droite et à gauche :

Établissement manuel du réseau de neurone

Nous allons donc traiter ces deux cas, l'un après l'autre, ce qui nous permettra de donner des valeurs fonctionnelles pour P11, P21 et P31 d'une part, puis P41, P51 et P61 d'autre part.

1-Si un obstacle se présente sur la gauche...

... dans ce cas, les capteurs de gauche, C1, C2 et C3 auront une valeur importante (les capteurs de distance renvoient une valeur d'autant plus importante que l'obstacle est proche).
Or, on voulait que si un obstacle se présente à gauche, le moteur gauche devait avancer... pour cela, on doit lui envoyer une valeur positive. :

Vu ce que l'on cherche à faire, le plus simple pour répondre à cette demande serait - en premier lieu - d'imposer P11 = 1, P21 = 1 et P31 = 1 (ce sont des valeurs possibles). Notre équation devient donc :

$$M_1 = C_1 + C_2 + C_3 + C_4 \times P_{41} + C_5 \times P_{51} + C_6 \times P_{61}$$

Établissement manuel du réseau de neurone

2-Si un obstacle se présente sur la droite...

... alors là, à l'inverse, ce sont les capteurs C4, C5 et C6 qui auront une valeur importante. Or, on voulait que si un obstacle se présente à droite, notre robot doit tourner à gauche, et pour cela, le moteur gauche doit "reculer"... pour cela, il faut lui envoyer une valeur négative. On va imposer P41 = -1, P51 = -1 et P61 = -1. Ainsi :

$$M_1 = C_1 + C_2 + C_3 - C_4 - C_5 - C_6$$

la méthode employée n'est pas très catholique : cependant, cela fonctionne ! Vérifiez par vous-même... si un obstacle arrive à gauche, C1, C2 et C3 auront une valeur importante, et C4, C5 et C6 auront une valeur faible.
d'autant plus faible que l'objet est loin, voire nulle si l'objet est à l'infini... on peut donc considérer que dans le cas où un obstacle arrive sur la gauche, seules les valeurs de C1, C2 et C3 sont pris en compte... et regardez l'équation dans ce cas là :

Établissement manuel du réseau de neurone

$$M_1 = C_1 + C_2 + C_3 > 0 !!!$$

!!! Notre moteur gauche va donc "avancer" !

Et ça marche aussi si un obstacle arrive à droite : C1, C2 et C3 sont faibles cette fois-ci, voire négligeable... alors que C4, C5 et C6 auront une valeur importante, et dans ce cas, l'équation devient :

$$M_1 = -C_4 - C_5 - C6 < 0 !!!$$

!!! Notre moteur gauche va donc "reculer" !

Et le moteur droit alors ?

Cela pourrait faire un très bon exercice... je vous laisse chercher (il suffit de suivre exactement le même schéma que précédemment), mais voici la réponse :

$$M_2 = -C_1 - C_2 - C_3 + C_4 + C_5 + C6$$

Algorithme génétique d'apprentissage

Algorithme d'apprentissage simple

L'algorithme d'apprentissage simple consiste à apprendre par essais et erreurs.

Le robot commence avec des **poids choisis aléatoirement**, puis il teste son comportement dans l'environnement.

Pour évaluer si ces poids sont bons ou mauvais, on utilise un **nouveau capteur** : un **interrupteur (micro-rupteur)** qui se déclenche quand le robot touche un obstacle.

•Si le capteur se déclenche : les poids sont considérés comme *mauvais*.

→ On génère alors **un nouveau tirage aléatoire** des poids.

•Si le capteur ne se déclenche pas : les poids permettent au robot d'éviter les obstacles.

Ce processus est répété **jusqu'à trouver des valeurs de poids** permettant au robot de **ne plus toucher d'obstacles**.

C'est une forme primitive d'apprentissage automatique basée uniquement sur l'évaluation du résultat.

Algorithme génétique d'apprentissage

Algorithme génétique d'apprentissage

L'idée est d'améliorer l'apprentissage du robot en s'inspirant du principe de l'évolution génétique.
Au lieu d'un seul tirage aléatoire des poids, on crée **plusieurs "jets"** (par exemple une dizaine), chacun correspondant à un ensemble de poids différents.
Évaluation de la génération :
Pour chaque jet, on laisse le robot évoluer dans une arène pendant plusieurs minutes.
On mesure alors le **nombre de collisions** pour chaque jet.
Sélection :
On garde **le jet le plus performant**, c'est-à-dire celui pour lequel le robot s'est cogné **le moins souvent**.
Ce jet devient le **parent** de la génération suivante.

Algorithme génétique d'apprentissage

Création d'une nouvelle génération :

Les nouveaux jets ne sont plus entièrement aléatoires :

on conserve **au moins un poids provenant du jet parent** (héritage génétique),

les autres poids peuvent être choisis aléatoirement (mutation).

Boucle d'évolution :

On répète le processus :

→ on teste tous les jets

→ on sélectionne le meilleur

→ on produit une nouvelle génération à partir de lui.

Progressivement, les poids évoluent et le robot devient **plus performant à éviter les obstacles**, sans qu'on intervient directement dans son comportement.

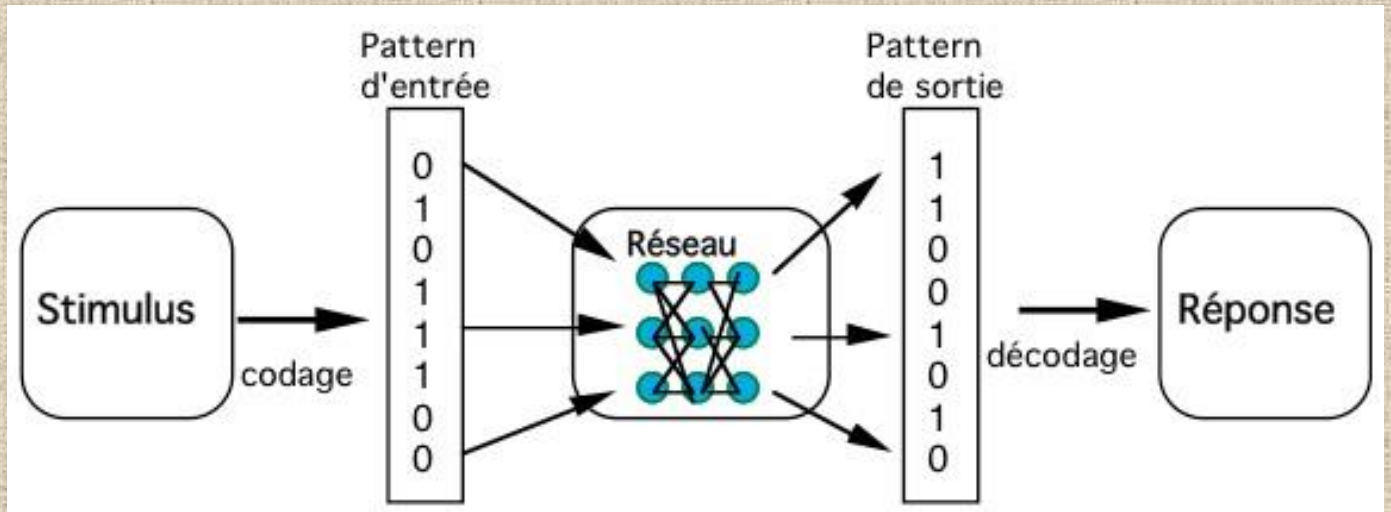
Algorithme génétique d'apprentissage

Dans les générations suivantes du processus évolutif, on augmente progressivement la quantité d'information héritée du parent :
Génération 1 : on conserve *1 seul* poids du meilleur jet (le parent).
Génération 2 : on conserve *2 poids* du parent.
Génération 3 : on en conserve *3*, et ainsi de suite…
Ainsi, au fil des générations, les nouveaux jets (les « enfants ») ressemblent de plus en plus au jet parent, c'est-à-dire au meilleur individu de la génération précédente.
Cette accumulation progressive d'héritage permet :
de **préserver les bonnes solutions**,
d'**améliorer le comportement du robot** étape par étape,
de transmettre un « savoir-faire » acquis au cours des générations.
Les enfants finissent par adopter les comportements efficaces des parents, mais de manière encore plus optimisée grâce aux variations (mutations) introduites à chaque génération.
Ce mécanisme illustre parfaitement le **principe simple mais puissant des algorithmes génétiques** :
→ *sélection des meilleurs individus + reproduction avec héritage + mutations contrôlées.*

Remarques

Les comportements du robot (avancer , gauche, droit, reculer, arrêter, „„) ?
Les actions du robot ?
Le type des résultats de l'algorithme (valeurs entières, réelles, …) ?
Le type de l'environnement(un seul obstacle, plusieurs obstacles,…) ?

Architecture générale d'un RNA



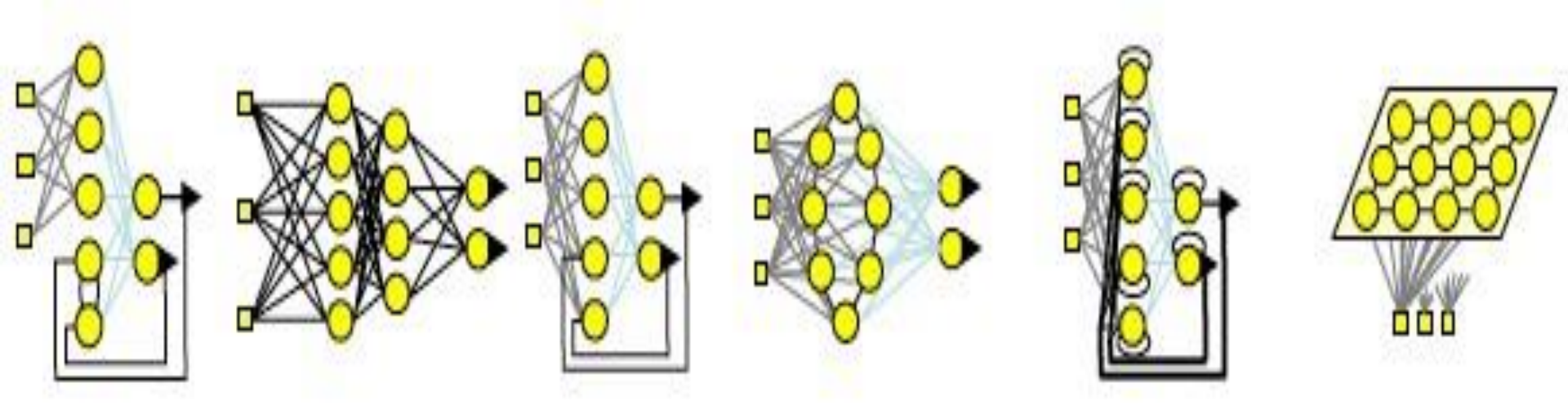
Stratégies d'apprentissage

L'apprentissage au sein des différentes architectures dépond de l'architecture du réseau et de l'environnement du problème.

Les 2 règles d'apprentissage pour mettre (faire) corriger les poids d'un neurones (règle de Hebb et Windrow) ne concernent qu'un neurone seul.

Ces règles peuvent servir pour mettre à jour les poids d'un neurone, de certains de neurones, mais ne peuvent être généralisées et s'appliquer à n'importe quelle architecture.

Chaque architecture possède ses spécificités et nécessite une règle d'adaptation des poids qui lui est propre.



4 types de règles d'apprentissage

- Correction d'erreur
- Apprentissage de Boltzmann
- Règle de Hebb
- Apprentissage par compétition

Correction d'erreur:

Si d est la sortie désirée et y la sortie obtenue, la modification des poids se fait en fonction de (d-y)

Algorithme de retro-propagation

Algorithme d'apprentissage du perceptron:

Initialisation des poids et du seuil à de petites valeurs aléatoires

Présenter un vecteur d'entrées $x^{(\mu)}$ et calculer sa sortie

Mettre à jour les poids en utilisant :

$$w_j(t+1) = w_j(t) + \eta\,(d - y)\,x_j$$

avec d la sortie désirée, w vecteur des poids

Stratégies d'apprentissage

L'**apprentissage** est le processus qui permet de **modifier les poids** d'un système (neurones artificiels, robot, modèle) afin d'améliorer son comportement ou ses prédictions.

Les poids sont ajustés à partir de **mesures**, **observations** ou **retours**, selon différentes stratégies.

On distingue principalement **trois grands types d'apprentissage** :

Stratégies d'apprentissage

- **L'apprentissage supervisé**
- C'est le type d'apprentissage le plus courant.

Il repose sur l'idée suivante :

- On fournit au système **des données d'entrée** *et* les **réponses correctes** correspondantes, qu'on appelle *cibles* ou *labels*.
- Le modèle apprend en comparant :
  - **sa sortie**
  - **la sortie attendue (la bonne réponse)**
- Puis il corrige ses poids pour réduire l'erreur.

Exemple :

- **Entrée** : image d'un chat **Sortie attendue** : « chat »
- **Le réseau prédit** « chien » → erreur **Les poids sont ajustés** pour réduire cette erreu

Utilisations :

Classification d'images, Reconnaissance vocale, Prédiction (météo, finance…),

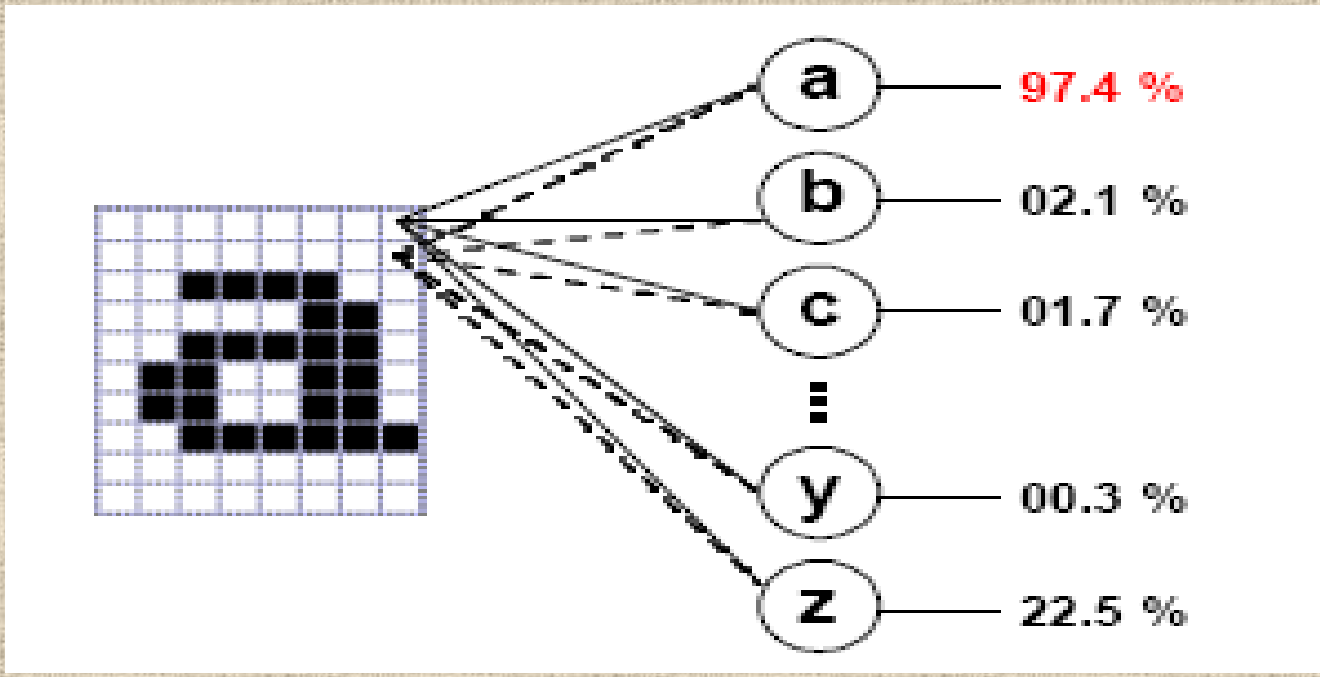
Traduction automatique

- **C'est un apprentissage très efficace, mais** il **nécessite beaucoup de données annotées**.

Exemple de classification

Classification :
Reconnaissance de formes :

- les lettres de l'alphabet
- les chiffres de 0 à 9



Stratégies d'apprentissage

- **L'apprentissage non supervisé**
- il n'y a **pas de bonnes réponses fournies**.

Le système ne reçoit **que les données d'entrée**, sans aucune indication sur ce qu'il doit trouver.

- Le but est de **découvrir automatiquement une structure cachée** dans les données :
- regrouper les données similaires (**clustering**),
- trouver des motifs récurrents,
- réduire la dimension des données.

Exemple :

- On donne des milliers d'images d'animaux **sans étiquettes**.
- Le modèle regroupe naturellement les images par similarité :

groupe 1 : chats
groupe 2 : chiens
groupe 3 : oiseaux

Stratégies d'apprentissage

L'apprentissage non supervisé

<u>Utilisations</u> :
Analyse de données (clients, comportements…)
Détection d'anomalies
Compression
Prétraitement avant apprentissage supervisé

Ce type d'apprentissage est utile lorsque **les données annotées sont rares ou coûteuses**.

Stratégies d'apprentissage

- **L'apprentissage par assignation de crédit**
- (*aussi appelé apprentissage par renforcement ou apprentissage basé sur la récompense*)
- Ce type d'apprentissage correspond à une situation où un agent (robot, programme…) **interagit avec un environnement** :
- il effectue des actions,
- reçoit des conséquences,
- et cherche à maximiser une **récompense**.
- L'**assignation de crédit** consiste à déterminer **quelles actions sont responsables du succès ou de l'échec**, afin d'ajuster les poids en conséquence.
- 🔍 **Exemple :**
- Un robot avance dans une arène :
- S'il évite un obstacle → récompense positive
- S'il cogne un mur → pénalité
- Le robot apprend quelles séquences d'actions mènent aux bons résultats.

Stratégies d'apprentissage

- **L'apprentissage par assignation de crédit**

Utilisations :

- Robotique
- Jeux
- Contrôle automatique
- Systèmes autonomes (drones, voitures…)

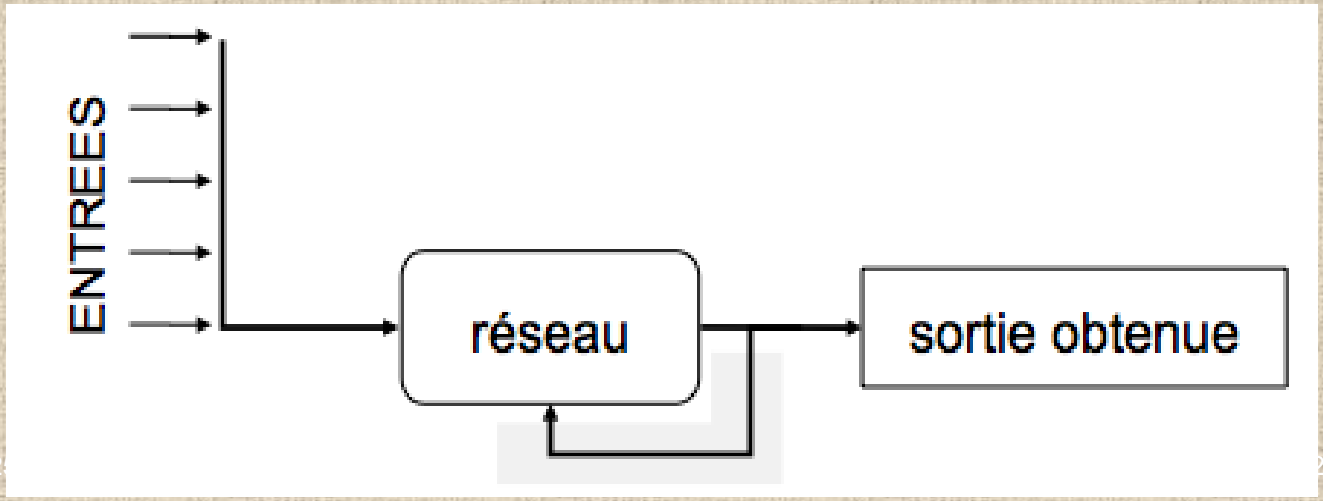
C'est un apprentissage très puissant pour des **agents autonomes**, car il ne demande ni réponses exactes (comme en supervisé) ni structure initiale (comme en non supervisé).

Apprentissage supervisé/ Apprentissage non supervisé

Apprentissage non supervisé

L'apprentissage supervisé s'effectue sous le contrôle d'un expert, alors que l'apprentissage non supervisé (appelé en anglais «unsupervised Lerarning» est autodidacte.

on ne dispose pas de base d 'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales.

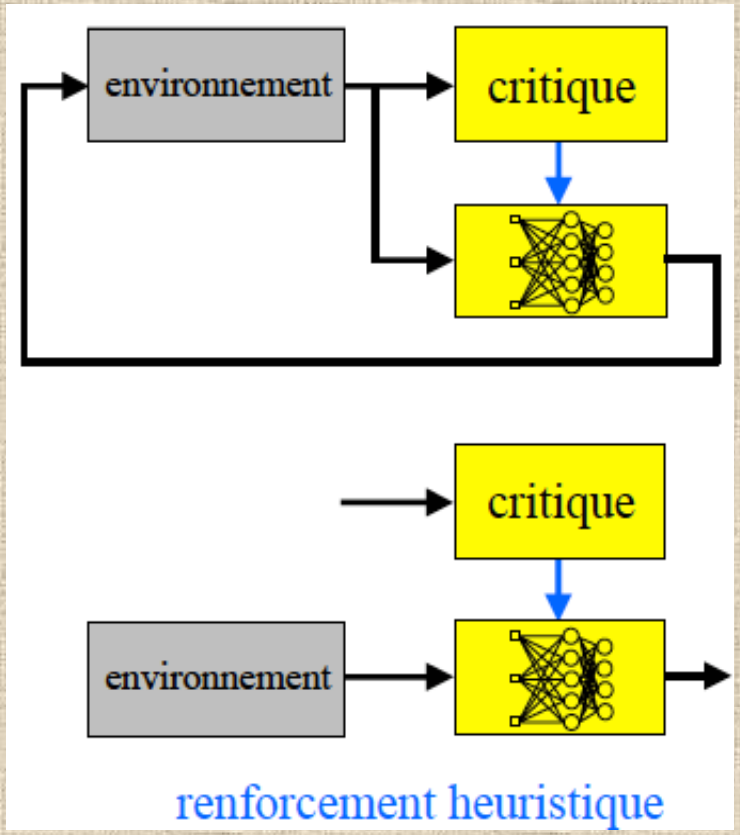


Apprentissage non supervisé

Apprentissage par assignation de crédits

Un apprentissage qui ne nécessite pas de comportement de référence explicite mais seulement d'informations grossières, comme un encouragement ou une pénalisation, est appelé apprentissage par renforcement.

Ce type d'apprentissage fait partie d'un schéma plus large d'apprentissage, le «Crédit-Assignement Problem» ou CAP.



Applications

Aujourd'hui, les réseaux de neurones artificiels ont de nombreuses applications dans les secteurs très variés :

Traitement d'images : reconnaissance de caractère et de signature, compression d'images, reconnaissance de forme, cryptage, classification, ect.

Traitement du signal : filtrage, classification, identification de source, traitement de la parole. . .

Contrôle : commande de processus, diagnostic, contrôle qualité, asservissement de robots. . .

Optimisation : planification, allocation de ressource, gestion et finances, etc.

Simulation : simulation de boite noir, prévision météorologique,

Créatures artificielles:

Evolution du réseau

- L'apprentissage évolutionnaire a pour but d'appliquer un algorithme génétique à l'évolution de contrôleur.

- Pour cela, on crée une population initiale, c'est-à-dire un ensemble de contrôleurs, et on poursuit le processus en boucle..

- L'application des algorithmes évolutionnistes pour la génération des réseaux de neurones semble une idée simple.

- Théoriquement, la flexibilité des algorithmes évolutionnistes (qui exigent seulement une fonction de fitness représentant un indicateur de progrès) pourrait permettre de surmonter les limitations de la formation arrière de propagation, particulièrement quand elle concerne les architectures complexes et récurrentes.

Evolution du réseau

L'apprentissage évolutionnaire a pour but d'appliquer un algorithme génétique à l'évolution de contrôleur.

Pour cela, on crée une population initiale, c'est-à-dire un ensemble de contrôleurs,

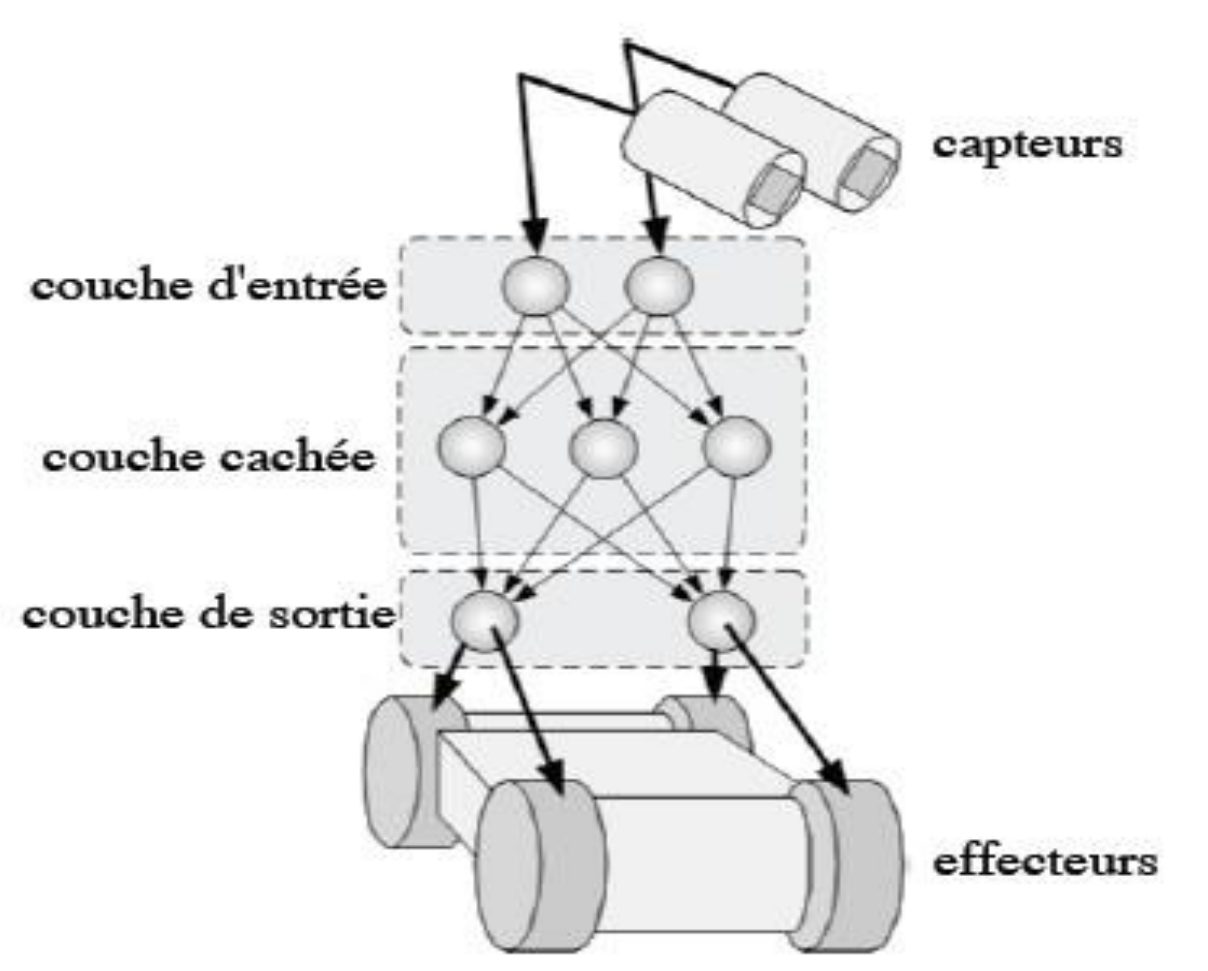
Chacun d'eux est ensuite évalué en situation dans le but de déterminer lesquels sont les plus aptes à survivre dans l'environnement.

À la suite d'une longue évolution, dans laquelle des générations de contrôleurs sont recombinés et altérés de plus en plus adaptés font leur apparition dans la population.

Evolution du réseau

L'apprentissage évolutionnaire utilise un **algorithme génétique** pour faire évoluer des contrôleurs. On crée une population de contrôleurs, on les évalue dans l'environnement, puis on sélectionne les mieux notés (fitness) pour produire la génération suivante. Au fil des recombinations et mutations, apparaissent des contrôleurs de plus en plus adaptés. Cette approche est bien adaptée aux réseaux de neurones complexes et s'intègre naturellement à l'**apprentissage par renforcement**, puisque la performance vient directement de l'environnement.

Le perceptron contrôleur

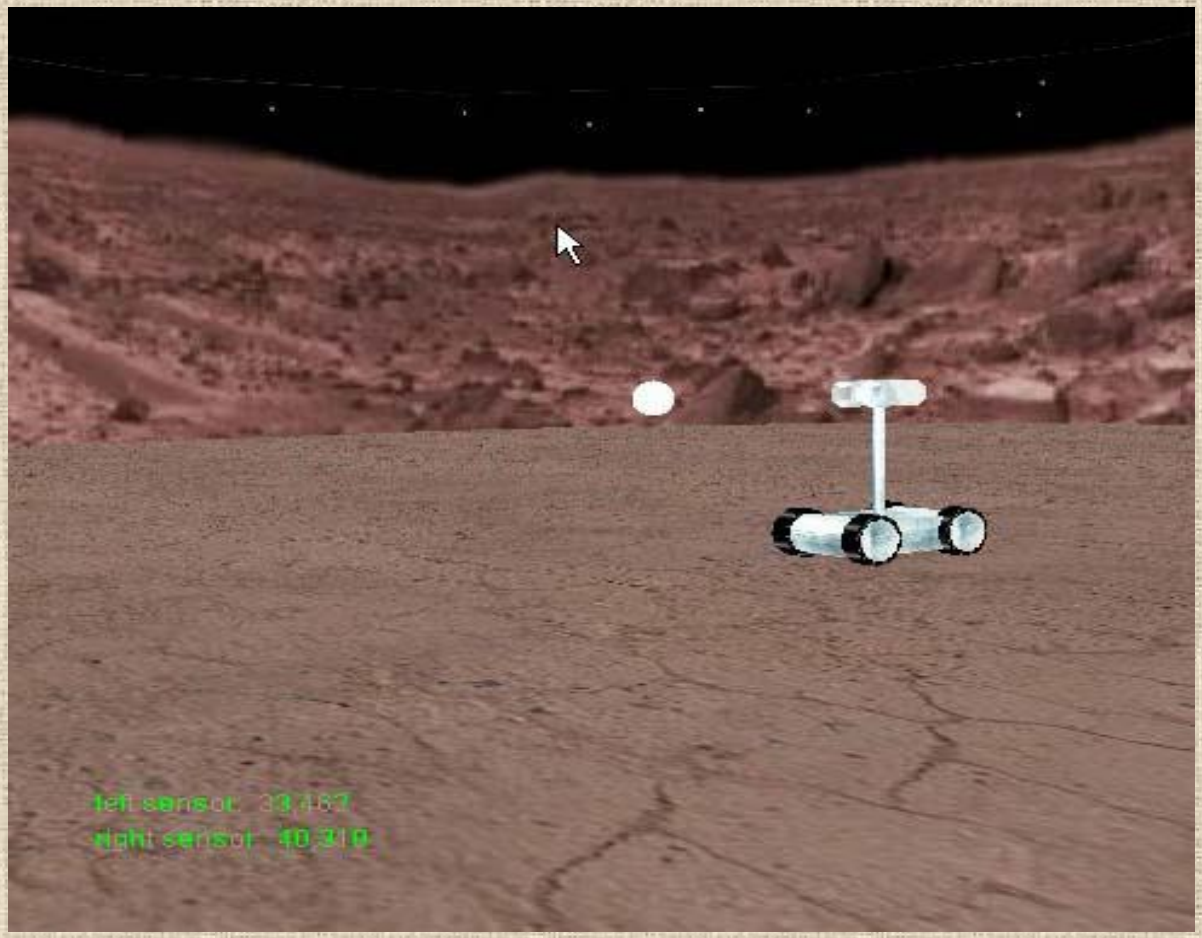


Le contrôleur de l'agent est un PCM.

Les capteurs sont reliés aux neurones de la couche d'entrée.

Les neurones de la couche sortie sont reliés aux effecteurs.

Le perceptron contrôleur



L'agent perçoit son environnement grâce à des capteurs et produit des actions motrices grâce à des effecteurs.

Réseaux évolutionnaire: exemple

Problématique :

• Apprentissage du comportement suivant : Trouver le plus court-chemin jusqu'à un point-cible et éviter des obstacles

Solution :

• Pilotage du robot par un réseau de neurones (PMC)

• Apprentissage par un algorithme évolutionnaire

Pas de base d'apprentissage

L'expérience permet d'apprendre : comportement émergent

Réseaux évolutionnaires: exemple

Modélisation de l'environnement : quadrillage NxM

• Modélisation d'un obstacle : indice de présence Iij d 'une menace sur une case (i ; j)

Iij = 0 si aucune menace et aucun obstacle

Iij = -1 si un obstacle

Iij = -0.5 si un obstacle sur les cases voisines

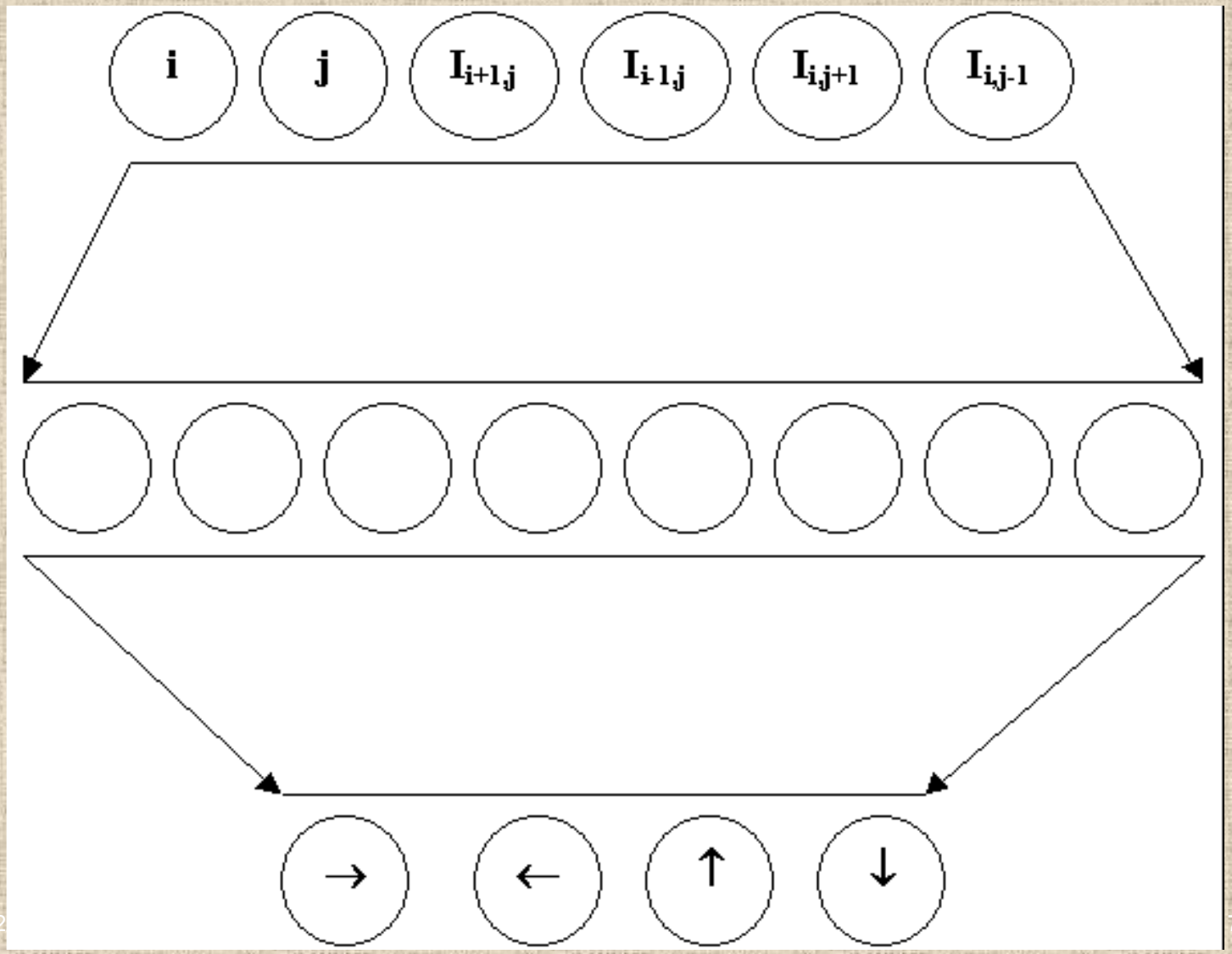
• PMC :

6 entrées : coordonnées du point courant et indices de présence d'un obstacle sur les cases voisines

4 sorties : probabilité de déplacement vers le nord, le sud, l'ouest ou l'est.

La sortie la plus élevée indique la direction à suivre

Réseaux évolutionnaires: exemple

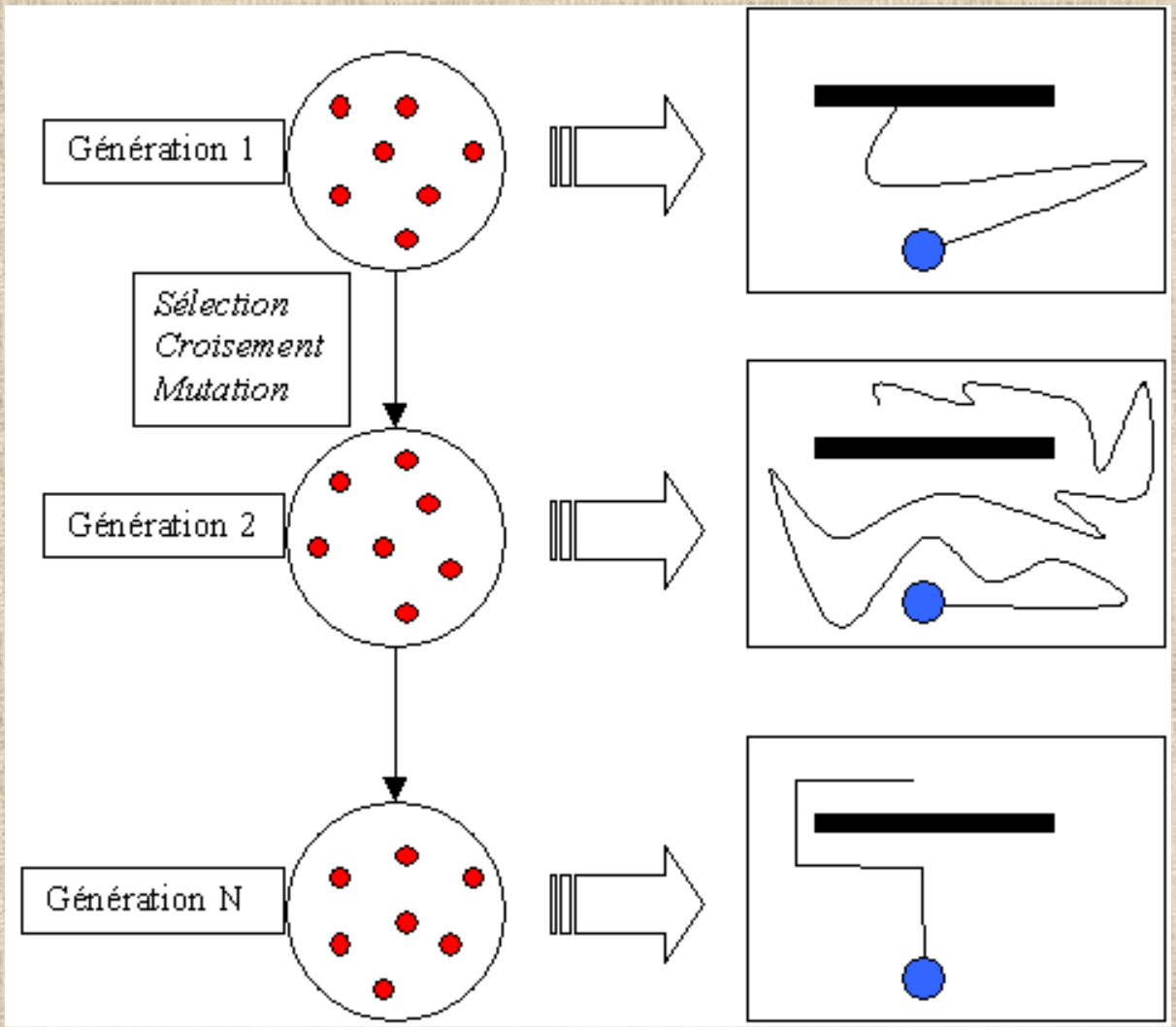


Réseaux évolutionnaires: exemple

Pas de base d'apprentissage ! Pas de couples {*position de départ ; liste des déplacements successifs*}

- Apprentissage par des algorithmes génétiques
  - un individu = 1 PMC = liste des coefficients synaptiques
  - population de PMC
  - opérateurs de croisement, mutation, sélection
  - au bout de N générations : le PMC optimal pour une fonction d 'évaluation (***fitness function***) ⇔ le PMC le plus adapté

Réseaux évolutionnaires: exemple



Principe de l'apprentissage

Pout tout point de départ d'apprentissage Xi
Point_Courant = Xi
nb_iteri = 0
Tant que (point_cible A non atteint) et
(Point_Courant n'est pas dans un obstacle) et
(nb_iteri < nb_iter_max) Faire
Incrémenter nb_iteri
Calculer les 4 sorties du PMC pour Point_Courant (coordonnées et voisinage)
Déplacement_Elémentaire = Max (Sortie →, Sortie ←, Sortie ↑, Sortie ↓)
Déplacer Point_Courant selon Déplacement_Elémentaire
Fin Tant que

Si (Point_Courant est dans un obstacle) Alors
nb_iteri = nb_iter_max
Fin Si
Fin Pour tout

Principe de l'apprentissage

Comment sélectionner les meilleurs réseaux ?

- Tenir compte de plusieurs critères :
  - la distance au point-cible
  - le nombre de déplacements effectués
  - le cumul des indices de présence de menace des cases parcourues
- D 'où la *fitness function* suivante à maximiser :

$$f(x) = \sum_{i=1}^{K} \frac{\alpha}{1+10\times d(X_i, A)} + \frac{\beta}{nb_iter_i}$$

Avec Xi un point de départ parmi les K points de départ d 'apprentissage,
A le point-cible à atteindre et *nb_iteri* le nombre de déplacements effectués depuis Xi.

Principe de l'apprentissage

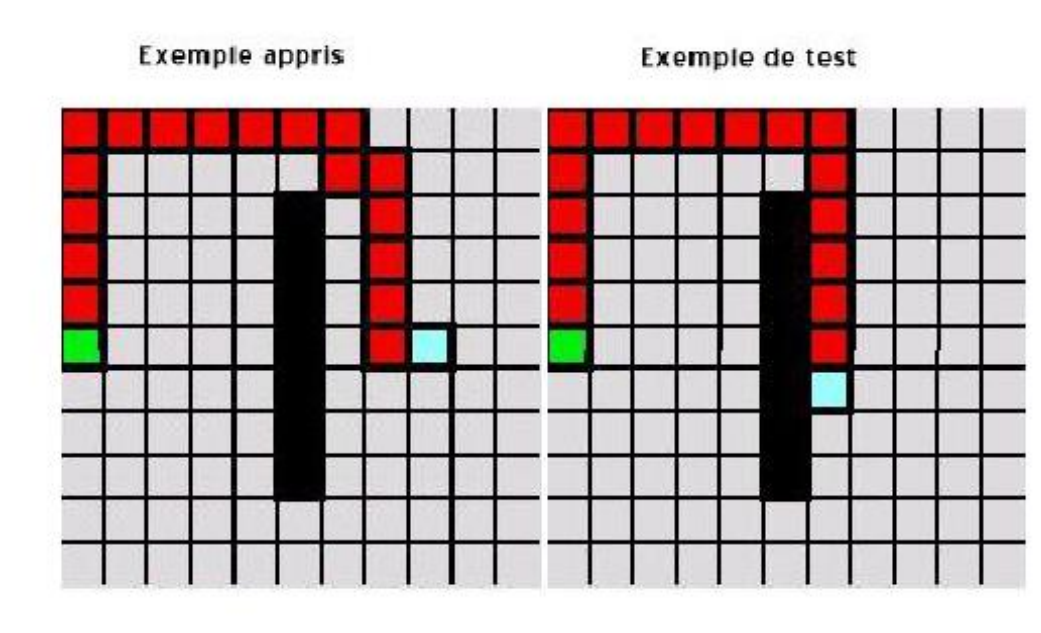
Paramètres AG :

– taille de la population : 100
– nombre de générations : 500
– taux de mutation : 0.1
– taux de croisement : 0.7
– fitness function :

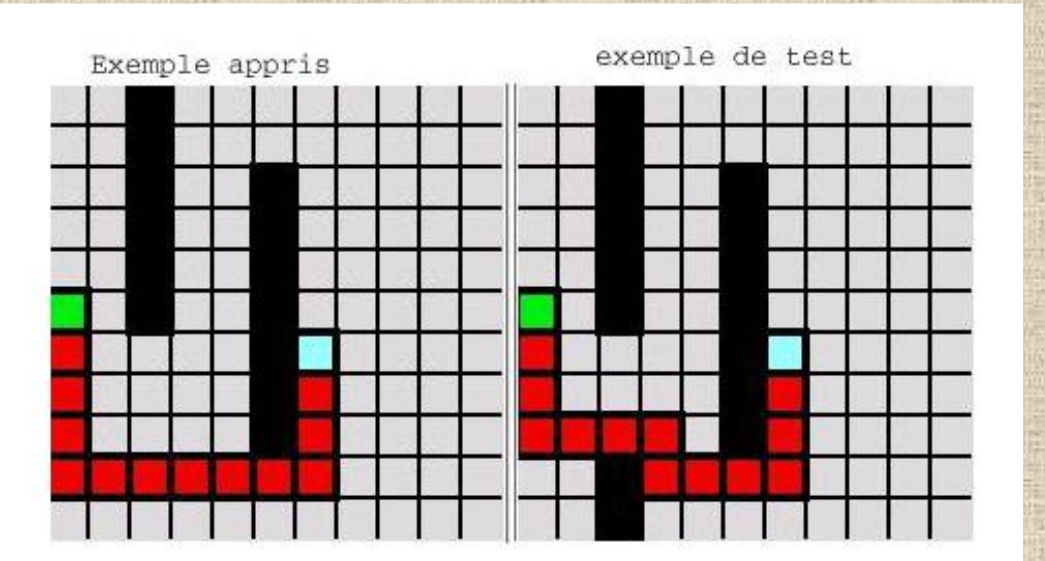
• $\alpha = 500$,
• $\beta = 100$

Principe de l'apprentissage

Tests : généralisation sur
- les points de départ



- la position et le nombre des obstacles



Réseaux de neurones

Avantage: Répondent à des situations non apprises

Inconvénients:

- Longs à paramétrer
- Difficile à interpréter
- Gourmands en temps de calcul

Réseaux de neurones(conclusion)

- il faut préciser que l'apprentissage du réseau ce traduit par la modification des poids de connections entre tous les neurones.
- La complexité de ce genre d'implémentation vient du fait de ce choix limite de topologie qui entraîne dans beaucoup de cas un surdimensionnement du problème qui rend l'apprentissage difficile.
- Aujourd'hui, l'utilisation de ce genre de réseaux est en pleine expansion, comme toutes les sciences de la vie artificielle.
- Le nombre de couches cachées et le nombre de neurones par couche ont une influence sur la qualité de l'apprentissage.

- **L'apparition du deep learning (apprentissage profond)**

Les systèmes de classeurs ou classifieurs

Les systèmes de classeurs ou classifieurs: les concepts

A: l'alphabet standard utilisé, A={0,1}, est ajouté un symbole supplémentaire, le *wildcard* #, celui-ci signifie soit 0 soit 1.

Ainsi, le classifieur 11## : 0110 : s'interprète comme " **Si les deux premiers bits du classifieur sont à 1 et si les deux derniers bits sont indifférents, alors sélectionner le classifieur pour envoyer le message 0110** ". Le # permet donc de représenter plusieurs classifieurs par un seul classifieur.

Si N est le nombre de # dans une chaîne, celle-ci est alors équivalente à 2^N classifieurs. Le classifieur de l'exemple précédent peut donc s'unifier avec quatre messages différents.

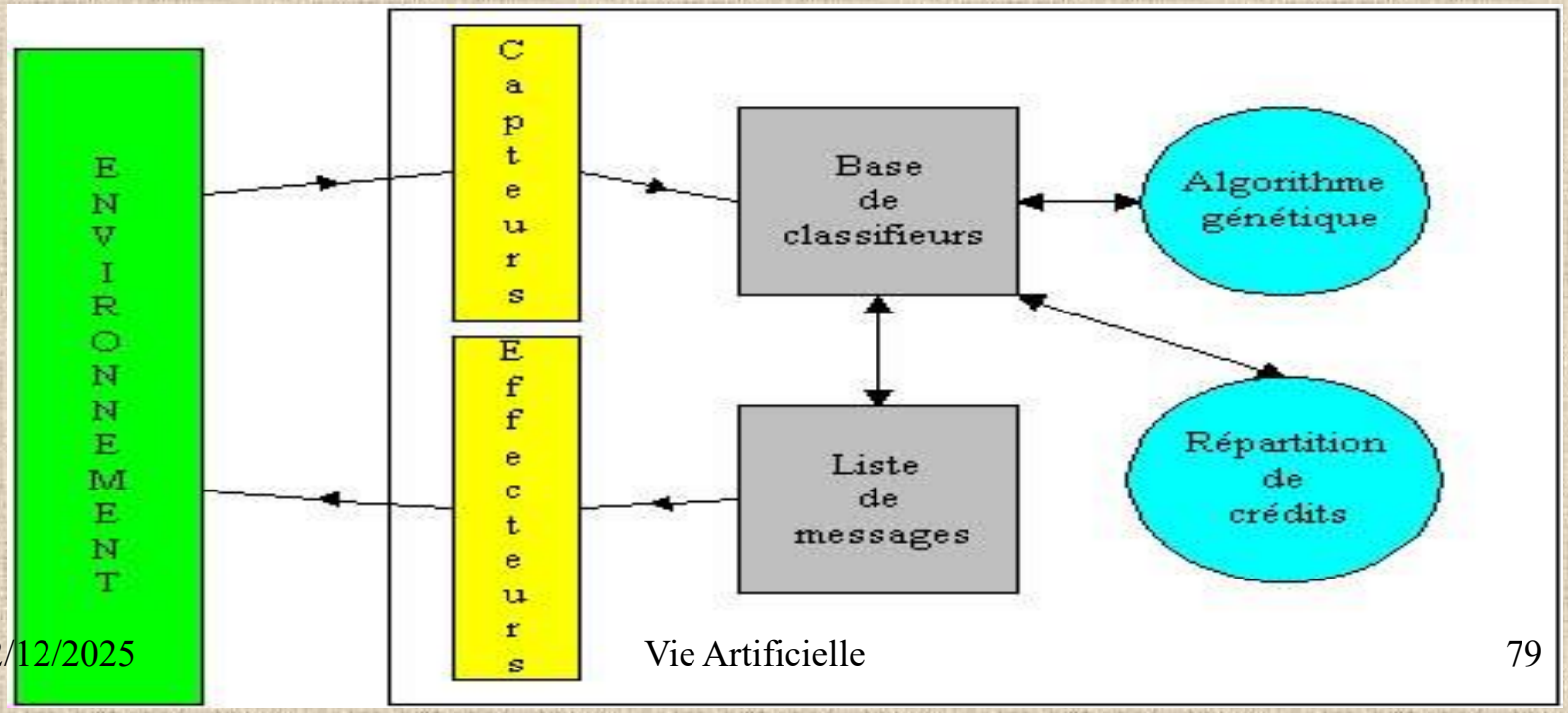
Les systèmes de classeurs ou classifieurs

- Les systèmes de classifieurs, dont les fondements théoriques se trouvent dans [Holland, 62], sont une méthode mêlant apprentissage automatique et évolution
- Les connaissances du système, données sous forme de règles nommées classifieurs
- différences par rapport à un système expert
  1) les règles ne sont ni définies par le programmeur, ni par les experts du domaine
  2) la possibilité de déclencher des règles en parallèle

- Un système de classifieurs est composé de quatre parties :
  1. Une base de classifieurs représentant les connaissances du système.
  2. Une interface d'entrée composée de capteurs.
  3. Une interface de sortie composée d'effecteurs.
  4. Une liste de messages.

Les systèmes de classeurs ou classifieurs

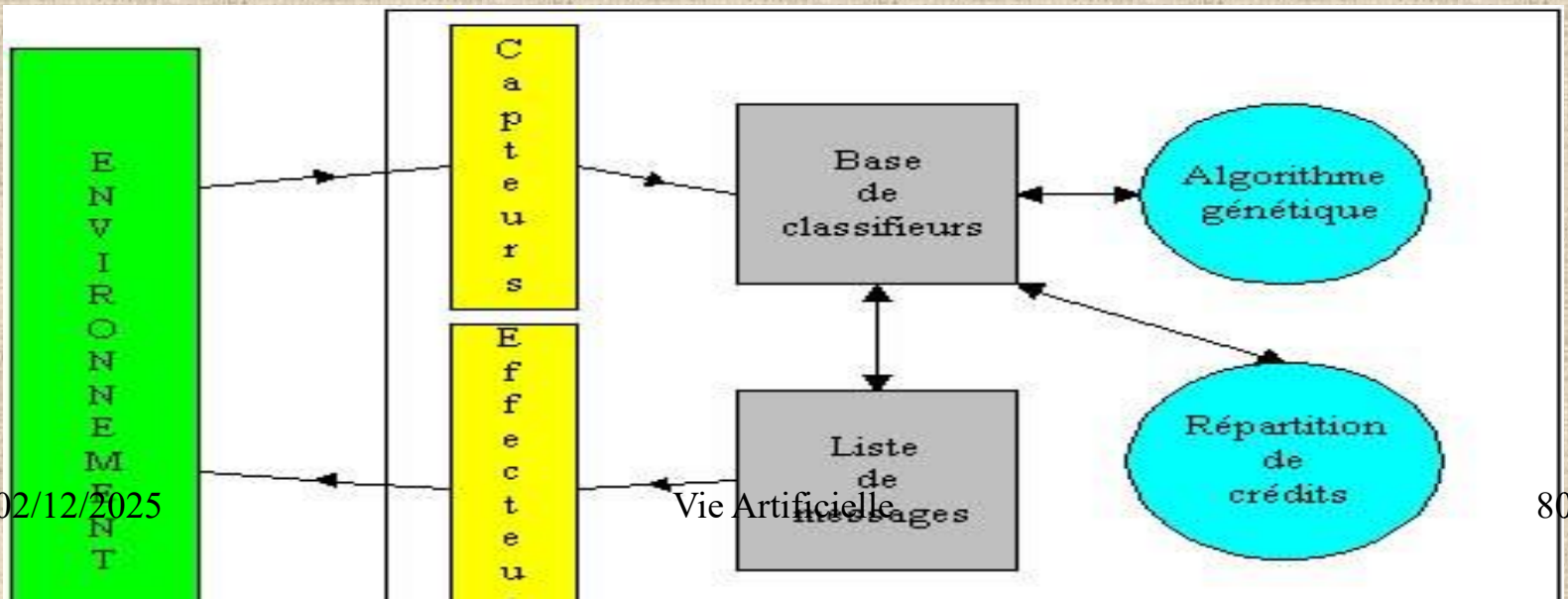
- **Deux algorithmes principaux gèrent alors les classifieurs :**
- **Un algorithme de répartition de crédits distribuant des récompenses aux classifieurs efficaces et pénalisant les inefficaces.**
- **Un algorithme génétique permettant l'évolution de la base de classifieurs.**



Les systèmes de classeurs ou classifieurs

- Le principe de fonctionnement peux se développer dans un ordre défini :

1) Le système reçoit un message
2) Création d'un ensemble de règles correspondant au message
3) Sélection dans cet ensemble des règles qui ont données le résultat le plus pertinent
4) Exécution de l'action et distribution aux règles ayant permis de générer l'action
5) Application de l'algorithme génétique pour améliorer la population de règles
6) Tous les messages de la liste des messages du cycle précédent sont supprimés (les messages persistent pour un seul cycle).



Les systèmes de classeurs ou classifieurs

Les systèmes de classifieurs sont introduits par John Holland en 1978 comme une architecture utilisant les algorithmes génétiques afin d'étudier l'apprentissage des systèmes à bases de règles du type condition/action.

La base des travaux de Holland est fondée sur deux idées essentielles:

- La théorie Darwinienne de survie d'ou le meilleur peut être utilisé pour conditionner l'adaptation d'un système artificiel à un environnement inconnu.

- Apprendre à effectuer une tâche juste en interagissant avec un environnement partiellement connu et en essayant de maximiser les récompenses qu'il reçoit de cet environnement en retour de ses actions.

Les systèmes de classeurs

Le Principe d'un système de classifieurs est donc de faire évoluer un ensemble de règles de production, les classifieurs, afin d'apprendre à un agent la réalisation d'une tâche précise.

L'agent perçoit la totalité ou une portion de l'état de l'environnement à l'aide de capteurs.

Les messages issus de cette perception sont transmis à la base de classifieurs afin de déterminer celui ou ceux qui doivent être activés (en fonction de leur performance) afin de produire une action.

L'action est exécutée à l'aide d'effecteurs et un retour de l'environnement permet une mise à jour des indications de performence des classifieurs.

Les systèmes de classeurs

Pour exécuter un système de classeur, on sélectionne l'ensemble des règles applicables pour la situation en cours et l'on effectue un tirage aléatoire proportionnel aux poids de chaque règle.
Si l'action est mauvaise, on diminue le poids de la règle ayant été sélectionnée. Si au contraire elle a été bénéfique, on augmente ce poids.
L'apprentissage consiste à exécuter un algorithme génétique qui va produire d'autres classeurs avec des règles différentes.

Les systèmes de classeurs

Tout systéme de classifieurs comporte les éléments suivantes :
Une base de régles (les classifieurs)

Un mécanisme d'arbitrage des classifieurs concurrents,

Un mécanisme de renforcement et un mécanisme de découverte des nouvelles règles.

Codage des informations et base de régles

Un classifieur est une représentation génétique formalisée d'une règle de type "SI condition ALORS action".
Un classifieur est donc formé de trois parties distinctes :
Une partie condition, Une partie action
Et une valeur scalaire associée, appelée force

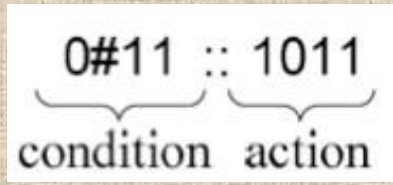
L'ensemble des classifieurs est stocké dans la base de règles qui, à l'instant t de la population d'individus d'un AG, est initialisée aléatoirement au début de l'apprentissage.

Codage des informations et base de régles

La partie condition représente une transcription de l'état courant de l'environnement.

Elle est codée à l'aide de l'alphabet ternaire 0,1,# sous forme d'une chaîne de caractères de longueur fixe.

La partie action, ainsi que les messages en provenance de l'environnement, sont codés à l'aide de l'alphabet binaire 0,1.



Généralisation

Le caractère "#" présent dans la partie condition joue le rôle d'un "Joker" pouvant à la fois représenter "0" ou "1".

Ceci permettra une certaine forme de généralisation de la base de règles.

Il est important de noter que pour qu'un message correspond à un classifieur, chaque bit le composant doit correspondre à chacun des caractères de la condition.

condition	message	correspond?
0##1	0000	non
0##1	0101	oui
0##1	1010	non
0##1	0011	oui

Généralisation

On considère que plus une règle comporte de symboles "#", plus elle est générale. A l'inverse, elle est dite spécifique.

un classifieur est surgénéralisé (over-general) s'il contient un nombre de symboles "#" trop important pour résoudre efficacement le problème posé,

un classifieur généralisé au maximum (maximally general) est un classifieur auquel on ne peut pas ajouter un "#" supplémentaire sans qu'il ne devient surgénéralisé.

un classifieur généralisé sous-optimal (suboptimally general) pourrait être plus généralisé tout en conservant la même performance.

Hiérarchie par défaut

Les classifieurs de plus haut niveau sont les plus généraux. Ils sont capables de répondre à une large gamme de messages provenant de l'environnement.

Un classifieur plus spécifique, de plus bas niveau, répond uniquement à un sous ensemble des états pris en charge par un classifieur plus général.

De ce fait, il répond à moins de situations mais le plus souvent avec moins de risque d'erreurs.

Lorsqu'il est activé à la place d'un classifieur général, un classifieur plus spécifique permet donc à la fois de fournir une réponse correcte mais aussi d'empêcher le classifieur de haut niveau de commettre une erreur.

Force

- La force représente la performance passée du classifieur au sein de l'environnement dans lequel le système apprend.
- La valeur de la force est étroitement liée au retour que le classifieur reçoit de l'environnement :
- plus le classifieur génère de récompenses positives, plus son indice de performance est élevé.
- La force d'un classifieur va remplir deux rôles complémentaires : prédiction et fitness.
- Lors du processus de sélection d'une action, elle sera considérée comme une prédiction de la récompense à venir :
- le classifieur dont la force est la plus élevée est celui qui, théoriquement engendrera la meilleure récompense s'il est activé.

Force

- Enfin, la force est utilisée en tant que fitness lors de l'application de l'algorithme génétique à la population de classifieurs.
- Ainsi, les classifieurs les plus performants ont plus de chance de se reproduire afin d'améliorer la performance globale du système en générant des règles potentiellement mieux adaptées.

Fonctionement des LCS

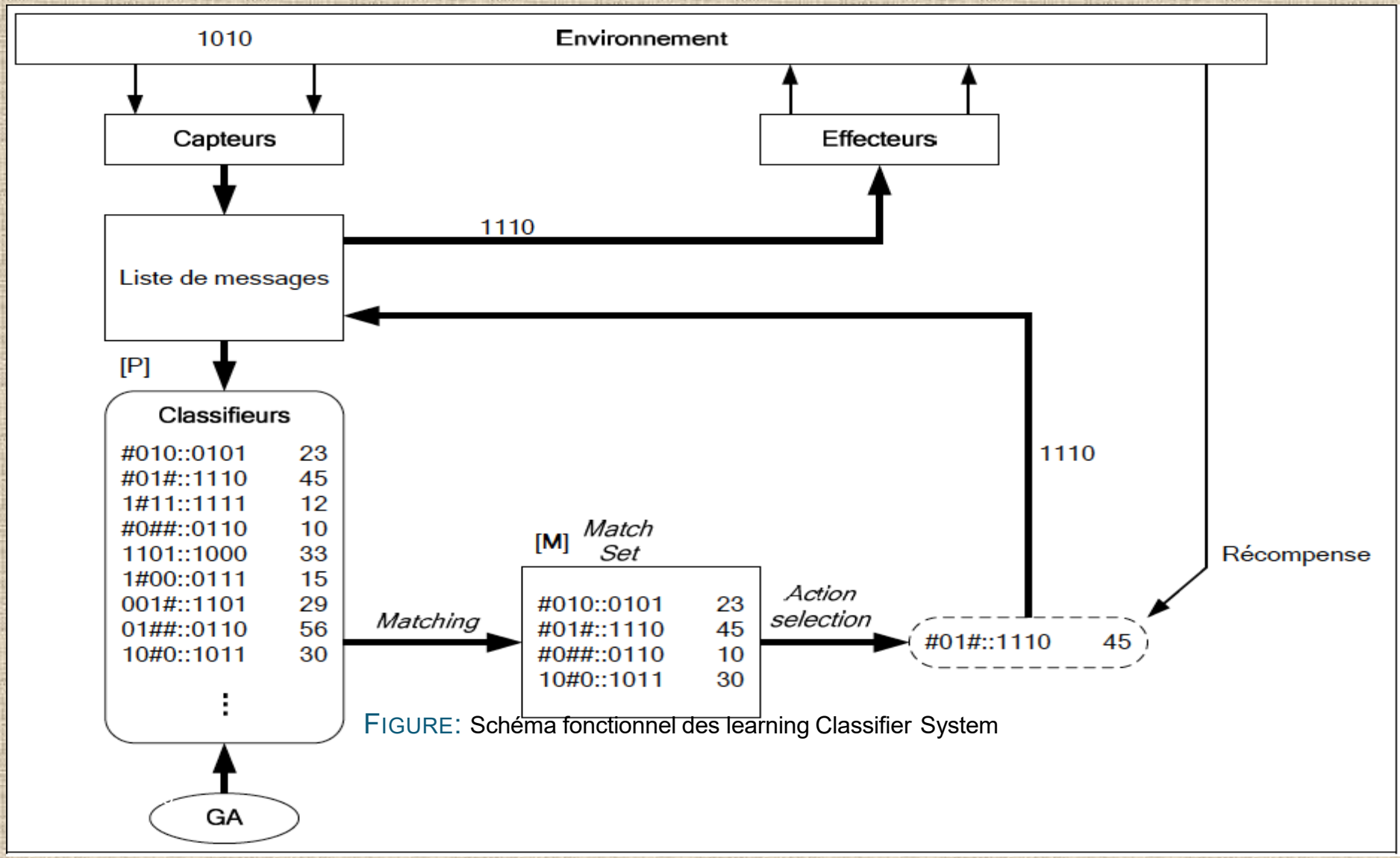


FIGURE: Schéma fonctionnel des learning Classifier System

Fonctionement des LCS

Les systèmes de classifieurs fonctionnent de manière itérative jusqu'à ce que le système ait apprit à résoudre parfaitement la tâche désirée ou, si un apprentissage parfait n'est pas réalisable, jusqu'à l'exécution d'un nombre fixé d'itérations.

Une itération du système se déroule comme suit :

Un message perçu par les capteurs de l'environnement est fournit à la liste de messages.
La base de règles [P] est alors scannée pour déterminer quels sont les classifieurs dont la partie condition correspond au premier message de la liste.
Ces classifieurs deviennent alors membres de l'ensemble des classifieurs correspondants [M] (Match Set).

Fonctionement des LCS

Le système de résolution des conflits va alors déterminer dans l'ensemble [M] quel doit être, en fonction de sa force, le classifieur à activer afin de déclencher son action. Cette action sera stockée dans la liste de messages.

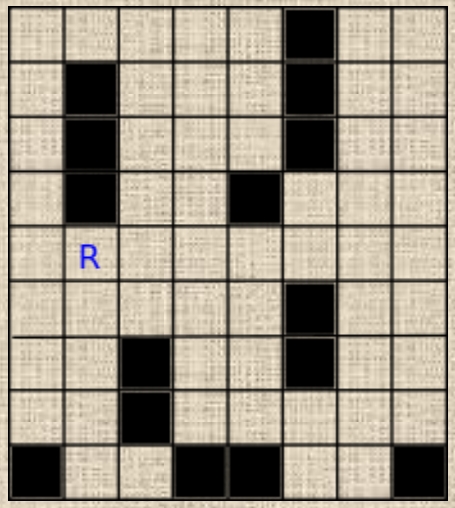
Si elle est éligible, elle sera transmise aux effecteurs pour modifier l'environnement. Sinon, elle sera traitée comme un message provenant de l'environnement (cycle d'inférences internes).

Une fois l'action effectuée dans l'environnement, la liste de messages et remise à zéro, et un algorithme de distribution des crédits assure l'apprentissage par renforcement, en répartissant équitablement parmi les différents classifieurs successivement activés, les récompenses obtenues en retour.

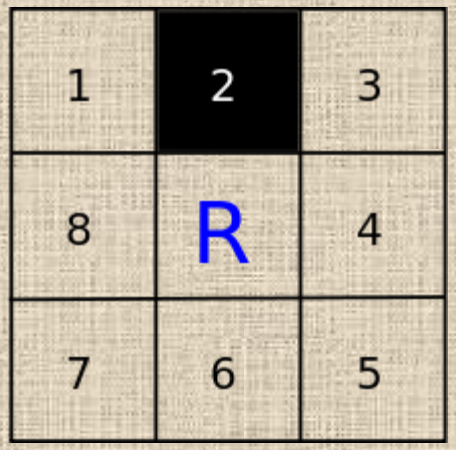
La découverte de nouvelles règles est assurée par l'application régulière d'un algorithme génétique (GA) sur la population [P] des classifieurs.

Exemple :des systèmes de classeurs

Cet exemple présente un système de classeurs très simple.
Considérons un environnement constitué de cases pouvant être remplies ou vides.
Un robot évoluant dans cet environnement peut se déplacer sur une des cases adjacentes si celle-ci est vide.



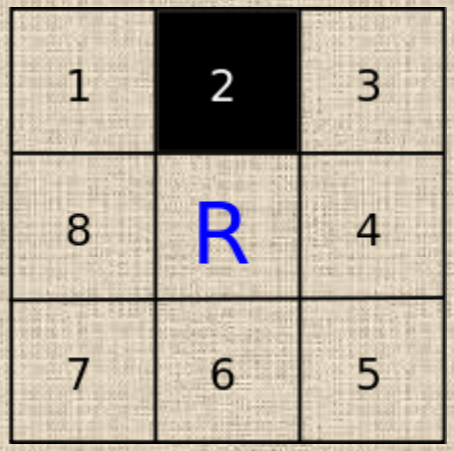
Visualisation d'un état de l'environnement



Numérotation des cases adjacentes au robot

Exemple :des systèmes de classeurs

En numérotant les cases entourant le robot de 1 à 8
on peut écrire une condition de 8 éléments binaires (0 ou 1), où le ième élément représente si la ième case adjacente est pleine ou vide.
Par exemple, la condition 01000000 est vraie si le robot n'a qu'une seule case remplie autour de lui (la case au "dessus", voir figure).

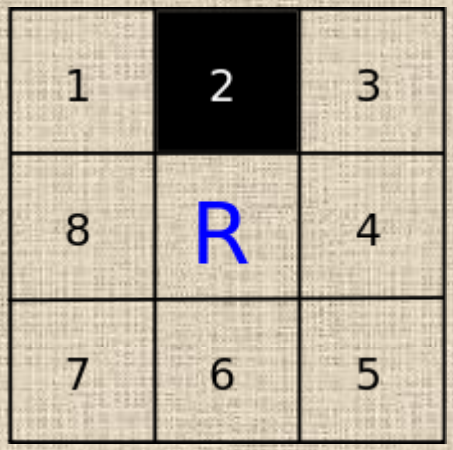


Exemple :des systèmes de classeurs

Un caractère supplémentaire « # » étant une sorte de joker dans la condition. Par exemple : #1#0#0#0 est vrai si la case du dessus est remplie, si les cases de droite, gauche et du dessous sont vides, et quels que soient les états des cases en diagonales.

Numérotons de 1 à 8 les actions possibles, le chiffre correspondant à la case sur laquelle le robot doit se déplacer.

Initialisons maintenant le classeur :
Condition → Action (Poids)
#0###### → 2 (0.5)
#1#0#1## → 3 (0.2)
0#11#0## → 8 (0.4)
0#00#0## → 5 (0.3)
#1#0#0## → 2 (0.9)
...

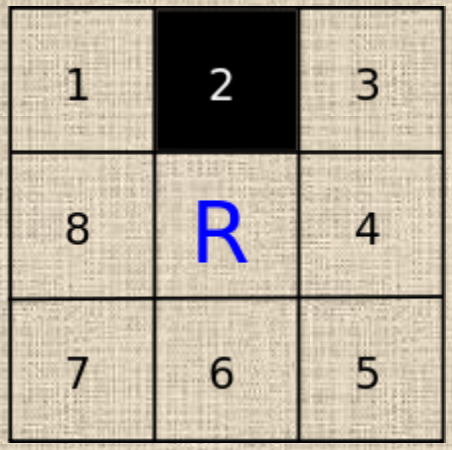


Exemple :des systèmes de classeurs

Dans l'exemple représenté sur le schéma, on sélectionne l'ensemble des règles applicables.
Admettons que nous en ayons 2 :

- 0#00#0## → 5 (0.3) (Règle 1)
- #1#0#0## → 2 (0.9) (Règle 2)

Nous en sélectionnons maintenant une, avec une probabilité proportionnelle à leur poids:
Admettons que la Règle 2 soit sélectionnée. Dans ce cas, le robot effectuera l'action d'aller sur la case 2 qui est remplie. Cette action n'est pas permise par l'environnement.
La règle 2 sera donc "**punie**", c'est-à-dire que l'on va diminuer son poids.
Pour améliorer le comportement du robot, le principe va ensuite être de modifier les règles du classeur (ou même produire d'autres classeur), par un algorithme génétique.



Exemple :des systèmes de classeurs

Exemples:

Nous prenons 2 règles du classeur qui sont assez bien notées afin de les "croiser" :

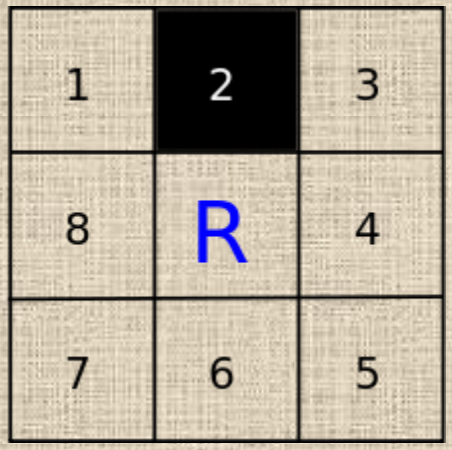
- #1#0#1## → 3 (0.2)
- 0#11#0## → 8 (0.4)

En les croisant, nous produisons 2 autres règles qui viendront remplacer leurs parents ou d'autres règles moins performantes.

Par exemple:

- 0#10#1## → 8 (0.2)
- #1#1#0## → 3 (0.4)

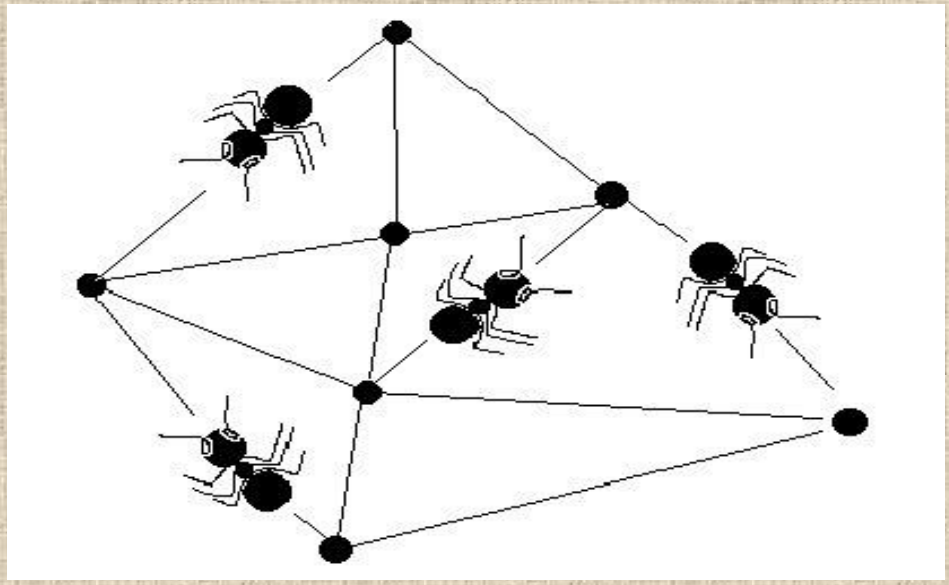
Ces actions sont ensuite exécutées jusqu'à ce que le comportement du système (ici le robot) convienne.



Les 3 leçons principales du vivant

Leçon I: Emergence

- **L'émergence**: ou comment des processus nouveaux (souvent complexes) apparaissent à partir de règles simples, itérées infiniment.
- Les sociétés d'insecte en sont un exemple concret .



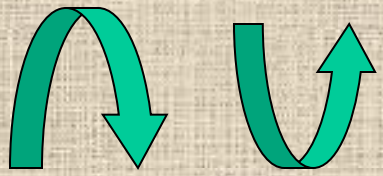
Leçon II:
Interaction avec l'environnement

- Le monde est son meilleur modèle:
- Il faut incarner la cognition dans les processus sensori-moteurs. Les concepts naissent à partir des interactions sensori-motrices et agissent en support de celles-ci.

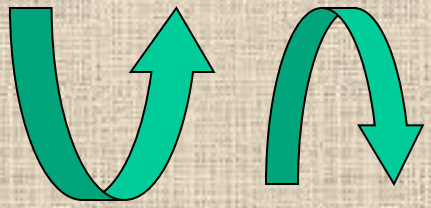
Leçon III:
apprentissage et adaptation

- Les systèmes doivent s'adapter de manière autonome:
  - Réseaux de neurones
  - Algorithme génétique

Emergence



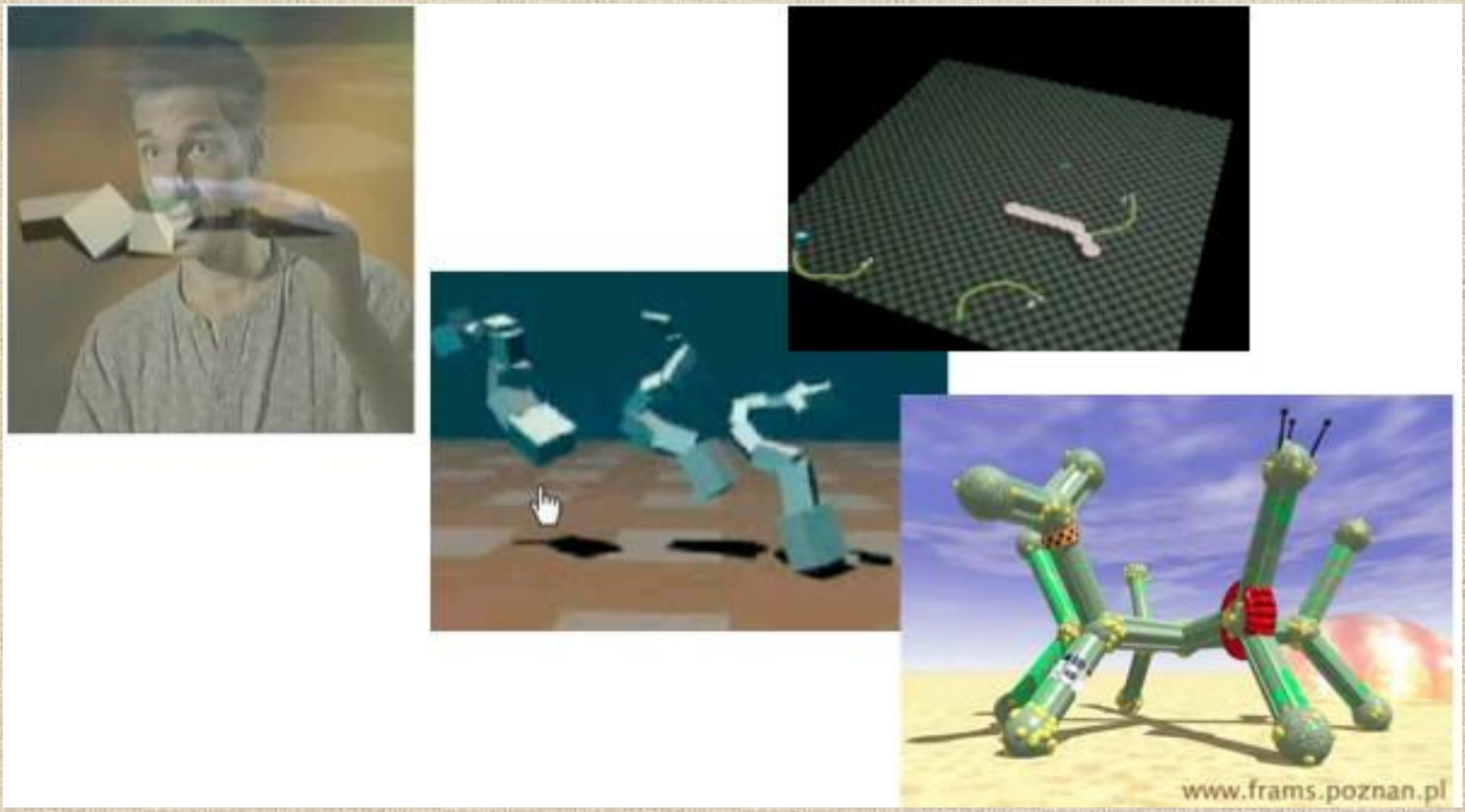
Interaction environnementale



Apprentissage

Les Créatures artificielles

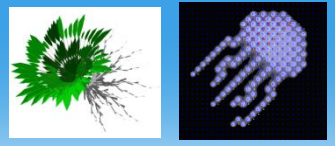
Contrôle de créatures artificielles



Les Travaux sur les Créatures Artificielle

1

Evolution de la morphologie



2

Evolution de la morphologie et du contrôleur



3

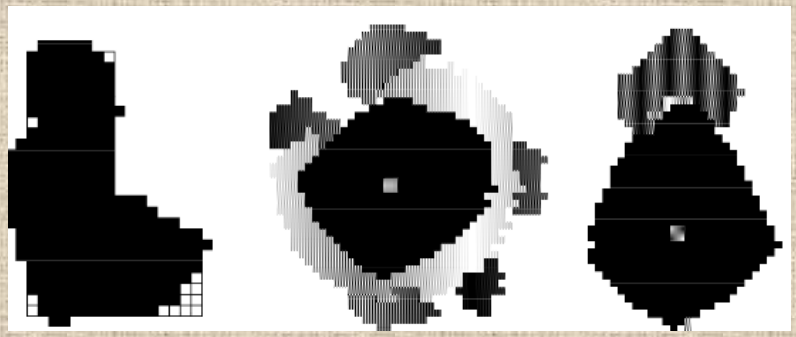
Evolution du contrôleur seul



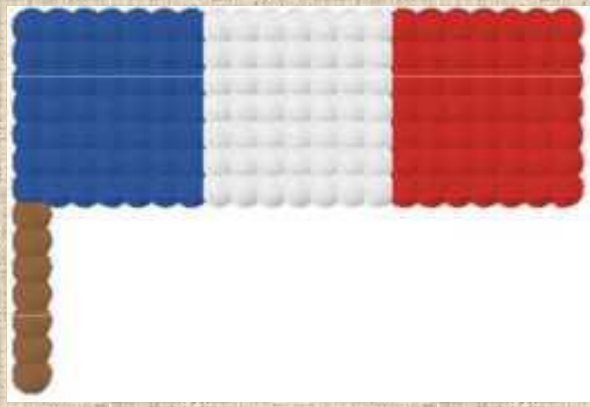
Evolution de la Morphologie

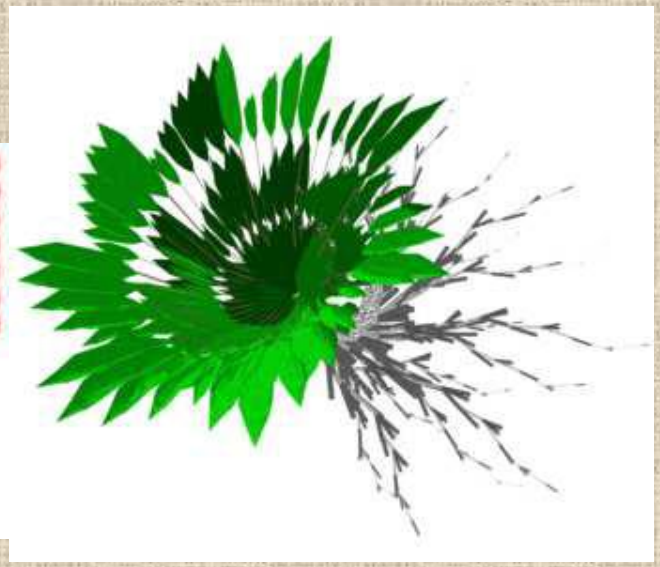




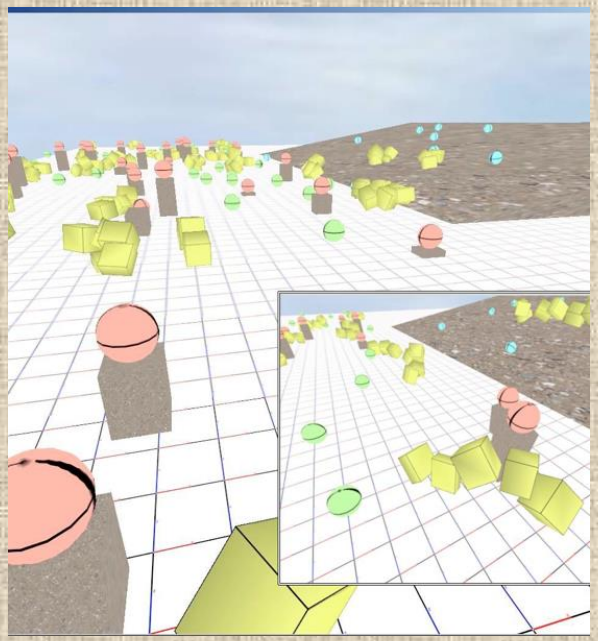






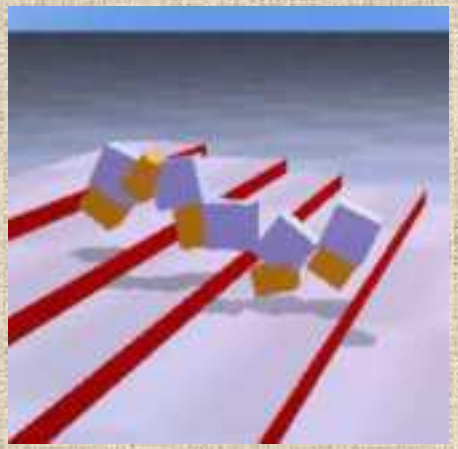


Co-évolution de la morphologie et du contrôleur

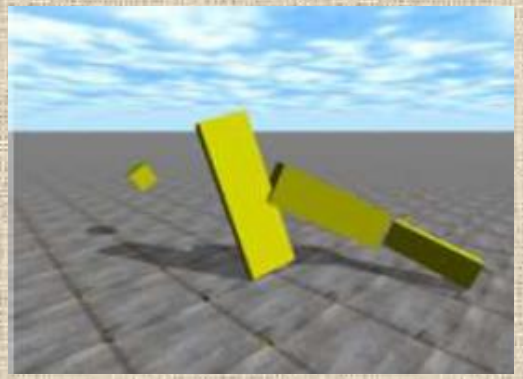


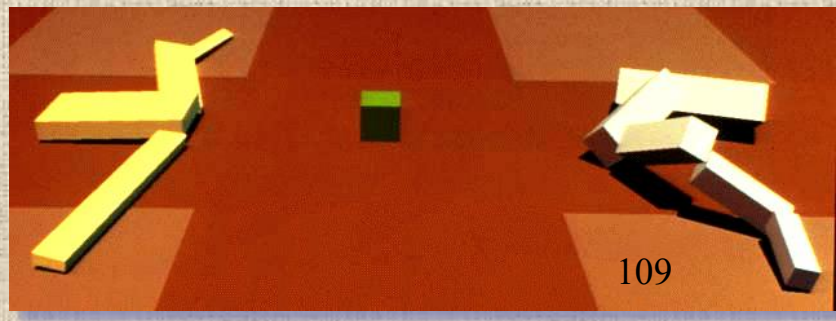








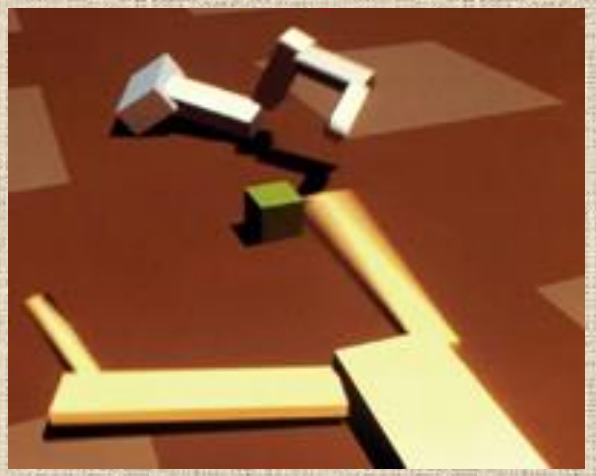




Exemples de Créatures

Evolved Virtual Creatures(Karl Sims)

- Sims (1994) : créatures cubiques générées par un réseau de neurone
  - Le réseau de neurones est la meilleure solution calculée avec un algorithme génétique
- Le système génère :
  - le comportement
  - la morphologie associée



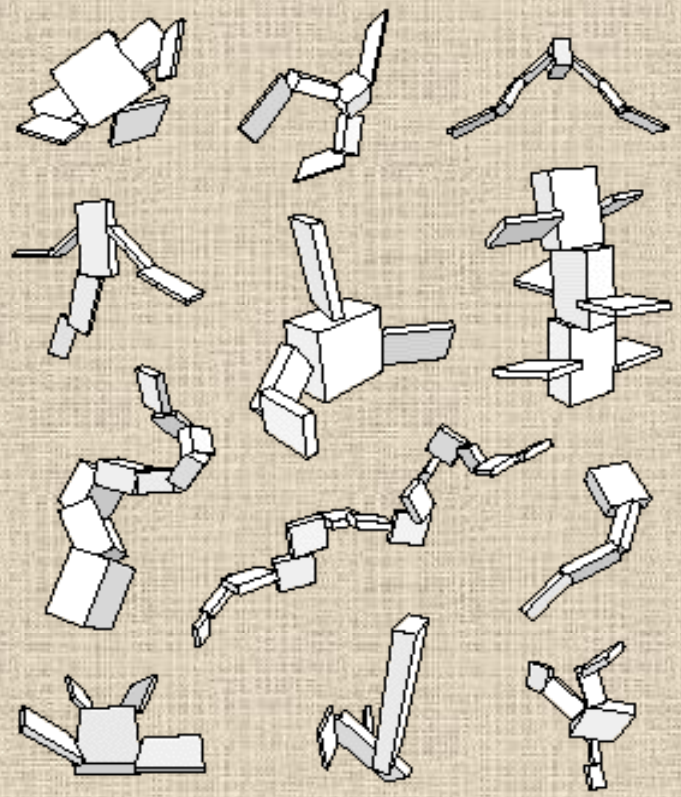


Evolved Virtual Creatures (KARL Sims)

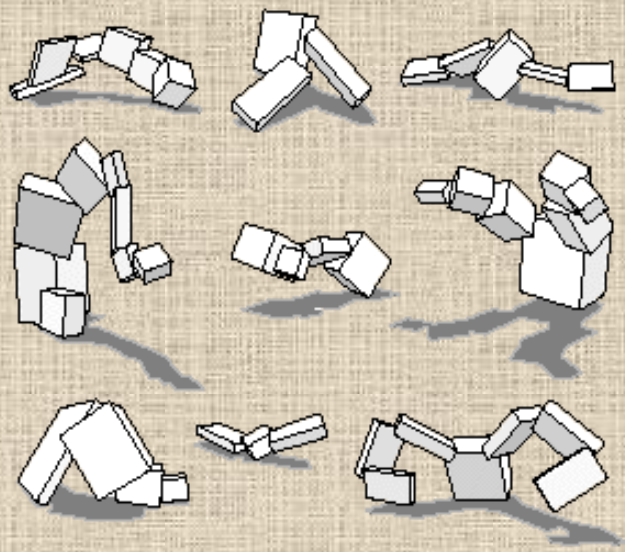
- Ses créatures virtuelles utilisent un réseau de neurones pour traiter l'information qu'elles reçoivent par le biais de capteurs et agir sur des muscles virtuels en fonction de leur environnement.
- Les créatures ont été développées pour montrer les modes de déplacement classique en fonction de leur milieu et de leur environnement :

  - comme la nage du serpent de mer ou du poisson,
  - le saut,
  - ou encore la chute
  - Poursuite d'une source de lumière
  - Compétions pour attraper un cube

Les créatures adaptées à la nage



Les créatures adaptés à la marche



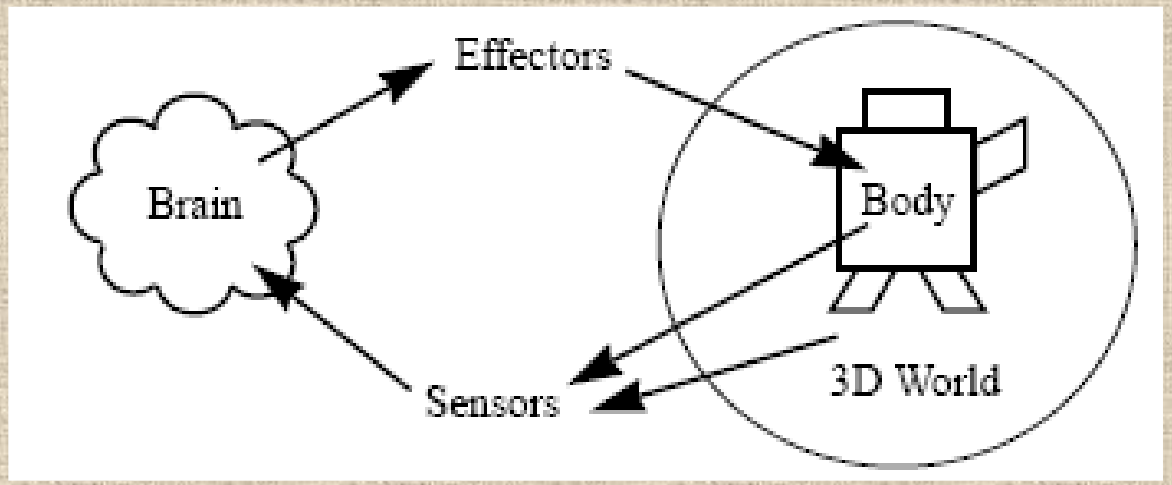
Les créatures de Karl Sims

Les créatures de Karl Sims sont considérées comme les plus évoluées, il a réalisé des créatures avec très peu de contraintes et a obtenu une grande diversité de créatures donnant naissance aux comportements de rampement et de natation (1994).

Le système de contrôle



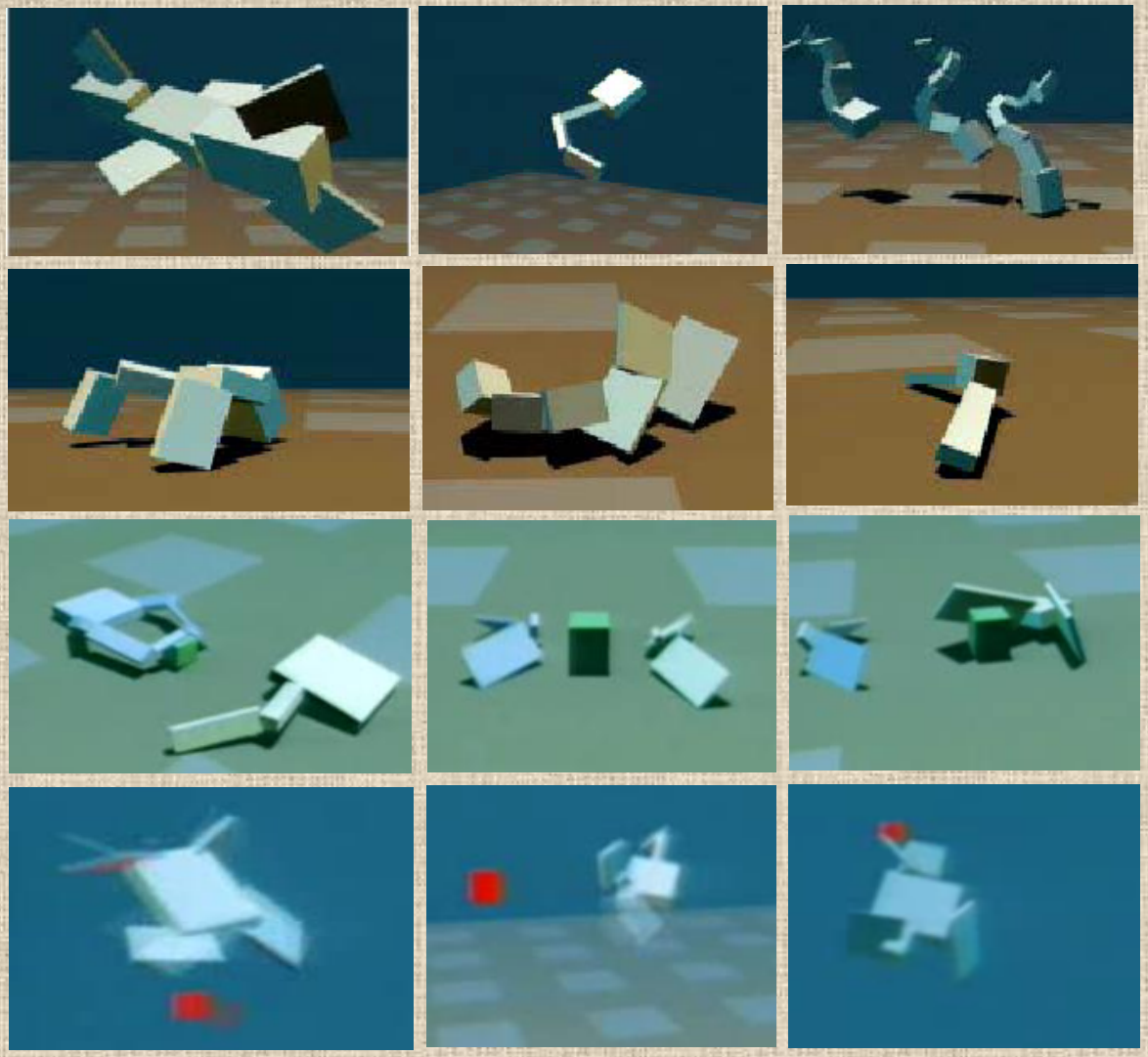
La simulation physique



Les créatures de Karl Sims

Evolved Virtual Creatures

Examples from work in progress



Karl Sims(1-Représentation Génétique en Détail)

Les créatures sont définies par un **génome modulaire**, qui encode à la fois leur **morphologie** (structure du corps) et leur **contrôleur neuronal** (comportement).
Le génome est souvent représenté sous forme d'un **arbre**, qui permet une description hiérarchique et récursive des structures corporelles.

a) Morphologie (Structure du Corps)
Chaque segment du corps est défini par :

- **Sa forme géométrique** : Cube, cylindre, sphère, etc.
- **Ses dimensions** : Longueur, largeur, hauteur.
- **Sa position relative** : Par rapport aux segments parents (structure hiérarchique).
- **Ses connexions** : Articulations qui relient chaque segment avec un ou plusieurs autres segments.

Karl Sims(1-Représentation Génétique en Détail)

Ces connexions permettent de générer des structures complexes comme :

- **Des créatures quadrupèdes** (similaires aux mammifères).
- **Des créatures serpentines** (avec des segments en chaîne).
- **Des créatures arboriformes** (organismes à structure ramifiée).

b) Articulations

Les segments sont reliés par des articulations, caractérisées par :

- **Leur type** :
  - Pivot (rotation autour d'un axe).
  - Sphérique (rotation dans plusieurs directions).
  - Articulation fixe (pas de mouvement).
- **Leur degré de liberté (DOF)** : Nombre de directions dans lesquelles l'articulation peut bouger.
- **Leur amplitude** : L'angle maximal que peut atteindre l'articulation.

Karl Sims(1-Représentation Génétique en Détail)

c) Contrôleur Neuronal

Chaque créature possède un réseau de neurones artificiels qui contrôle ses mouvements. Ce réseau est également codé génétiquement et définit :

- **Les entrées sensorielles** : Capteurs simulés (position, orientation, contact avec le sol, etc.).
- **Les sorties motrices** : Commandes envoyées aux articulations pour initier des mouvements.
- **Les connexions synaptiques** : Poids synaptiques qui déterminent comment les signaux se propagent à travers le réseau.

Karl Sims(2. Simulation Physique)

Une fois la créature générée, elle est placée dans un environnement simulé qui obéit aux lois de la physique. La simulation prend en compte :

- **La gravité** : Influence le poids et la chute des créatures.
- **La friction** : Influence la capacité des créatures à se déplacer sur différentes surfaces.
- **Les collisions** : Détection des interactions avec l'environnement et d'autres créatures.
- **Les forces appliquées** : Par les articulations, pour produire un mouvement.

Le moteur physique résout les équations du mouvement en temps réel, permettant d'observer comment chaque créature se déplace et interagit avec son environnement.

Karl Sims(3. Fitness : Évaluation des Performances)

Le système utilise un **critère de fitness** pour évaluer chaque créature. Ce critère dépend de la tâche que la créature doit accomplir.

Exemples :

- Se déplacer sur la plus grande distance possible.
- Atteindre un point cible.
- Nager dans un fluide simulé.
- Pousser ou tirer un objet.

La fitness est souvent mesurée par :

- La **distance parcourue** en un temps donné.
- La **stabilité du mouvement** (marche sans tomber).
- L'**efficacité énergétique** (minimisation de l'énergie dépensée).

Karl Sims(4. Algorithme Évolutif)

Une population initiale de créatures est générée aléatoirement, puis soumise à un processus évolutif inspiré de l'évolution biologique.

a) Sélection

Les créatures avec le score de fitness le plus élevé sont sélectionnées pour se reproduire.

b) Reproduction (Crossover)

Deux créatures parentales combinent leurs génomes pour produire une nouvelle créature (enfant). Ce processus implique :

- **Le croisement des structures morphologiques** (mélange des segments corporels des deux parents).
- **Le croisement des contrôleurs neuronaux**.

Karl Sims(4. Algorithme Évolutif)

c) Mutation

Des modifications aléatoires sont introduites dans le génome pour créer de nouvelles variations :

- Changer la taille d'un segment.
- Ajouter ou supprimer une articulation.
- Modifier un poids synaptique dans le réseau neuronal.

d) Boucle Évolutive

Le processus de sélection, croisement et mutation est répété sur plusieurs générations, permettant aux créatures d'évoluer vers des formes et comportements de plus en plus performants.

Karl Sims(5. Émergence de Comportements Complexes)

À travers plusieurs générations, des comportements inattendus et complexes émergent :

- **Marche bipède** avec coordination des jambes.
- **Nage ondulatoire** inspirée des poissons.
- **Sauts et roulades** pour contourner des obstacles.
- **Stratégies collaboratives** (lorsque plusieurs créatures interagissent).

Ces comportements ne sont pas programmés explicitement mais émergent de l'interaction entre la morphologie, le contrôle neuronal et l'environnement.

Karl Sims(6. Exemple de Créature et Comportement)

Imaginez une créature initialement aléatoire composée de trois cylindres reliés par des articulations simples.

- Au début, elle peut uniquement **osciller** sur place.
- Après plusieurs générations, elle évolue pour **balancer ses segments** de manière coordonnée, produisant un mouvement d'avance semblable à une chenille.
- Plus tard, elle optimise son réseau neuronal pour **sauter**, se stabiliser et continuer à avancer plus vite.

Karl Sims(7. Impact de la Simulation de Karl Sims)

L'impact des travaux de Karl Sims a été immense dans divers domaines :

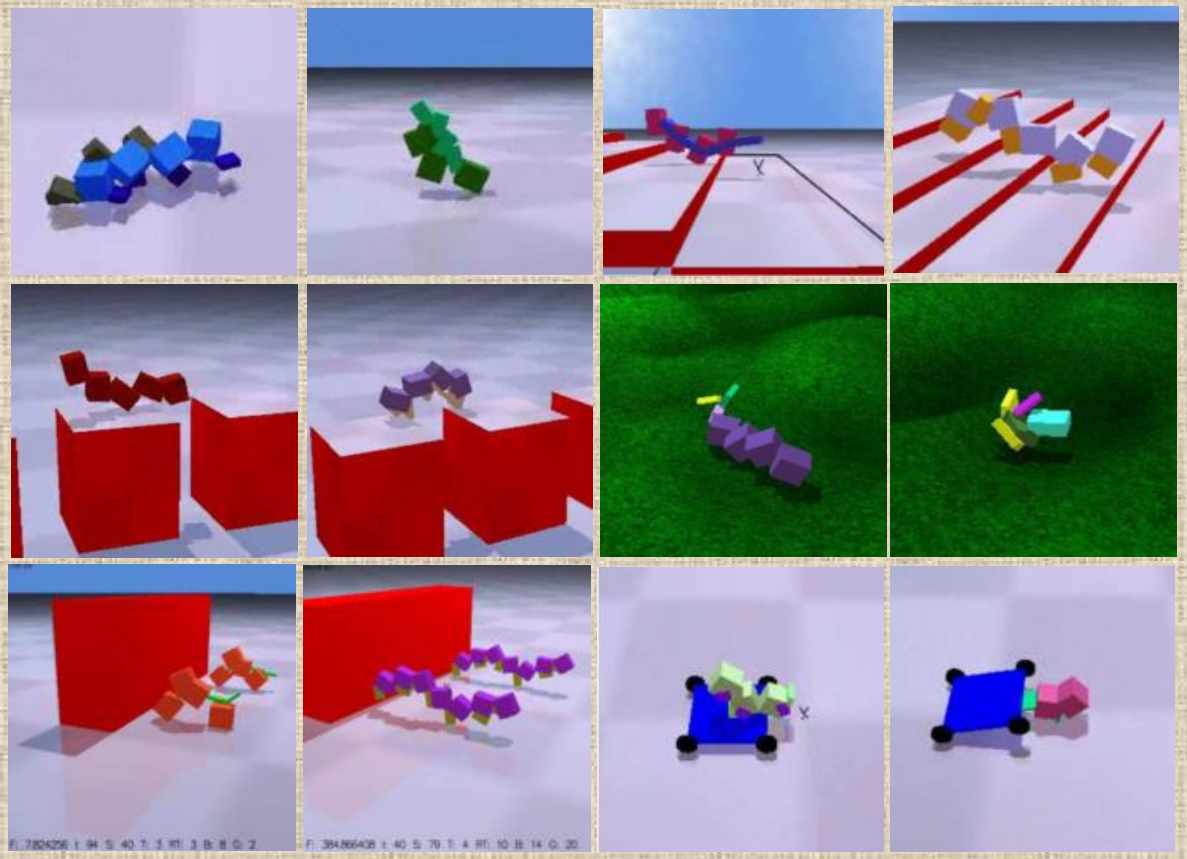
- **Intelligence Artificielle** : Utilisation de l'évolution artificielle pour résoudre des problèmes complexes.
- **Robotique Évolutive** : Conception automatique de robots capables de marcher ou de nager.
- **Biologie Computationnelle** : Étude des formes biologiques et de la locomotion.
- **Art Génératif** : Création d'animations et de formes organiques à partir de processus évolutifs.

Les créatures de Nicolas Lassabe

- Génération de Créatures artificielles capables d'interagir avec un environnement complexe par l'évolution artificielle (2008) (IRIT de Toulouse)

  Il a eu des brevets pour ses travaux

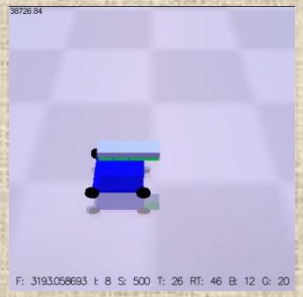




Les créatures de Nicolas Lassabe





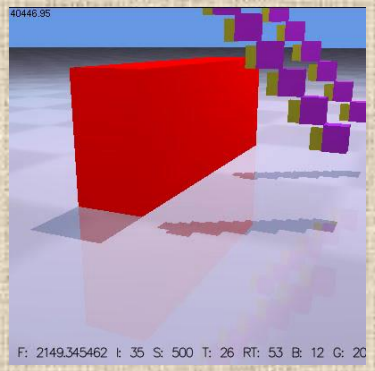


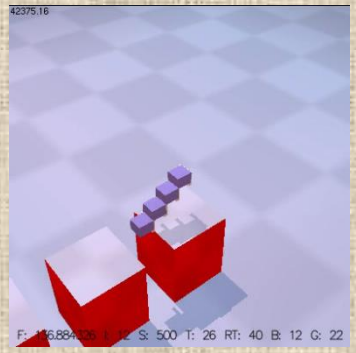


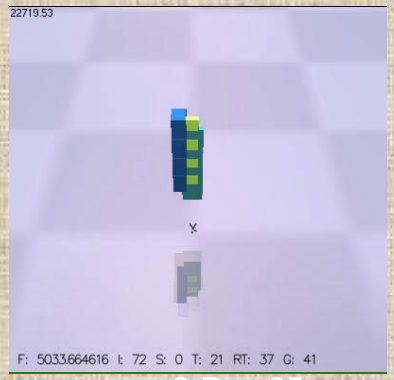


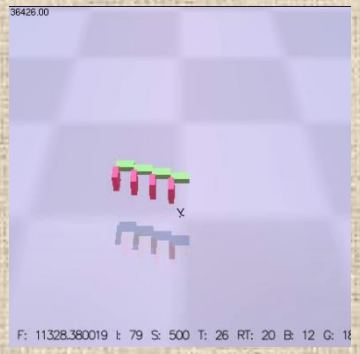


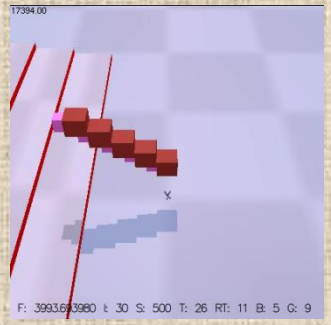


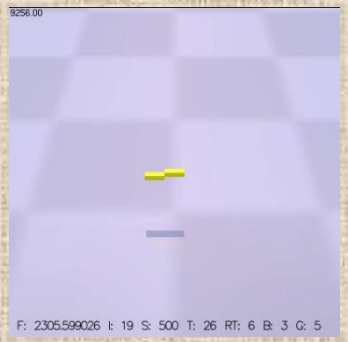


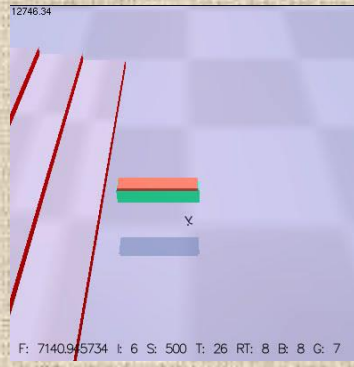












Les créatures de Nicolas Lassabe

Différences par Rapport à Karl Sims:

Karl Sims a introduit les concepts fondamentaux de la simulation des créatures évolutives, **Lassabe** a exploré des approches uniques, telles que :

- **Codage développemental plus poussé** : Incluant des algorithmes de morphogenèse avancés.
- **Réseaux complexes** : Des structures neuronales plus élaborées permettant une plus grande adaptabilité.
- **Environnements plus diversifiés** : Simuler des terrains ou des défis plus réalistes.

Les créatures de Nicolas Lassabe

Limites et Défis:

a) Complexité Computationnelle

- Les simulations nécessitent des ressources importantes pour modéliser l'interaction physique et la locomotion.
- L'évolution artificielle peut nécessiter des centaines ou des milliers de générations avant d'aboutir à des comportements viables.

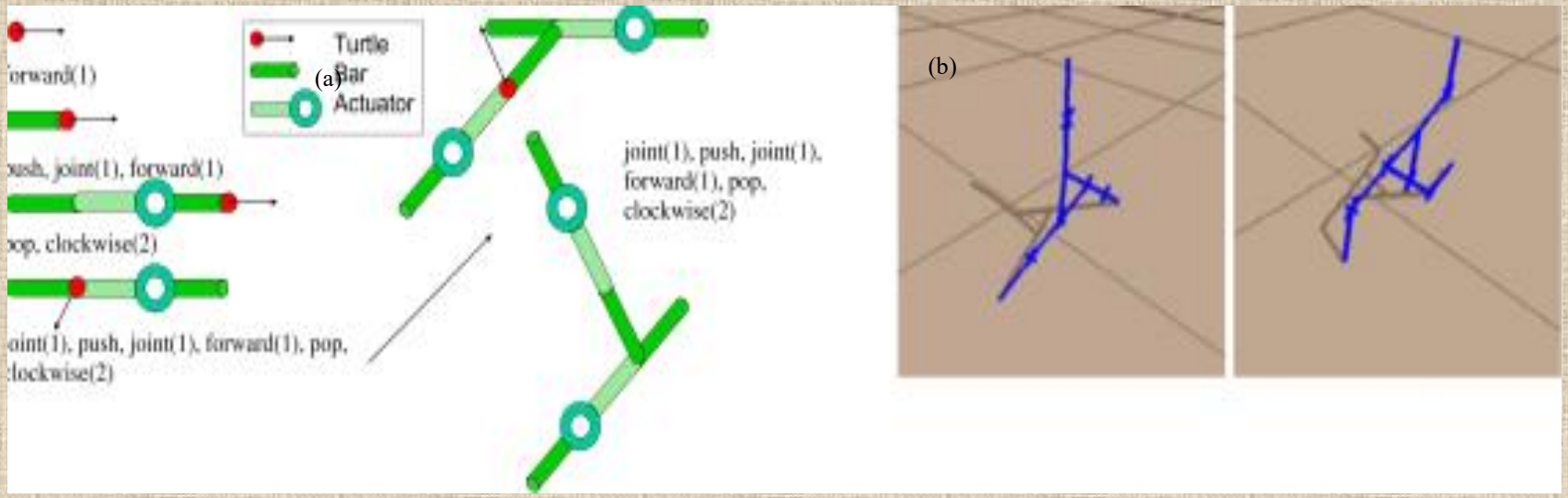
b) Contrôle du Processus

Les créatures générées peuvent parfois produire des comportements inattendus ou "stupides", comme des mouvements inefficaces mais qui satisfont les critères de fitness.

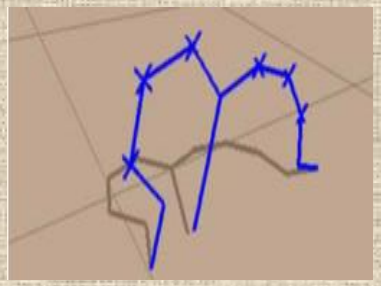
c) Généralisation

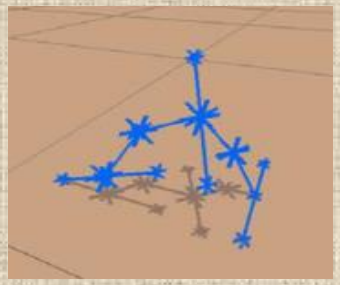
Une créature optimisée pour un environnement donné peut être totalement inadaptée si les conditions changent.

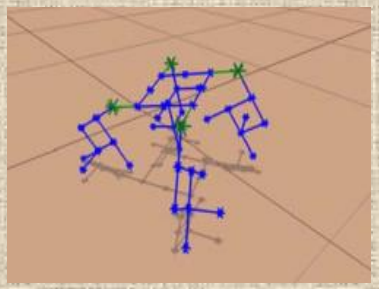
Evolution des L-systèmes pour générer des créatures artificielles



(c)







(d)







Evolution des L-systèmes pour générer des créatures artificielles

Les L-systèmes sont une méthode formelle de modélisation mathématique utilisée pour décrire la croissance des organismes biologiques.
Leur évolution pour générer des créatures artificielles repose sur une adaptation de ces systèmes pour créer des structures dynamiques et complexes, souvent en lien avec des algorithmes évolutionnaires.

Adaptation pour Générer des Créatures Artificielles:
Les L-systèmes sont particulièrement adaptés pour générer des formes organiques grâce à leur capacité à produire des structures fractales et symétriques.
Pour créer des créatures artificielles, plusieurs ajustements sont nécessaires.

Evolution des L-systèmes pour générer des créatures artificielles

Adaptation pour Générer des Créatures Artificielles:

a) Morphologie Générée par les L-systèmes

- Les symboles de l'alphabet peuvent représenter des segments corporels (membres, articulations, etc.).
- Les règles définissent comment ces segments se connectent et évoluent, formant un "squelette".
- L'interprétation géométrique donne une forme 3D à la créature.

b) Ajout de Comportements

En plus de la morphologie, on peut associer aux structures générées des comportements tels que :

- La locomotion (ex. : coordination des membres).
- Les interactions avec l'environnement.

Evolution des L-systèmes pour générer des créatures artificielles

Adaptation pour Générer des Créatures Artificielles:

c) Évolution des Règles

Les règles de production elles-mêmes peuvent évoluer via des **algorithmes évolutionnaires** :

- **Mutation** : Modification aléatoire d'une règle (par exemple, $F \rightarrow F+F-F$ devient $F \rightarrow F-F+F$).
- **Croisement** : Combinaison de règles entre deux individus.
- **Sélection :** Conservation des règles qui produisent des créatures efficaces selon un critère donné (comme se déplacer rapidement ou survivre dans un environnement).

Evolution des L-systèmes pour générer des créatures artificielles

Exemples de Créatures Générées:

Les créatures générées avec des L-systèmes peuvent inclure :

- Des **formes fractales** : Simulant des structures naturelles comme des coraux, des arbres ou des pattes articulées.
- Des **organismes locomoteurs** : Avec des membres dynamiques optimisés pour marcher, ramper ou nager.
- Des **créatures hybrides** : Combinaisons inattendues de segments corporels, produisant des structures biomimétiques uniques.

Evolution des L-systèmes pour générer des créatures artificielles

Avantages des L-systèmes dans ce Contexte:

- **Flexibilité** : Les règles simples permettent de générer une grande variété de formes complexes.
- **Évolutivité** : Les L-systèmes se prêtent naturellement à l'optimisation par des algorithmes évolutionnaires.
- **Biomimétisme** : Ils reproduisent des modèles observés dans la nature, rendant les structures générées crédibles et fonctionnelles.

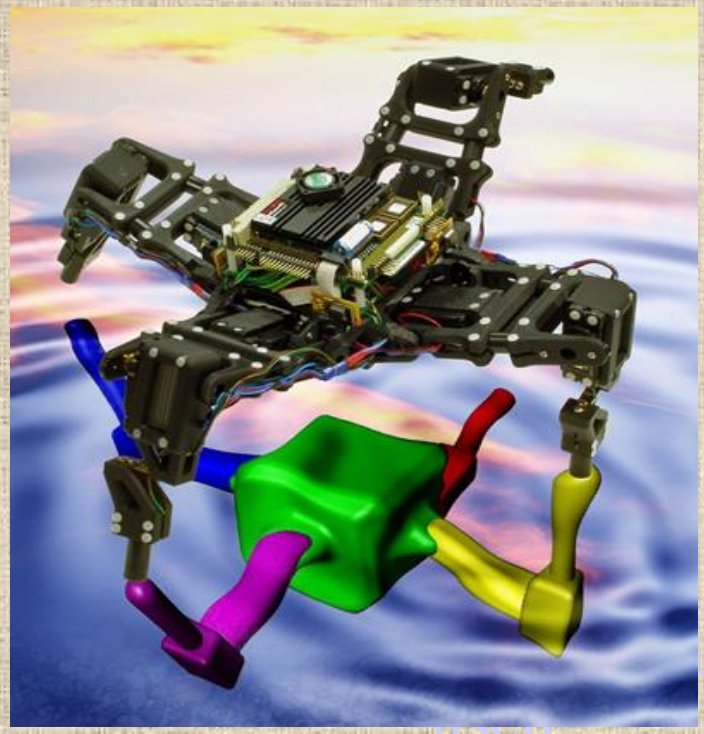
Limites et Défis:

- **Complexité croissante** : À mesure que les règles évoluent, la structure générée peut devenir chaotique ou inefficace.
- **Calcul intensif** : Les simulations nécessitent des ressources importantes.
- **Contrôle des comportements** : L'association de comportements réalistes à des structures complexes reste un défi.

Robots résilients

Le robot de Cornell est un robot innovant conçu pour s'auto-découvrir et s'adapter lorsqu'il subit des modifications, comme la perte d'un membre.
Ce projet est emblématique des avancées en robotique adaptative et en intelligence artificielle.





http://www.news.cornell.edu/stories/2006/11/cornell-robot-discovers-itself-and-adapts-injury

Robots résilients

Fonctionnement du Robot Cornell

1-Phase d'Auto-Découverte :

- Lorsque le robot est activé, il ne dispose pas d'une connaissance précise de son corps.
- À l'aide de capteurs et de simulations internes, le robot crée une **modélisation interne** de sa structure physique.
- Par des essais et des ajustements, il explore les relations entre ses composants (par exemple, articulations, moteurs).

2-Adaptation aux Modifications :

- Lorsqu'une perturbation survient (comme la perte d'une jambe), le robot détecte une divergence entre son **modèle interne** et son **état réel**.
- Il met à jour sa modélisation pour refléter les changements physiques, puis réajuste ses stratégies de locomotion en conséquence.

(Starfish robot introspects, modeling itself

http://www.news.cornell.edu/stories/2006/11/cornell-robot-discovers-itself-and-adapts-injury

Robots résilients

Fonctionnement du Robot Cornell

3-Algorithmes d'Apprentissage :

- Le robot utilise des **algorithmes évolutifs** et des **techniques d'apprentissage par renforcement**.
- Il génère et teste différentes solutions pour continuer à se déplacer efficacement malgré ses blessures.

4-Technologies Clés:

- **Capteurs intégrés** : Ils permettent au robot de mesurer ses mouvements, forces et positions pour détecter les anomalies.
- **Simulation interne** : Le robot exécute des milliers de simulations pour anticiper et tester des comportements alternatifs avant de les appliquer dans le monde réel.
- **Apprentissage évolutif** : Ce type d'apprentissage repose sur un processus de sélection naturelle, où les solutions les plus efficaces sont retenues et optimisées.

(Starfish robot introspects, modeling itself

Robots résilients

5-Applications:

1. **Robots d'intervention** : Les robots capables de s'adapter rapidement peuvent être utilisés dans des environnements hostiles (missions de secours, zones de guerre) où ils risquent d'être endommagés.
2. **Prothèses intelligentes** : Les principes de ce robot pourraient être appliqués pour concevoir des prothèses capables de s'adapter à l'utilisateur.
3. **Exploration spatiale** : Dans des missions spatiales, où les réparations humaines sont souvent impossibles, ces robots pourraient s'adapter aux imprévus.

6-Exemple Concret de Résilience:

1. Un robot quadrupède perd une patte.
2. Initialement, il boîte ou tombe.
3. Après quelques minutes, il découvre une nouvelle manière de se déplacer, en utilisant ses trois pattes restantes de manière optimisée.

(Starfish robot introspecta modeling itself

http://www.news.cornell.edu/stories/2006/11/cornell.robot.discovers.itself.and.adapts.injury

Robots résilients

7-Enjeux et Limites

1. **Consommation en calcul** : La simulation interne peut être coûteuse en termes de temps et de ressources.
2. **Robustesse des capteurs** : La fiabilité dépend fortement des capteurs, qui doivent être précis et durables.
3. **Complexité du modèle** : Plus le robot est complexe, plus la modélisation interne nécessite des ajustements sophistiqués.

(Starfish robot introspects, modeling itself

http://www.news.cornell.edu/stories/2006/11/cornell-robot-discovers-itself-and-adapts-injury

Self-Replicating and Self-Improving Robots

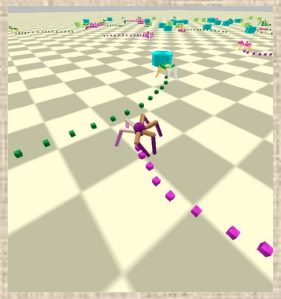
Les robots autoréplicants et auto-améliorants représentent un domaine avancé de la robotique, combinant les principes d'automatisation, d'intelligence artificielle (IA) et de biologie évolutive. Ces robots visent à imiter les processus biologiques de réplication et d'adaptation, leur permettant de se reproduire, de se réparer ou d'améliorer leur conception de manière autonome.

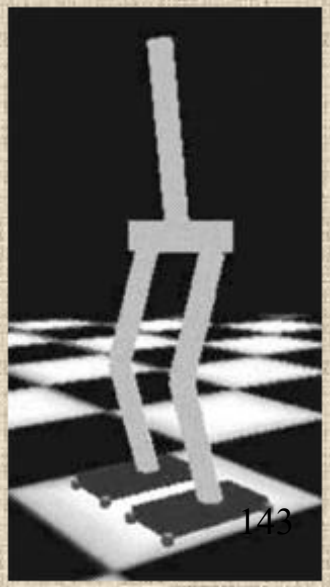
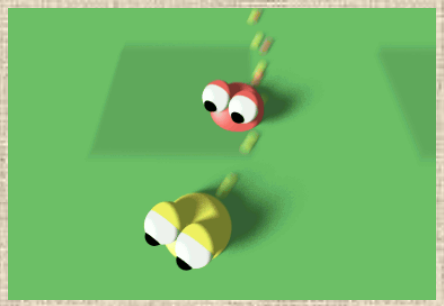
Un robot **auto-améliorant** est capable d'optimiser son comportement, sa structure ou ses composants au fil du temps grâce à l'apprentissage et l'adaptation.



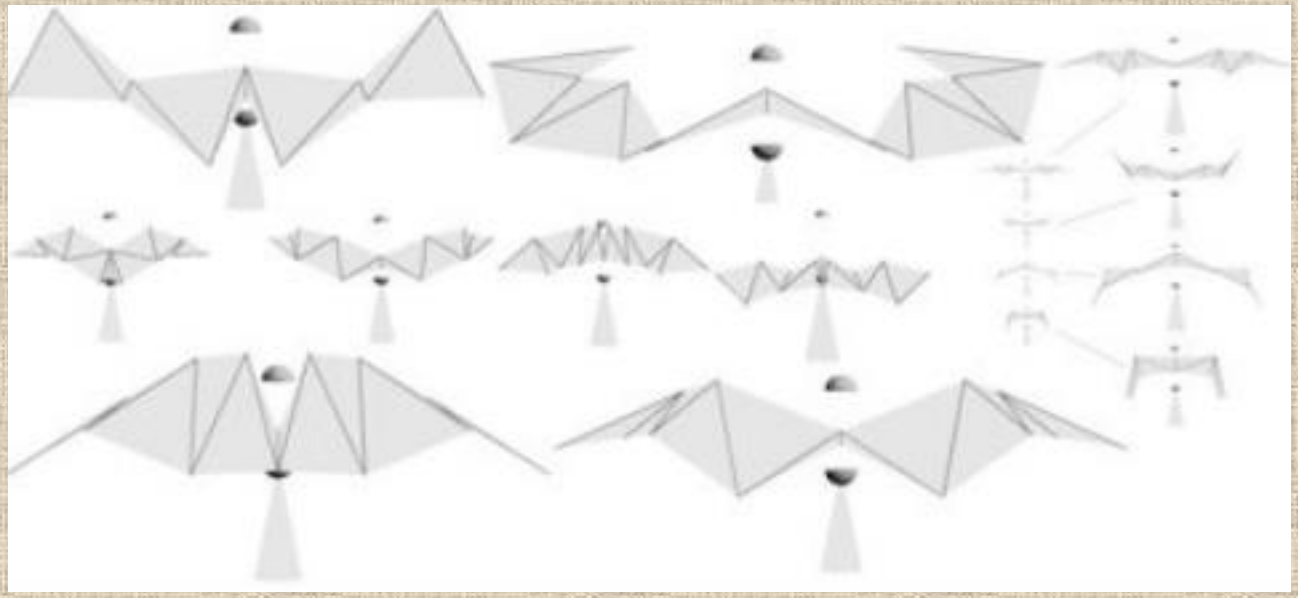
Evolution du Contrôleur Seul

L'évolution du contrôleur seul fait référence à un processus d'optimisation où **seules les règles** ou les mécanismes régissant le comportement d'un système ou d'un robot (son "contrôleur") sont modifiés. Ce type d'évolution est utile pour adapter un robot à son environnement ou à une tâche particulière, **sans changer sa morphologie** ou sa structure physique.




Créatures de Shim 2004

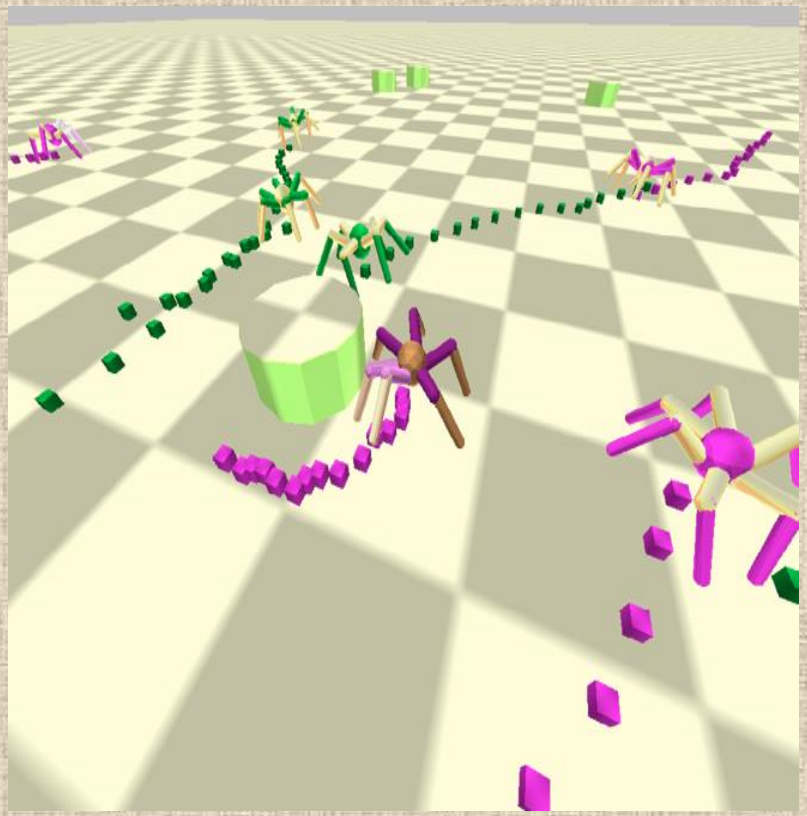


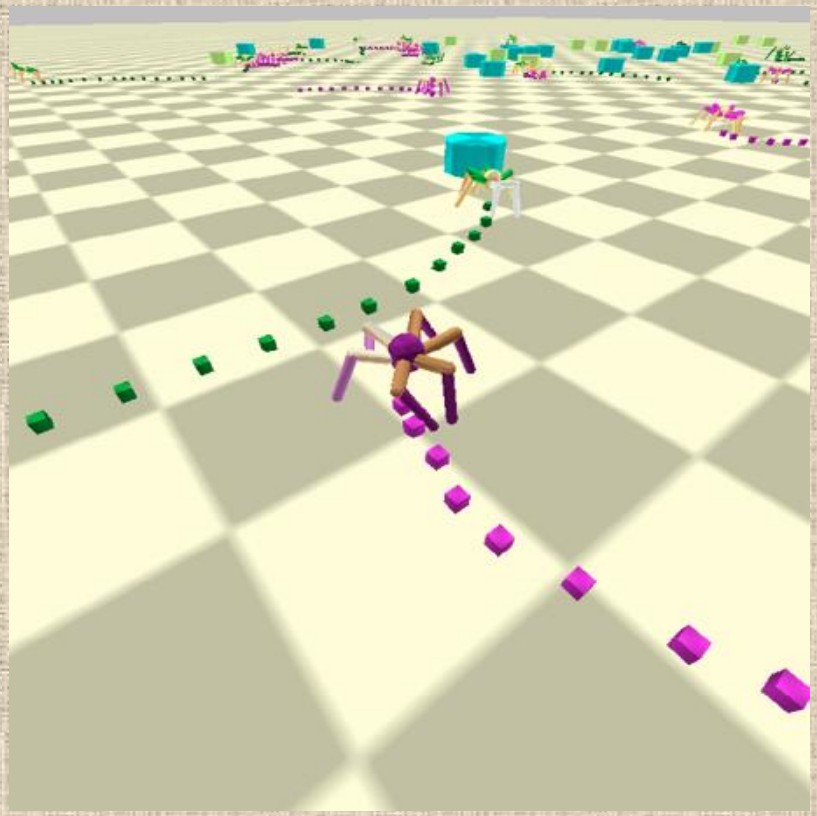
Ensemble des ailes générées par évolution



Les créatures de Shim évoluent pour adapter les ailes au vol

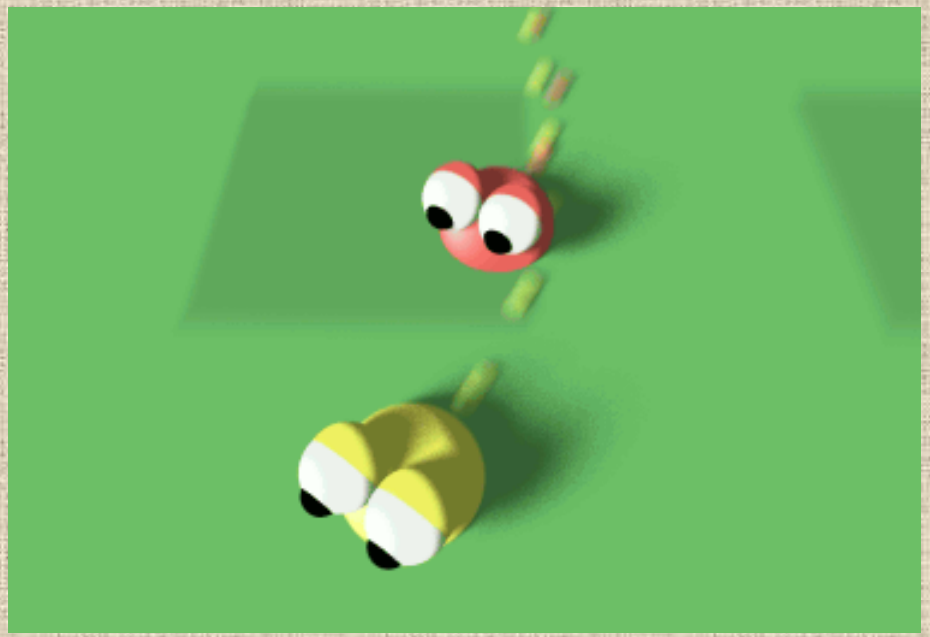
Michael Palmer Prédateurs-Proies





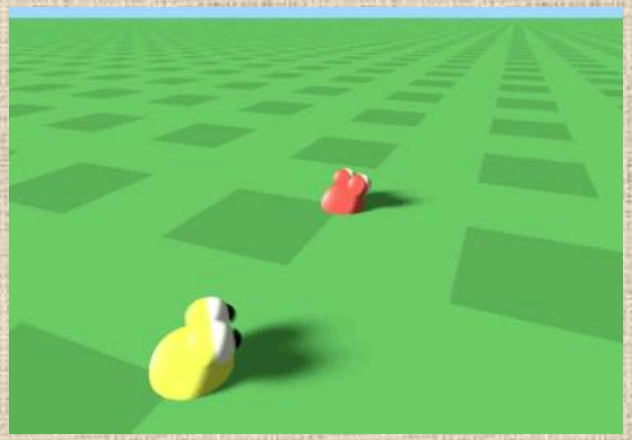
Poursuite de Cliff et Miller

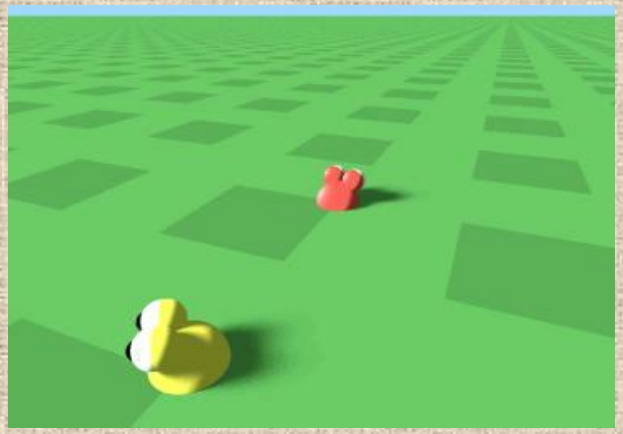
- Co-évolution de réseaux de neurones par A.G.
  - Poursuite / évasion
  - Modèle de prédateurs-proies
  - Optimisation du Comportement
- Gestion d'énergie pour le déplacement

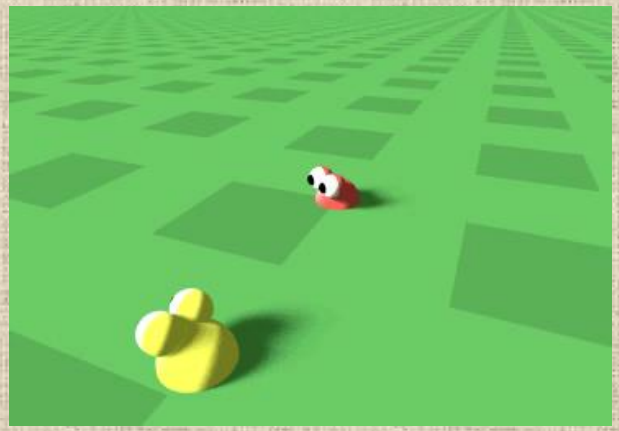


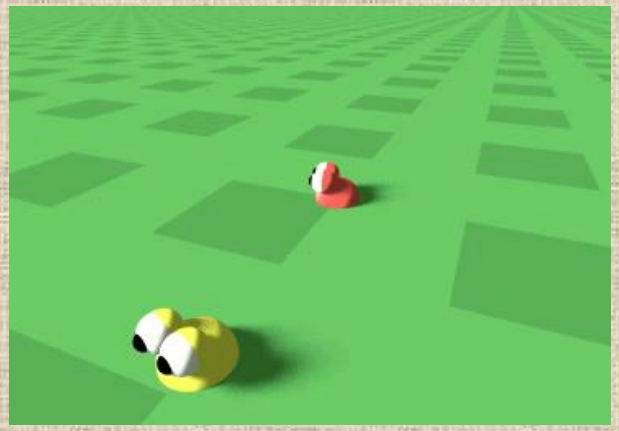
Poursuite Cliff et Miller

Génération simultanée du contrôleur et des capteurs (angles, position) Co-évolution de réseaux de neurones par A.G.





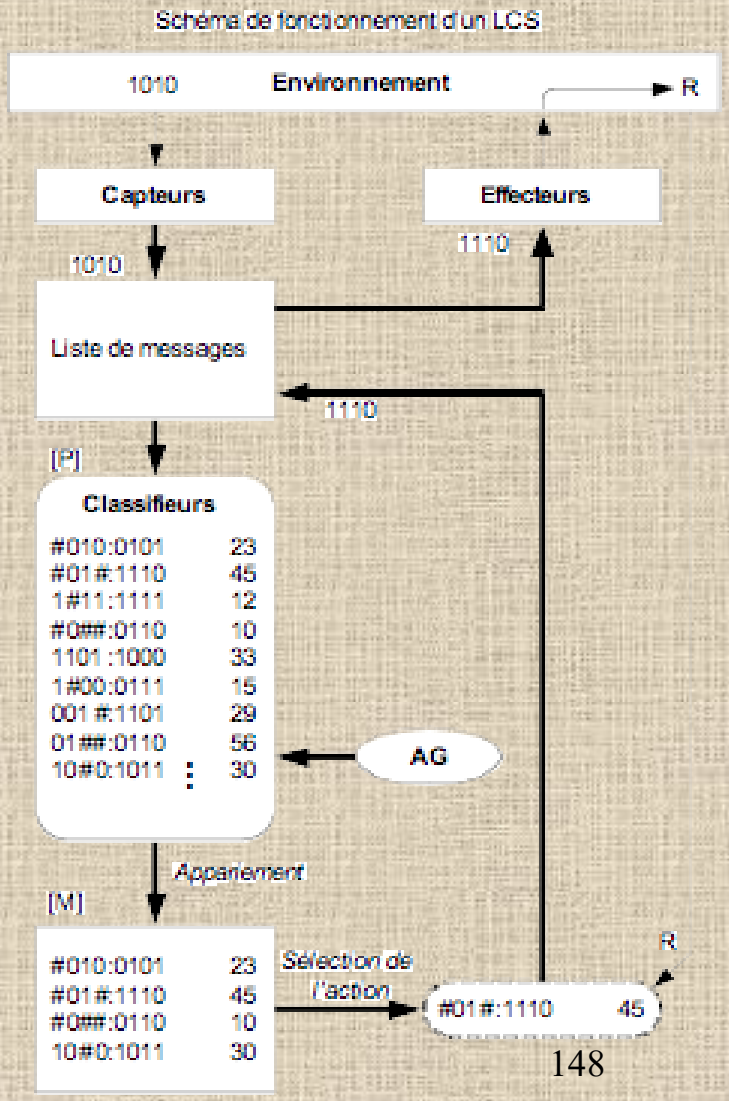




Évolution d'entités virtuelles coopératives par système de classeurs

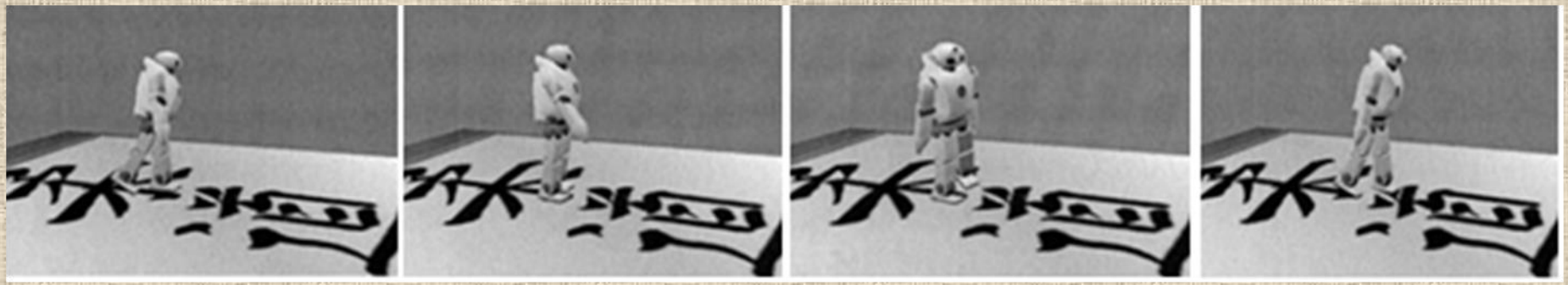


Sanza2001



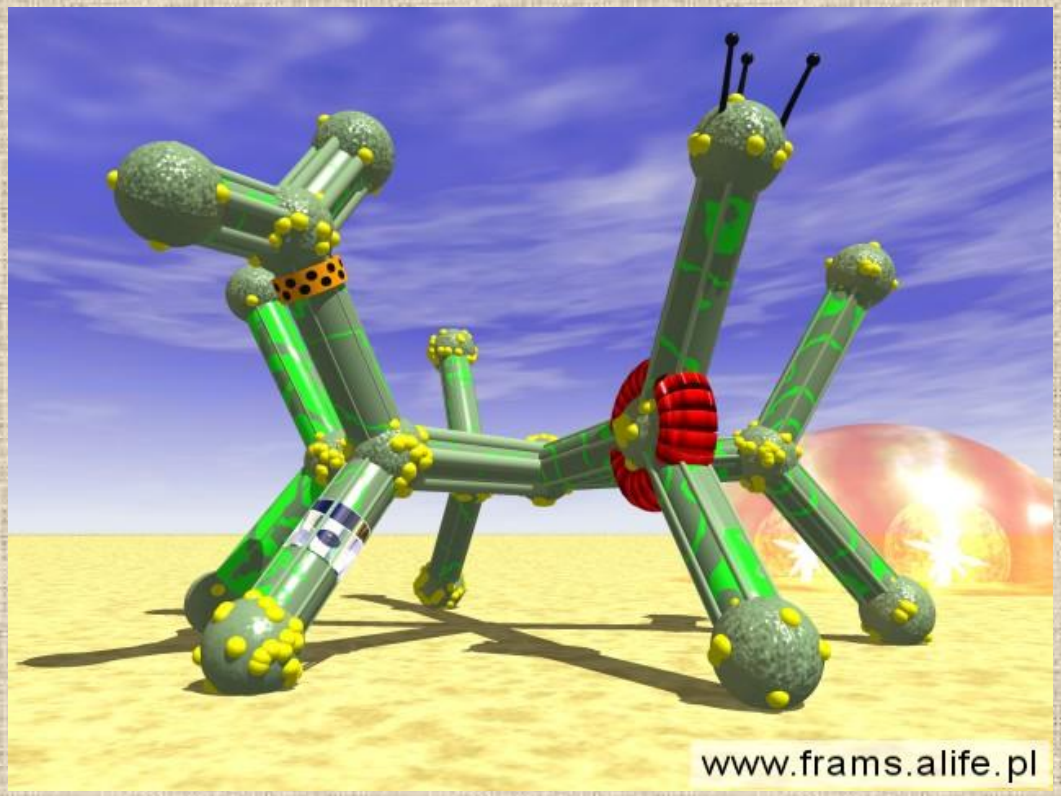
Les robots humanoïdes de Malachy Eaton

- **But:** La marche d'un robot humanoïde avec 20 DOF (2006).
- Simulateur de robots mobiles qui est Webots



[Commercial Mobile Robot Simulation Software.

FRAMSTICK



Une créature faite de bâtonnets, chacun spécialisé dans une fonction. Vous pouvez voir les 3 récepteurs : contact, odorat, équilibre, et un muscle en rouge.

Framsticks est un projet pour simuler la vie dans un espace tridimensionnel. La structure physique des créatures peut se transformer, et leur système de commande peut s'affiner. Pour cela, des algorithmes évolutionnaires sont utilisés pour la sélection, les croisements et les mutations des créatures.

FRAMSTICK



Une créature aquatique cherchant de l'énergie.

FRAMSTICK



Chasse : Une créature essayant d'attraper une autre, et de la tuer.

FRAMSTICK



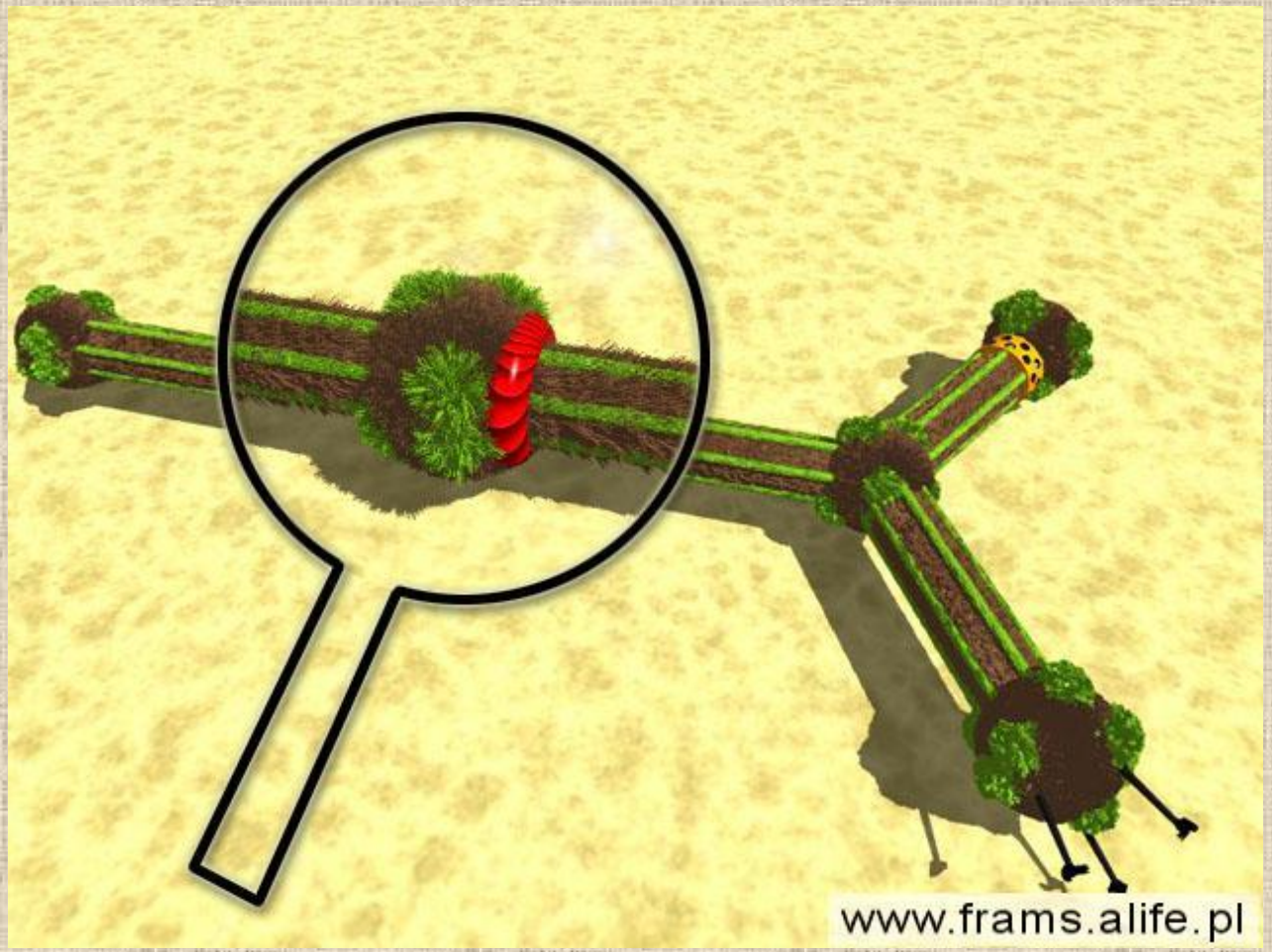
Pensiez-vous pouvoir construire des Framsticks avec des blocs ?

FRAMSTICK

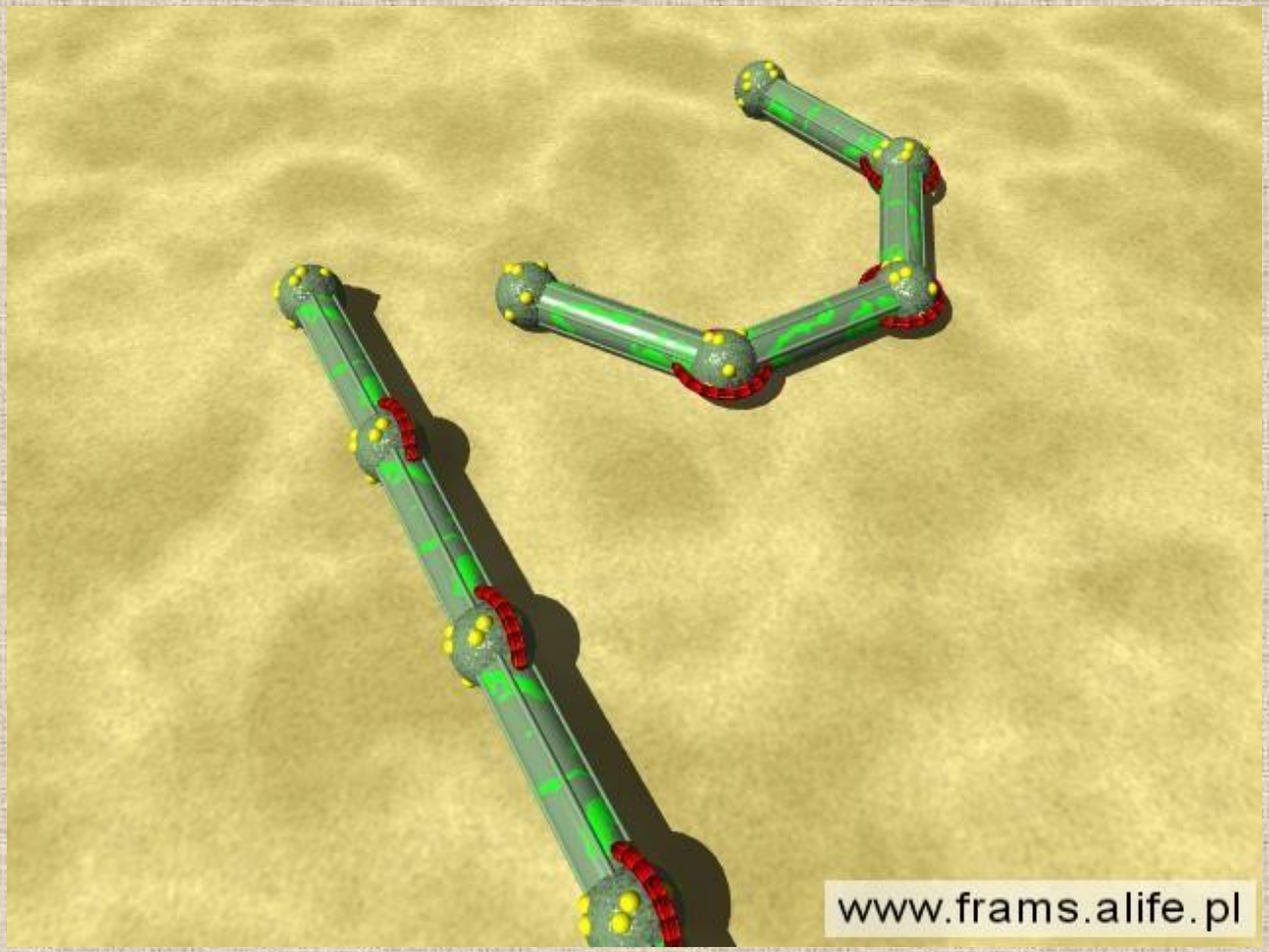


Quelques Framsticks encore en vie sur leur ancienne planète.

FRAMSTICK

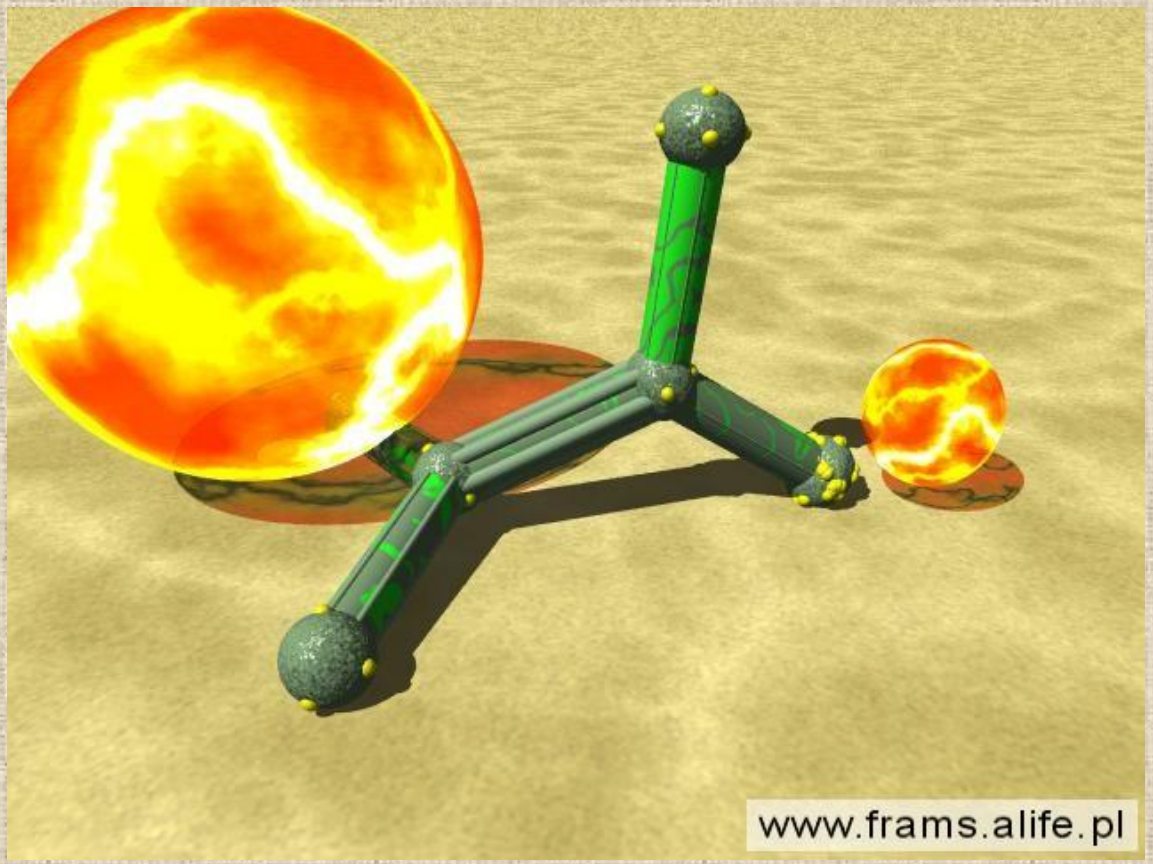


FRAMSTICK



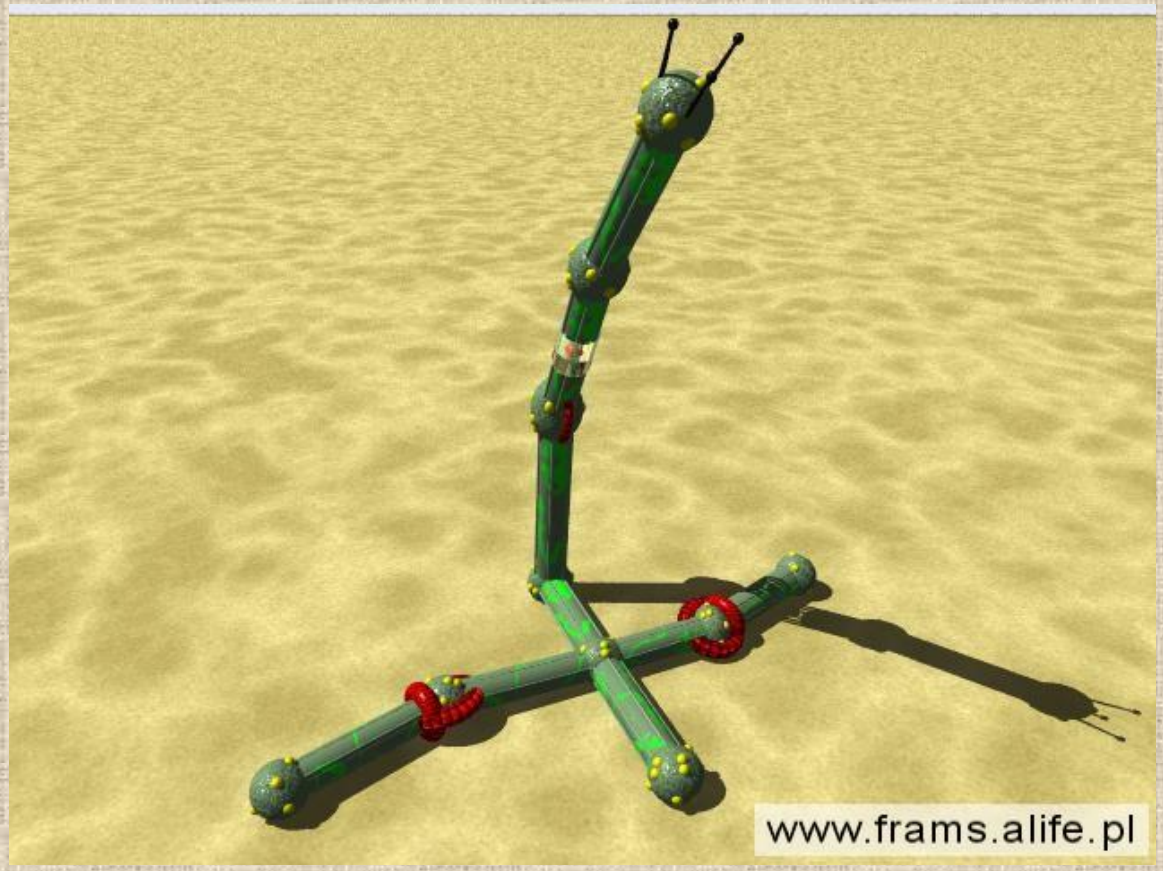
Les créatures sont faites de bâtonnets (membres). Les muscles (en rouge) sont contrôlés par un réseau de neurones, qui permettent des flexions et des rotations.

FRAMSTICK



Les bâtonnets peuvent ętre spécialisés dans des buts différents : Assimilation (en vert), force (épaisseur), ingestion (petit point jaune), etc...Les sphères d'énergie à droite sont assimilées rapidement grâce aux terminaisons spécialisées.

FRAMSTICK



Les créatures peuvent avoir différents récepteurs. Cette créature a un sens tactile (au-dessus), et un sens de l'équilibre (cellule transparente)

Conclusion

- La Vie Artificielle est un défi qui, nous force à réexaminer d'une nouvelle manière la nature et la vie.
- Bien que vie et intelligence artificielle soient deux notions distinctes (la vie n'est pas forcément "intelligente", l'intelligence n'est pas forcément "vivante"), elles sont fortement liées,