**CM1-2-3 Cognition artificielle**

**Introduction au connexionnisme**

= réseau de neurones artificielles

1. **Introduction**

* La cognition artificielle consiste en la création de programmation
* Le langage utilisé est Python, il a été créer pour faciliter la lecture des codes même s’il n’est pas facile à écrire.

En cas de besoin, lire : apprendre à programmer avec python 2 Gérard Swinnen (20premières/473), soit les deux premiers chapitres.

<http://inforef.be/swi/download/apprendre_python3_5.pdf>

L’intelligence « naturelle » par Wikipédia :

L’intelligence est l’ensemble des facultés mentales permettant de comprendre les choses et les fiats, de découvrir les relations entre elles et d’aboutir à la connaissance conceptuelle et rationnelle (par opposition a la sensation et à l’intuition). Elle permet de comprendre et de s’adapter à des situations nouvelle s et peut en ce sens être également définie comme la faculté d’adaptation. L’intelligence peut être également perçue comme la capacité à traiter l’information pour atteindre ses objectifs.

L’IA dans la littérature et au cinéma (et dans les séries) :

Quelques exemples d’IA dans la littérature et du cinéma.

Asimov : cerveau positronique, les lois de la robotique (pour le meilleur et le pire), etc.

Data : encore du positronique, guidé par la recherche d’une humanité (par exemple via les émotions).

L’agent smith : une IA purement logicielle, a priori sans émotion et avec un complexe de supéririté (puis une IA dépassée qui fiat de la résistance).

Tron : le simple programme, le MCP un programme qui a très « mal tourné » et Quorra une « IA » non programmé qui émerge de la complexité de son environnement.

Jane (l’IA amie de Ender) : émerge également de la complexité, reconnu par Ender comme étant une nouvelle forme de vie « consciente ».

Caprica (l’ado, pas celle avec la robe rouge) : les premiers pas du transhumanisme.

Terminator : le fantasme de l’IA qui prend le pouvoir (shynet) avec une ramée de robots plus ou moins évolués (dont un T2 doué d’une certaine capacité d’apprentissage)

Z-6PO (aka C-3PO) : ne comprend rien aux humains mais efficace dans son domaine d’expertise (traducteur)

Hall 9000 : pour moi... le danger de l’anthropomorphisme (des humains) dans la relation IA-humain.

Le WOPR : il apprend très bien, pour lui le nombre de morts n’est qu’une variable, mais heureusement il préfère les échecs à la guerre thermonucléaire globale.

Réaliste pour les années 2020 (2020-2030) :

* Hall 9000 : prévu pour tenir compagnie, assistant numérique, joue aux échecs, identifie les personnes, lit sur les lèvres, prend des décisions en conséquence.
* Z-6PO et D2R2 : comprend son environnement physique mais pas humain, capacités moteurs limitées.
* WOPR (et skynet ?) : de l’aide à la décision

Les autres : pour le moment encore du domaine de la SF. Le système à la base de règles sans explosion combinatoire ? cerveau positronique ? vie émergente ? singularité ? scan du cerveau humain ? et circuit suffisant pour un tel scan ?

1. **Intelligence artificielle symbolique et numérique**

Le connexionnisme est au centre de recherches pluridisciplinaires.

Les différents domaines s’intéressant aux réseaux de neurones artificiels (5) :

* **Biologie** : neuropsychologie et fonctionnement du cerveau, réseaux de neurones…
* **Statistiques** : alternative aux méthodes classiques
* **Alternative** **fondamentale** : mémoire associative, théorie des graphes, réseau d’automates
* **Informatique** **appliquée** : reconnaissance de formes (pattern) robotique, traitement de la parole… parallélisme massif, circuit VLSI, tolérance aux pannes, …
* **Sciences** **cognitives** : modélisation (mémoire, émotions, raisonnement …), validation d’hypothèses …

Les réseaux de neurones :

* La définition en **biologie** d’un réseau de neurones : ensemble de cellules vivantes mettant en œuvre une très grande quantité de neurones connectés par des synapses. Ils représentent le siège de la mémoire, de l’intelligence, de la conscience.
* Définition en **informatique** : ensemble d’automates élémentaires interconnectés qui s’influencent par l’intermédiaire de ces connexions
* Peut-être simulé sur un ordinateur !
  + Application : e.g. science de l’ingénieur
  + Recherche : e.g. test d’hypothèse biologiques

Comparaison IA symbolique/numérique :

L’intelligence artificiel comprend :

* L’IA **numérique** comprend les réseaux de neurones artificiels s’appuyant sur des exemples pour apprendre.
* L’IA **symbolique** comprend les « systèmes experts » nécessitant la création de règles qui provoqueront par la suite des faits etc..), « logique floue ».

|  |  |
| --- | --- |
| Intelligence artificielle  **« IA symbolique »** | Connexionnisme  **« IA numérique »** |
| * Ce construit sur une base de logique, c’est à dire que les données sont structurées * Le traitement séquentiel est plus maitrisé que lorsque c’est en parallèle. * Est localisé ce qui est rassurant * La programmation est connue | * Ce construit par analogie, c’est-à-dire que les données sont brutes * Le traitement se fait de manière parallèle, ce qui explique sa rapidité * Distribué 🡪 tolérance aux pannes * Il est relativement adaptable puisqu’il n’y a pas de développement (pas de programme) |

**En conclusion** : l’IA numérique a eu du mal à percer au vu des nombreux bénéfices qu’apportes l’IA symbolique.

Choisir entre IA symbolique et numérique

* IA **symbolique** à utiliser pour des problèmes de **« haut niveau »** (humain), lorsqu’on possède déjà :
  + Des variables clairement identifiées
  + Une très bonne expertise sur le domaine (les règles)
  + Par exemple : aide à la décision, diagnostique, modélisation de processus cognitifs de haut niveau
* IA **numérique** à utiliser pour les **données brutes**, éventuellement avec du bruit, peu structurées
  + Par exemple : classification, prévisions, modélisation de prétraitement intelligents

Les fondements biologiques du connexionnisme

* XVIIème siècle, cerveau se caractérisait comme un rôle central, comme organe de commande
* XIXème siècles est né le début de la neuropsychologie, un nouveau rapport est apparu entre « la matière » et la fonction, ce qui a permis de localiser et cartographier les fonctions cérébrales.

Ordres de grandeur comparés des réseaux de neurones :

* Système nerveux central : **1011**neurones

Un processeur récent : **106**transistors

* **1 mm3** de néo cortex :
  + **60K** de neurones
  + **3\*109** synapses
  + **1,5km** de dendrites
  + **2km** d’axone

**Rappel du fonctionnement d’un neurone biologique**

Le rôle du neurone biologique est de :

* **Recevoir** l’influx des autres neurones, via des dendrites
* **« Intégrer »** les influx reçus, dans le corps cellulaire
* **Engendrer** éventuellement un influx nerveux
* **Transmettre** l’influx à un autre neurone, via l’axone (fait de 1mm à 1m)

**Attention**: l’influx, le « potentiel d’action », est codé en fréquence, c’est-à-dire que tous les potentiels d’action on la même amplitude.

1. **Le neurone formel et le perceptron**

Le neurone formel de Mc Culloch et Pitts (1943)

Vecteur d’entrée **X** = (x1, x2, …, xn)

Vecteur poids **W** = (w1, w2, ..., wn)

La sortie dépend de la fonction d’activation

**Remarque** : on est passé à un codage en amplitude !

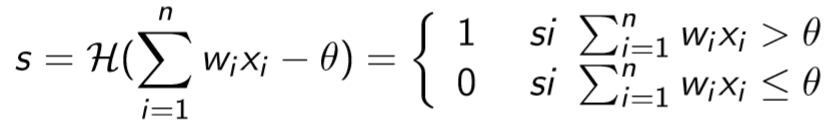
Exemple de « l’automate booléen à seuil »

Description de l’automate à seuil

* Vecteur d’entrée **booléen**
* Fonction d’activation de Heaviside (notée H)
* Sortie aussi **booléenne**
* Poids et le seuil réels

Fonctionnement :

* Calcul de la somme pondérée des entrées : **Σni = 1wixi**
* Calcul de la sortie de la cellule (S)



Présentation du perceptron de Rosenblatt (1958)

Déception du perceptron :

* Entrées **réelles**
* Fonction d’activation de Heaviside (notée H)
* Sortie encore **booléenne** (0 ou 1)
* Poids et le seuil réel

**Principe** : un perceptron, avec un neurone, divise l’espace d’entée IRnn en deux (et seulement 2) sous espaces séparées par un hyperplan d’équation :

W1\*x1 + w2\*x2 + wi\*xi + wn\*xn – θ =0

* On va pouvoir faire un choix => de la classification

Interprétation géométrique du perceptron

Si on se limite à IR2 : le perceptron divise l’espace d’entrée en 2 demi plan par une droite d’équation :

* Selon l’intensité de l’influx en entrée de la cellule, on considère un demi plan ou l’autre
* Le perceptron est capable d’effectuer une discrimination de l’espace d’entée en deux classes linéairement séparables.

Simulation des fonctions ET/OU et XOR

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 |  | ET | OU | XOR |
| 0 | 0 |  | 0 | 0 | 0 |
| 0 | 1 |  | 0 | 1 | 1 |
| 1 | 0 |  | 0 | 1 | 1 |
| 1 | 1 |  | 1 | 1 | 0 |

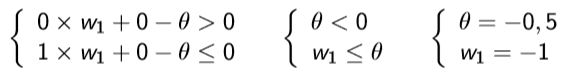
Le problème du « ou exclusif » (XOR), c’est qu’il est impossible de discriminer les valeurs de vérité du OU exclusif.

La limitation est significative car le pourcentage de fonctions booléennes linéairement séparables diminue très rapidement si on augmente la dimension de l’espace d’entrée (n) :

|  |  |  |
| --- | --- | --- |
| n | Séparable | Total |
| 2 | 14 | 16 |
| 3 | 104 | 256 |
| 4 | 1882 | 65536 |
| 5 | 94572 | 4.3 109 |
| 6 | 5028134 | 1.8 1019 |

Théorème de McCulloch et Pitts

Remarque : on peut simuler un NON logique



* On a les 3 opérateurs de la logique **booléenne**
* On peut synthétiser toutes les fonctions booléennes

Théorème : toute fonction booléenne de (n) variables peut être synthétisée par un réseau d’automates à seuil possédant (n) entrées

…mais encore faut-il savoir le construire.

1. **Apprentissage et généralisation**

Objectif : trouver les wi et θ automatiquement en présentant une base d’exemples au réseau

Il faut :

* Un algorithme d’apprentissage
* Une base d’exemples

= couples (vecteur d’entrée X, sortie désirée pour X)

Exemple : algorithme d’apprentissage du perceptron

Algorithme :

* On présente un exemple = on calcule la sortie du réseau
* Si la sortie ne correspond pas à la sortie désirée
  + Si on n’est pas du bon côté de l’hyperplan
  + On modifie les poids et le seuil
* On boucle

Remarques : l’algorithme converge toujours…si c’est linéairement séparable

La généralisation

Définition : la généralisation est la réponse du réseau sur des exemples qui n’étaient pas dans la base d’apprentissage

Objectifs :

* Connaître la réponse pour des exemples « inconnus » couples (vecteurs d’entrée X, ?)
* Evaluer la qualité de l’apprentissage couples (vecteur d’entrée X, sortie connue)

Performances et généralisation

Les performances en généralisation dépendent :

* De la **qualité** de l’algorithme **d’apprentissage**
* De **l’adéquation** du réseau (topologie, taille, dynamique)
* De la **qualité** des **exemples** utilisées pour l’apprentissage, il faut notamment une bonne distribution des exemples

Danger du sur-apprentissage

Dans une base d’apprentissage « réelle », on a souvent :

* Un **bruit** important
* Quelques exemples **non** **représentatifs**
* Eventuellement des **erreurs** d’étiquetage …
* Pour certains problèmes il est inutile de chercher à apprendre tous les exemples
* Arrêt de l’apprentissage une fois un certain pourcentage de la base d’exemples apprise (« taux d’apprentissage »)
* Éviter un « sur apprentissage », en apprenant trop « par cœur » on généralise moins bien

The credit assigment problem

On a vu que l’on peut **synthétiser** toute fonction booléenne en **utilisant** plusieurs **couches** de **perceptron**

Mais on n’a pas la « sortie désirée » des couches non terminales

* Comment apprendre ?
* « credit assigment problem »

C’est un problème mit en évidence par Minsky et Papert en 1969, il est resté ouvert pendant 20 ans.

**La solution trouvée** : un algorithme de la rétro propagation du gradient.

Résumé chronologique des débuts du connexionnisme

* **Années 40** : **Van Neumann, Mc Culloch, Ruring**
* Fondement d’une science des systèmes capables **d’auto-organisation** (la cybernétique)
* **Années 60** : **Rosenblatt** reprend les automates de **Mc Culloch et Pitts** pour faire la reconnaissance de formes
* **Années 70** : **Minsky et Papert** mettent en évidence les limites du perceptron
* Les années de l’IA symbolique (gros travaux). On veut simuler directement le raisonnement grâce a de la logique / des inférences (systèmes experts, logique floue).
* **Problème** : explosion combinatoire...
* **1986**: naissance de l’algorithme de la rétropropagation du gradient

1. **Topologie des réseaux de neurones**

**Les trois grandes familles :**

La capacité du réseau à résoudre le problème va dépendre de :

* L’algorithme **d’apprentissage**...
* Des **exemples** à apprendre
* De **l’architecture** du réseau :
  + **Topologie** du réseau
  + **Dynamique** du réseau (propagation de l’inﬂux)
  + **Taille** du réseau

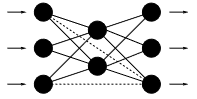
Il existe 3 grandes familles de réseaux :

* Les réseaux à **couches**
* Les réseaux à **connexions latérales**
* Les réseaux **récurrents**

1. **Les réseaux à couches**

**Déﬁnition** : un réseau à couches se caractérise comme un « graphe acyclique de cellules »

On peut déﬁnir une ou plusieurs :

* Cellules **d’entrée**
* Cellules de **sortie**

**Remarques** :

* Les connexions sont **orientées** (entrée → sortie)
* Le nombre de **couches** correspond au nombre de **cellules** entre les cellules d’entrée et les cellules de sortie
* De loin les réseaux les **plus** **utilisés**

1. **Les réseaux à connexions latérales**

Le réseau à connexions latérales est un réseau a couches comprenant des connexions entre cellules d’une même couche.

La dynamique, pour chaque couche de l’entrée vers la sortie :

1. **Activation** en fonction de la **couche précédente**, via les **connexions inter-couches exclusivement**
2. **Activation** en fonction du **nouvel état de la couche courante**, via les connexions **intra-couche exclusivement**

Remarques :

* Généralement **peu** de **couches**
* Généralement **pas** de **connexions** **latérales** sur chaque couche

**Exemple** : les cartes de Kohonen (on verra plus tard...)

1. **Les réseaux récurrents**

**Déﬁnition** : les réseaux récurrents se définissent comme un « graphe avec une ou plusieurs boucles ».

**Les particularités** : **partiellement** ou **totalement** **interconnecté** avec des connexions orientées, éventuellement dans les 2 sens

**Remarques** : L’état **d’activation de l’ensemble** des cellules à un instant (t) forme une **« conurbation d’états ».** Il existe des configurations d’états dites **« stables ».**

La propagation de l’influx dans un réseau récurrent

Le « Feedback neural networks » se définit comme une « alimenté en retour ».

Dynamique du réseau :

* **L’exemple à apprendre** (en apprentissage) ou la **forme à reconnaître** (en généralisation) est **présentée** au réseau, certaines (voire toutes) les cellules sont **« forcées »**
* Pour chaque cellule : **l’influx** se **propage** d’une cellule à l’autre, on active toutes les cellules ce qui forme une **nouvelle « configuration** **d’états »**
* Si une configuration **stable** est **atteinte**, la réponse du réseau peut être **lue** sur certaines (voire toutes) les **cellules**

L’ordre d’activation des cellules :

En **« feedforward »** **l’ordre** d’activation n’a **pas** **d’importance** : il n’a **pas** **d’influence** sur le comportement / la **sortie** du réseau mais ce n’est pas le cas avec une dynamique **« feedback ».**

**Important** : dans un réseau **récurrent** **l’ordre** d’activation des cellules **influe** sur l’évolution des **configurations** d’états du réseau, et donc sur l’éventuelle **réponse** du réseau

L’activation peut être :

* **Synchrone** : toutes les cellules s’activent « simultanément » sur la base de l’état d’activation précédent
* **Asynchrone** : les cellules s’activent les unes après les autres, dans un ordre donné (fixé) ou aléatoire, sur la base de l’état d’activation courant

Lecture de la réponse (la sortie) d’un réseau récurrent

Dans un réseau à couches : la réponse est lue sur la **dernière** **couche** après un temps **constant** et **connu**

Dans un réseau récurrent :

* Le réseau peut **converger** vers une **configuration** d’états stable, propager de nouveau l’influx ne change pas la conﬁguration puisque la réponse du réseau est la conﬁguration stable trouvée.
* Le réseau peut **converger** vers un **cycle**, voire ne pas converger, dans ce cas il n’y a pas de réponse.

**Remarque** : le temps de réponse est variable (stabilisation), le temps réel est délicat à mettre en œuvre

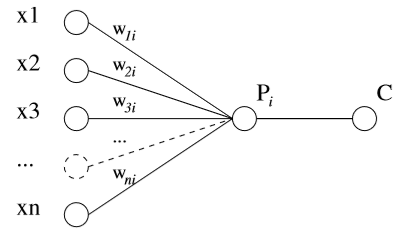
Principe de l’apprentissage d’un réseau récurrent

**Principe**: on utilise les états **stables** du réseau récurrent pour **mémoriser** les exemples à apprendre. On va modifier les poids et les seuils dans le but que le réseau converge vers des exemples qu’on apprend, que les exemples correspondent à des états stables du réseau (sur une forme apprise à partir d’une forme), elle peut être :

* Incomplète
* Complète mais bruitée
* Incomplète et bruitée

**Exemple de réseau récurrent** : les réseaux de Hopﬁeld (on verra plus tard, t’es bientôt ;-)

1. **Les modèles à base de prototypes**

Un prototype correspond aux cellules dont les connexions entrantes, pondérées par des réels, stockent un exemple représentatif d’une classe.

La fonction d’activation évalue la ressemblance entre :

* Un **exemple** en entrée X = (x1,x2…xn)
* Le **prototype** : Pi = (W1i,w2i…wni)

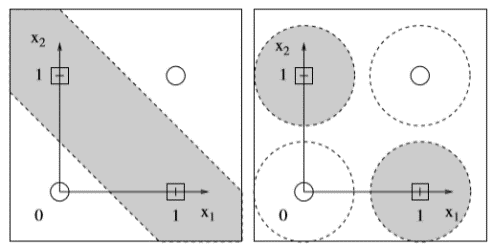
Le choix de la mesure est lié à la nature de l’espace d’entrée

Hypersphère contre hyperplans

Un prototype est activé si l’entrée lui ressemble, si l’exemple est dans la zone d’influence définie par ses poids.

Le prototype sépare l’espace en deux sous espace :

* À **l’intérieur** de l’hypersphère
* À **l’extérieur** de l’hypersphère



**Exemple** sur le problème du XOR

Prototypes et localité

Dans un réseau à base de prototypes, « la connaissance » stockée est :

* **Localisée**, c’est-à-dire qu’elle est facile à trouver
* **Cohérente** avec l’entrée, c’est-à-dire qu’elle est facile à « visualiser »
* On peut analyser la connaissance du réseau, le pouvoir explicatif (limité)

Apprentissage pour un réseau à base de prototypes

Pour apprendre, on peut :

* **Changer** les **poids**, c’est-à-dire déplacer le centre de l’hypersphère
* **Changer** les **seuils** c’est-à-dire changer le rayon de l’hypersphère
* **Ajouter** des **prototypes**, on parle de « réseau incrémental »
* **Supprimer** des **prototypes**, on parle « d’étalage »

Les principaux modèles à base de prototypes :

* Les cartes de Kohonen (1982-84)
* Les algorithmes LVQ de Kohonen (1988-90)
* Le néocognitron Fukushima (1980)
* Le modèle ART de Carpenter et Grossberg (1987-88)