

Méthodes numériques d'intelligence artificielle

Boigelot Denis

26 mai 2006

Sommaire

1	Introduction	2
1.1	Test de Turing	2
1.2	Bref historique	2
1.3	Vie artificielle	3
1.4	Apport de l'informatique	3
2	Réseaux de neurones	5
2.1	Perceptron	5
2.2	Mémoires associatives	6
2.3	Réseaux à couches cachées	7
3	Data mining	8
3.1	Clustering	8
3.2	Arbres de décisions	8
3.3	Logique floue	9
3.4	Lazy	9
3.5	BagFS	10
4	Apprentissage par renforcement	11
4.1	Dilemme exploration-exploitation	11
4.2	Q-learning	11
4.3	Q(λ)-learning	12
5	Stochastique locale search	13
6	Intelligence collective	14
6.1	Fourmis	14
6.2	Foraging	14
6.3	Swarp-bot	15
7	Réseaux	16
7.1	Effets dynamiques	17
7.2	Réseaux et IRIDIA	18
8	Algorithmes génétiques	19
9	Compléments	20
9.1	Applets	20
9.2	Coup de gueule de Bersini	20
9.3	Vidéo	20
9.4	Lectures conseillées	21

Chapitre 1

Introduction

« Demander si un ordinateur peut penser revient à demander si un sous-marin peut nager. »
Edsger Dijkstra (1930–2002)

Qu'est-ce que l'intelligence artificielle? Dans les années 50, il n'y avait pas de différence entre intelligence artificielle et informatique. Depuis, l'IA est devenu la branche de l'informatique qui se préoccupe surtout du raisonnement, l'apprentissage, la perception, la motricité, la créativité et le langage (ce qui est plutôt humain).

Définir l'intelligence artificielle n'est pas si simple. Le terme artificiel signifie qu'il s'agit du résultat d'une création humaine et non d'un processus biologique. Le terme intelligence est par contre beaucoup plus complexe à définir. L'homme peut faire des choses sans intelligence comme la reconnaissance de visages ou se déplacer. Kasparov joue pas comme Deep Blue qui évalue beaucoup plus de coup à l'avance, jusqu'à 24. Par contre Kasparov peut sans réfléchir éliminer des coups inutiles. Ainsi, l'ordinateur peut faire aussi bien mais avec des techniques très différentes. Peut-on néanmoins dire que l'ordinateur réfléchit pour autant?

1.1 Test de Turing

Il est très difficile de définir ce qu'est l'IA. C'est pourquoi Turing (1912–1954) donne une définition opérationnelle de l'IA, c'est-à-dire qu'il explique ce qu'il faut pour qu'il y ait intelligence artificielle. On discute avec quelqu'un, homme ou femme, par écran interposé. Si on croit que la machine est un être humain alors la machine est intelligente. Un être intelligent est donc quelque chose qui peut se faire passer pour humain.

Searle (1932), philosophe, n'est pas d'accord avec le test de Turing et pour expliquer son avis il utilise l'exemple de la « chambre chinoise », dont le principe est le suivant. Vous êtes emprisonné en Chine et ne comprenez pas le chinois. Le prisonnier voisin essaye de communiquer par papier sous la porte en chinois. Vous possédez un livre qui fait correspondre à un symbole chinois un autre symbole de la même langue. Vous utilisez ce livre pour communiquer. Si le chinois croit que vous parlez chinois alors vous êtes intelligent. Pourtant vous ne comprenez rien. L'ordinateur ne comprend pas ce qu'il fait il n'y a donc pas d'intelligence.

Eliza est un programme informatique avec lequel on peut dialoguer et qui illustre bien le problème de la « chambre chinoise. »

1.2 Bref historique

Dans les années cinquante on fait surtout de la traduction de langage, de la logique, etc. D'après Von Neumann, pour avoir de l'intelligence, il faut un hardware neuronal sophistiqué, parallèle et adaptatif. C'est l'époque du cybernétique, science du contrôle des systèmes en interaction, inventé par Norbert Wiener. En psychologie, on pensait encore que le cerveau apprenait par la carotte et le bâton, c'est ce qu'on appelle le béhaviorisme qui donne lieu à un apprentissage par renforcement, conditionnement. Ainsi, pour apprendre les échecs, l'ordinateur joue contre lui-même et apprend par un système de récompense.

Dans les années soixante, la psychologie redécouvre le cognitivisme: « Cognitif qualifie les processus mentaux par lesquels un organisme acquiert des informations sur son environnement et les traite pour ajuster

son comportement. »¹.

Dans les années septante, apparaît l'intelligence artificielle symbolique basé sur la logique. Le support matériel n'est plus important. Cette IA utilise un moteur de raisonnement appliqué sur des connaissances diverses (propres à l'application). Les Systèmes experts fonctionne sur ce principe, ainsi que le problème des cruches d'eau: avec un cruche de quatre litres et une autre de trois litres, on cherche à obtenir deux litres. Pour résoudre on définit un état initial et un état final ainsi que des opérations qui fasse évoluer la situation.

Dans les années quatre-vingts l'IA s'inspire de la biologie. La biologie c'est simple quand on isole mais mit ensemble ça devient incroyablement complexe. C'est sur ce principe que fonctionne les techniques collective d'unité simple. Penser n'est pas agir. Penser c'est résoudre des problèmes. On fait de l'IA comme nous devenons intelligent: d'abord la machine ne sait rien puis apprend petit à petit.

Le hardware n'es pas important en IA. L'intelligence est dans le software. L'ordinateur est une machine séquentielle, contrairement à ce que voulait Von Neumann. Un ordinateur fait une instruction à la fois alors que le cerveau en fait des milliard. Pour accélérer l'ordinateur on peut utiliser le parallélisme.

uh...

1.3 Vie artificielle

La vie artificielle consiste à reproduire dans des substrats artificiels (logiciels et robotiques) des mécanismes inhérents au vivant, à des fins: de biologie théorique (reproduire le vivant) et ingénieriste (s'inspirer du vivant). Il s'agit d'une vision essentiellement dématérialisée et fonctionnelle du vivant. La vie artificielle, ce n'est pas l'informatique qui va à la biologie mais l'inverse: la biologie à l'informatique.

La vie artificielle n'est pas de la bioinformatique dont les objectifs sont différents: séquençage, comparaison de séquence, prédiction de la structure 3D des protéines, data mining sur les microarrays,...

La biologie et l'informatique ont toujours fait bon ménage. En effet, ils fonctionnent à différents niveaux d'abstractions: aux niveaux ultimes, les mécanismes doivent être simples; ils ont une possibilité infinie d'essais et d'erreurs; ils utilisent la « force brute » pour faire émerger ou trouver des solutions complexes.

L'ingénieur essaye de faire des choses utiles pour l'homme alors que la science essaye de comprendre. Est-ce que la vie artificielle peut faire progresser la science, en particulier la biologie? Sûrement, mais pas encore on n'en est qu'au début. . .

1.4 Apport de l'informatique

Quel est l'apport de la modélisation informatique à l'étude des organismes vivants? Mais, qu'est-ce que la vie? Le premier à se poser cette question est le physicien Schrilinger(?), qui est un non-biologiste.

Morgan (1866–1945) était un généticien. Il étudia les variations phénotypiques chez la drosophile (mouche). Il reçut le Prix Nobel de médecine en 1933 pour avoir prouvé que les chromosomes sont les supports physiques de l'information génétique. Depuis, de nombreux décodages d'ADN ont été faits et l'on remarque beaucoup de ressemblance entre brin d'ADN. Néanmoins cela ne signifie rien. En effet, comme en informatique, le résultat du code dépend aussi du lecteur. Ceux qui disent que tout est dans le code ont tort.

Le professeur de l'UCL Christian de Duve (1917), prix Nobel de Médecine en 1974, définit la vie par sa structure physique. Mais durant la division cellulaire d'un embryon humain, la question se pose de savoir quand peut-on parler d'organisme vivant?

Pour les chimistes, le vivant, c'est ce qui est composé de carbone, mais on ne peut en faire avec du silicium des ordinateurs. On veut comprendre la vie par son fonctionnement et pas sa matière.

Pour les biologistes, ce qu'il faut nécessairement pour avoir la vie c'est la formation d'une membrane. En effet, il faut pouvoir ce différencier, se séparer. La molécule d'une membrane est constituée de deux parties: une tête, qui est attirée par l'eau (hydrophile), et une queue, qui n'aime pas l'eau (hydrophobe).

Réflexe de l'informaticien, il simule cela, en gardant la simplicité définit par le biologiste, et regarde ce que sa forme. La simulation montre que les membranes fusionnent. Ainsi, l'informatique met en évidence un manque sur la connaissance des membranes cellulaires.

Von Neumann (1903–1957) mathématicien, passionné de biologie au point d'étudier comment s'auto-répliquaient les cellules du cancer qui mit fin à sa vie. Il inventa une solution fonctionnelle. Pour pouvoir

¹François Bresson

refaire une machine, il faut le plan à côté de la machine. On a un constructeur universel C , $C(P_m) = M^P$ où P_m est le plan et M^P la machine qui correspond au plan. De plus, il faut que la nouvelle machine soit accompagnée du plan donc $C(P_m) = M^P(P_m)$ et que le constructeur universel soit aussi répliatif $C(P_c) = C^P(P_c)$. Ensuite, Von Neumann traduisit cela sous forme d'automate. Il démontra que l'autoréplication était tout à fait possible d'un point de vue logique. En 1953, Crick et Watson découvrent l'ADN et comprennent que l'existence de deux brins est utile pour l'autoréplication². Un brin est la machine l'autre le plan.

Turing, avec ces automates cellulaires, créer des formes en rayure de zèbre qui devrait expliquer comment de telles rayures apparaissent sur leur peau.

Beaucoup de biologiste pense que, grâce à l'informatique, on peut mieux comprendre la biologie. Pour faire des expériences biologiques complexe il faut faire de l'orienté objet. Par contre, il n'existe pas de hasard en informatique comme en biologie. D'après les biologistes, leur hasard est ni du pseudo aléatoire ni le hasard de la théorie quantique (physique). Qu'est-il alors?

²J.D. Watson, F.H.C. Crick. *Molecular structure of nucleic acids*. In Nature n° 4356. 25 avril 1953. pp 737-738. Disponible sur Web: <<http://www.nature.com/nature/dna50/watsoncrick.pdf>>. Note: s'ils ont reçu le prix Nobel c'est grâce au petit paragraphe de trois ligne ou ils disent que l'ADN permet l'autoréplication: « It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material. ».

Chapitre 2

Réseaux de neurones

Un neurone est un moteur oscillant qui ne stimule que s'il est stimulé (avec un seuil minimal). Les réseaux de neurones sont des systèmes inspirés du fonctionnement des vrais neurones. Ils sont pour la plupart implémentés par logiciel, mais on peut aussi avoir des réseaux de neurones électroniques qui ressemblent encore plus aux vrais neurones. Le principe est d'apprendre à partir d'une base de données avec des couples entrées-sorties. Le système va essayer d'aller de l'un à l'autre, mais on ne sait jamais vraiment comment il fait.

2.1 Perceptron

Le perceptron est le premier réseau de neurones, inventé dans les années cinquante. Il s'agissait d'un système de *perception* d'images. Le perceptron s'inspire de la biologie: associe des entrées à des réponses. Concrètement, l'entrée est une couche qui reçoit des informations. Il existe une couche de sortie qui fait une somme linéaire de la couche d'entrée pondérée par des poids. On rajoute une fonction de sortie qui seuilera la sortie.

Ceci est illustré à la figure 2.1(a). La figure 2.1(b) est la représentation d'un neurone de la couche de sortie, où x_i est la valeur de sortie de la i^{e} cellule de la rétine, w_{ij} est l'intensité de la connexion entre la i^{e} cellule d'entrée et la j^{e} cellule de sortie, a_j est l'activation de la j^{e} cellule de sortie, f est la fonction de transfert et o_j est la règle de décision: $o_j = 0$ pour $a_j \leq \theta_j$, 1 pour $a_j > \theta_j$. Le perceptron est non linéaire à cause de la fonction de transfert qui simule le seuil d'activation du neurone.

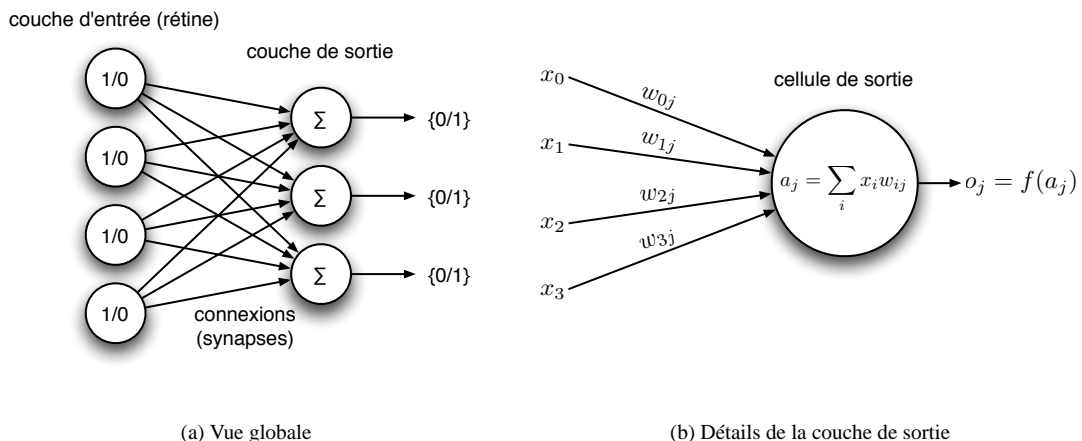


FIG. 2.1 – Le perceptron

Si les connexions sont connues à l'avance on peut les établir dès le début, sinon on fait des connexions

aléatoires qui vont s'ajuster ensuite pour s'améliorer. L'apprentissage se fait avec supervision car on connaît la sortie attendue. Si la sortie est bonne, on ne fait rien sinon, on augmente la valeur des connexions désactivées et diminue la valeur des connexions activées. Jusqu'au moment où les réponses sont toutes correctes. L'on augmente et diminue les valeurs en utilisant la règle d'apprentissage de Widrow-Hoff

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + n(t_j - o_j)x_i = w_{ij}^{(t)} + \Delta W_{ij} \quad (2.1)$$

où t_j est la sortie désiré et n un paramètre d'apprentissage (pas trop grand sinon, l'on aura sur apprentissage). Cette formule provient d'une méthode d'optimisation appelée « rétro-propagation du gradient » où l'on minimise l'erreur $(t_j - o_j)^2$. L'apprentissage ne dépend pas du nombre de neurones mais des connexions entre neurones.

Le perceptron est tombé en désuétude car les gens étaient moins intéressés par la perception que par la réflexion (yeux d'échecs, ...). De plus, Minsky montre qu'il est impossible d'apprendre le ou exclusif à un perceptron. En effet, le ou exclusif, représenté dans le plan en figure 2.2, est non linéairement séparable c'est-à-dire que l'on ne peut pas séparer les valeurs vraies et fausses (0 ou 1) par une droite (car deux entrées). Ainsi, pendant près de vingt ans, l'on entend plus parler de ce type d'intelligence artificielle.

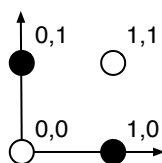


FIG. 2.2 – Ou exclusif dans la plan où le blanc est faux et le noir est vrai

2.2 Mémoires associatives

Dans les années septante, les réseaux de neurones réapparaissent sous le nom de mémoires associatives. Il existe deux types de mémoires associatives. Le premier type est la mémoire hétéro-associative, qui est une généralisation du perceptron, où l'activation de sortie est continue et non plus 0 ou 1. La constitution des mémoires hétéro-associatives est la même que le perceptron à part que

$$o_j = ya_j = y \left(\sum_i x_i w_{ij} \right). \quad (2.2)$$

Le second type de mémoire est dit mémoires auto-associatives, où tous les neurones de sortie sont reliés deux à deux comme à la figure 2.3. La recherche de réponse peut se faire soit par combinaison linéaire des stimuli stockés, soit de façon non linéaire, on parle alors de réseaux de Hopfield.

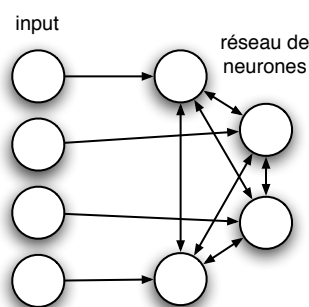


FIG. 2.3 – Mémoires auto-associatives

Dans les réseaux de Hopfield, les réponses sont binaires et valent -1 ou 1 et les mises à jours sont asynchrones. Si la $\sum w_{ij}a_i > 0$ alors la cellule a pour sortie $+1$, sinon -1 . Ce système converge en un point

fixe et se stabilise. L'intérêt principal de ce type de mémoire associative est que si on bruit l'entrée, comme les valeurs des voisins sont forts, elles vont activer les cellules qui ne l'ont pas été à cause du bruit dans l'entrée. Ainsi, on peut retrouver une information à partir d'une information bruitée. On peut aussi utiliser les mémoires auto-associatives aux lieux des systèmes experts dans des problèmes de décisions, comme le choix de prêter de l'argent pour une banque à un client. Bien qu'on préfère utiliser le data mining (voir chapitre 3) pour ce genre de problèmes.

2.3 Réseaux à couches cachées

Les réseaux à couches cachées sont des systèmes de neurones avec des cellules d'entrées et de sorties dont les connexions passent par d'autres cellules qui forment la couche cachée voir figure 2.4. Avec ce genre de réseaux, on peut tout faire, aussi bien linéaire que non linéaire (dont le ou exclusif). Pour l'apprentissage, on utilise la rétro-propagation de l'erreur qui se déroule comme suit. On place une entrée, après on calcule les réponses $h = f(Wx)$ pour la couche cachée et les réponses $o = f(Zh)$ pour la couche sortie. Ensuite, on calcule le signal d'erreur de sortie $\delta_{\text{sortie}} = f'(Zh)(t - o)$ pour ajuster Z avec le signal d'erreur $Z^{(t+1)} = Z^{(t)} + n\delta_{\text{sortie}}h = Z^{(t)} + \Delta^{(t)}Z$. Enfin, on calcule le signal d'erreur de la couche cachée $\delta_{\text{cachée}} = f'(Wx)(Z\delta_{\text{sortie}})$ pour ajuster W avec le signal d'erreur $W^{(t+1)} = W^{(t)} + n\delta_{\text{cachée}}x = W^{(t)} + \Delta^{(t)}W$. La fonction de transfert la plus populaire est la fonction

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

parce qu'elle donne de bon résultat et que ça dérivé est simple à calculer

$$f'(x) = f(x)[1 - f(x)]. \quad (2.4)$$

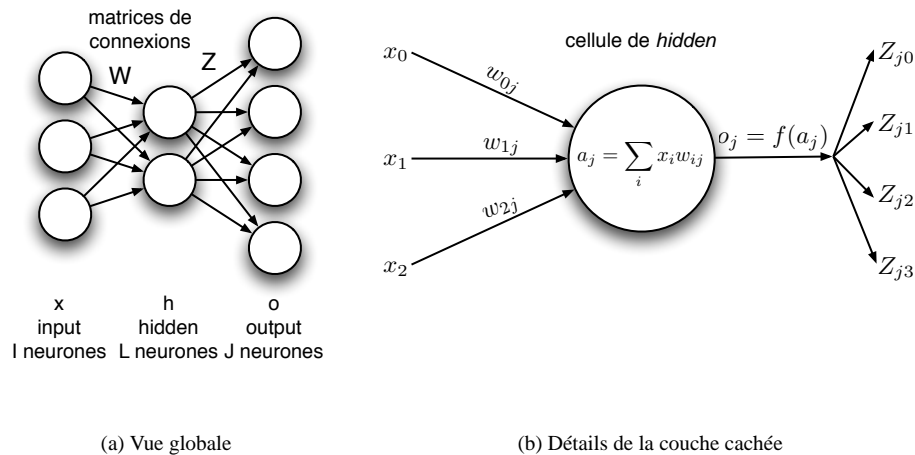


FIG. 2.4 – Réseau à couches cachées

Les réseaux de neurone sont utiles dans beaucoup de domaines. Parmi ceux-ci, on a la classification d'objet: reçoit quelque chose, analyse et renvoie quelque chose (prononciation de texte, reconnaissance d'image, ...). On peut aussi faire de la prévision: en entrant, on met $x_t, x_{t-1}, \dots, x_{t-n}$ et, en sortie, on reçoit x_{t+1} . Pour vérifier qu'un tel système fonctionne, on peut entrer 95% des données pour apprendre, et utiliser les 5% restant pour tester. Pour éviter le problème de sur apprentissage, on fait de la validation croisée, c'est-à-dire qu'on apprend tant que les résultats s'améliorent, une fois qu'on remarque une baisse dans les résultats, on s'arrête. Ce qui est important dans ce cas c'est la façon de généraliser, vu qu'on veut pouvoir « prédire », donc on préférera un réseau avec peu de neurone.

Chapitre 3

Data mining

Le *data mining* est né dans les années quatre-vingt et consiste à explorer des données pour y découvrir un *savoir*. Contrairement à l'étude statistique, où l'on émet des hypothèses qu'on vérifie avec les données, dans le *data mining*, on fait émerger des hypothèses à partir des données. Ainsi, le *data mining* fait apparaître des hypothèses que souvent on ne soupçonnait pas.

Par exemple, chez Belgacom, ils ont essayé de deviner quels clients allaient partir chez leurs concurrents. Leurs sources de données sont les communications téléphoniques (autrefois gardé jusqu'à la facturation), en plus des sources achetées chez d'autres entreprises comme Sol13. Ainsi, si quelqu'un téléphonait souvent à l'étranger, on lui proposait des forfaits « étranger. »

Pour un bon modèle, il faut des données correctes qui forment ce qu'on appelle le *data warehouse*. Un *data warehouse* doit être daté avec conservation de l'historique pour permettre des études comparatives et les données doivent provenir d'origine diverse. Il y a trois étapes à ce genre de spéculation: le rassemblement, la préparation de données et la modélisation.

Le *data mining* peut servir à la prédiction de série temporelle. Dans ce cas, la base de données contient les $x(t - n), \dots, x(t - 1), x(t)$ et le *data mining* nous donne $x(t + 1)$. En effet, le *data mining* modélise et donc cherche la fonction qui passe par les points de la base de données et permet ainsi de prédire la valeur suivante. On a une interprétation prédictive de la suite.

Un avantage de *data mining* sur les réseaux de neurones c'est qu'on sait pourquoi on arrive à un résultat vu qu'on garde toujours le contenu de la base de données. De plus, l'on peut faire des études sur des valeurs qualitatives (non chiffrés comme une nationalité, un métier, ...) et pas uniquement quantitatives (valeurs chiffrées comme le revenu, le nombre d'enfants, ...).

3.1 Clustering

Le clustering consiste à classer, regrouper une population par rapport à certaines caractéristiques comme illustrer à la figure 3.1. Par exemple, des médecins utilise cette technique pour essayer diagnostiquer le bon cancer d'un client. Les résultats aberrants peuvent être dû à des comportements illégaux, ainsi les mutuelles ont retrouvé des personnes qui trichaient à partie de leurs base de données. Si lorsque vous faites vos courses on vous donne à la caisse des bons de réduction que pour certains produits plutôt que d'autres, ce n'est pas par hasard... Pour valider un modèle on fait comme dans les réseaux de neurone, on utilise des données qui n'ont pas été utilisées pour l'apprentissage mais qui sont connues.

3.2 Arbres de décisions

Pour trouver à quelle classe fait partie un nouvel individu, il faut créer des règles de classification. Pour cela, on sépare les données en groupes et de là, on crée un arbre de décision. Les règles sont découvertes automatiquement vu que la classification est automatique. L'on prend d'abords l'attribut le plus discriminant et puis on fait de même pour les deux groupes avec d'autres attributs de discrimination et ainsi de suite pour le reste de l'arbre. Les arbres de décision sont très bons pour leur compréhension, car ils sélectionnent et séparent les variables. L'avantage de cette technique est qu'on divise avec des droites, alors que, pour les réseaux de neurones, c'est un peu du n'importe quoi...

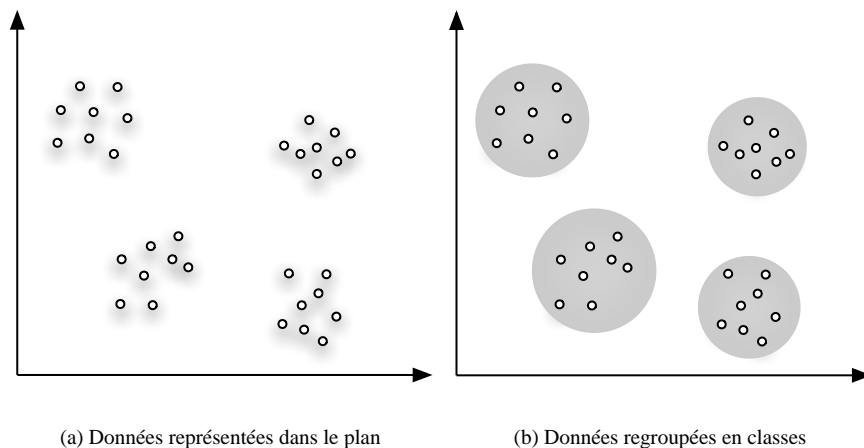


FIG. 3.1 – Exemple de clusering

3.3 Logique floue

Le but de la logique floue est de récupérer la connaissance d'un expert et de la mettre sous forme, ainsi on a des règles non-mathématiques mais intuitives. Cela permet d'approximer une fonction logique par une fonction continue. Un exemple est donné en figure 3.2 où la formule logique classique est: on est petit si on fait entre 1m60 et 1m75 et grand de 1m75 à 2m. Réguler par ensemble floue permet d'être plus proche de la réalité, plus souple et plus puissant. La logique floue était très répandue dans les années quatre-vingt et les toujours en Orient (surtout au Japon).

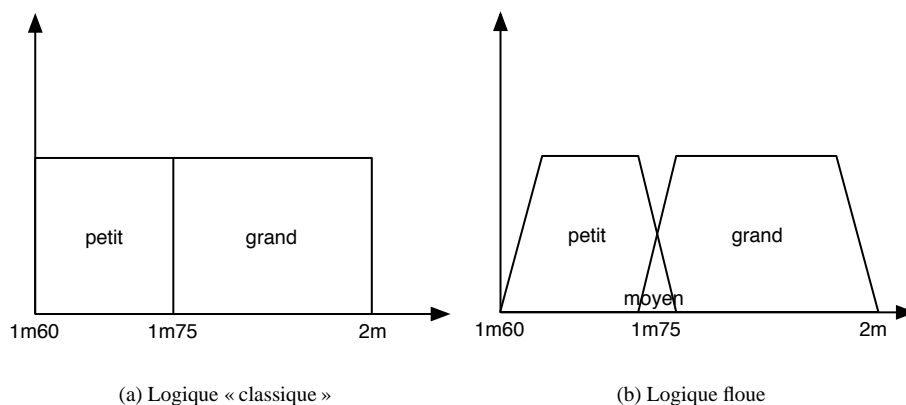


FIG. 3.2 – Comparaison entre la logique « classique » et la logique floue

On peut utiliser la logique floue pour la modélisation et surtout pour la prédiction de suite temporelle. Au lieu de chercher une fonction à variable réel tel qu'utilisé habituellement en mathématiques, on cherche une fonction à variable floue qui s'exprime, par exemple, de la façon suivante: si x est très petit alors y est petit, si x est petit alors y est moyen, si x est moyen alors y est moyen, ...

3.4 Lazy

Plutôt que de séparer les valeurs par des droites, on peut ne pas le séparer et regarder les plus proches voisins pour la valeur qui nous intéresse, dans la base de donnée. On recommence pour chaque valeur

différente. Amazone fonctionne ainsi pour conseiller des achats à partir des produits consultés par un client. Dans les techniques vues précédemment, on utilisait la base de données pour créer un modèle puis, on utilisait plus que le modèle. Une autre méthode consiste à toujours utiliser les données et jamais aucune modélisation. Ceci à pour avantage d'être moins dépendant de la qualité du modèle par contre il faut une puissance de calculs importants. Lazy est une de ces méthodes.

Le principe de lazy est de faire des approximations linéaire des voisins et à partir des droites obtenus on fait des prédictions. Comme l'approximation ce fait entre voisins proches, la prédiction sera locale et non globale. La qualité de la prédiction dépend du nombre de voisins, s'ils sont trop peu, le résultat sera imprécis (problème de bruits), s'ils sont trop nombreux, le résultat est trop général et donc pas précis.

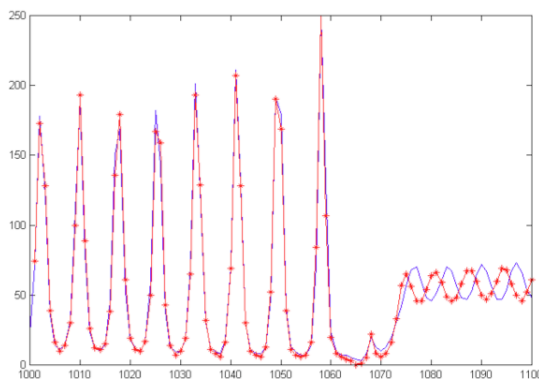


Fig. 3.3 – Lazy prédit, en rouge, les changements proches après 1060

3.5 BagFS

Il peut y avoir des erreurs dans la base de données. Pour diminuer l'effet de ces erreurs on va utiliser le BagFS, qui consiste à traiter le problème avec pleins d'arbres de décisions, qui traiteront chacun qu'une partie de la base de données et ce indépendamment des autres. Ceci affaiblit les arbres, ils deviennent naïfs, idiots, simplifiés mais ils sont beaucoup. Ainsi, quand on veut une approximation, on teste sur tous les arbres puis on choisit une des réponses (voix majoritaire, ...). Le principe qui se cache là derrière est que même si certains arbres vont se tromper, ils ne vont pas se tromper pour la même erreur, donc si beaucoup d'arbres donne la même solution c'est que ça doit être juste.

Parfois on connaît les liens entre variables. Il est donc inutile de les rechercher. On pourrait donc s'arranger pour qu'on puisse les entrer dans le système pour que ce dernier cherche seulement ce qu'on ne connaît pas.

Chapitre 4

Apprentissage par renforcement

Les arbres de décisions, c'est ce qu'il y a de mieux pour les échecs. Pour le backgammon, cela ne marche pas à cause de l'aléatoire dû aux dés et par le manque de pondération (reine > pion, ...). De plus, il n'existe pas de configuration favorable, où l'on est sûr de gagner. Ainsi, on laisse jouer l'ordinateur contre lui-même et l'on constitue une base de donnée sous forme d'arbre, avec des statistiques sur les configurations qui mènent à la défaite ou à la victoire. On donnera plus de poids sur les configurations près des feuilles. Cette méthode, appelé apprentissage par renforcement, est la meilleure connue pour le backgammon.

Il existe deux types d'apprentissage:

Supervisé pour chaque entrée, on a une sortie désirée qui permet de corriger le système; comme pour le perceptron ou les réseaux à couches cachées;

Non supervisé le système apprend grâce à une série d'input, mais aucune correction n'est faite; comme pour le clustering ou les réseaux d'Hopfield.

L'apprentissage par renforcement fait partie de l'apprentissage non supervisé. En effet, on ne connaît pas les règles (l'output). On apprend par soi-même. Si on trouve une solution, on est récompensé. Ainsi, on va s'améliorer en cherchant à obtenir le plus de récompenses possible.

L'apprentissage par renforcement peut se résumer aux étapes suivantes: je vois, j'agis et je modifie l'environnement selon l'action, ensuite je recevrai des récompenses. La récompense ne suit pas toujours l'action. En effet, parfois on ne sait pas tout de suite si le coup sera gagnant ou pas, comme pour le backgammon. L'idéal est de donner une récompense dès la fin de l'action car après plusieurs actions, il est plus difficile de créer des statistiques convenables sur les actions effectuées.

4.1 Dilemme exploration-exploitation

Quand, dans un jeu, on choisit tel et tel coup on a une information donnant la qualité des coups et l'on pourrait, si les coups étaient gagnants, refaire toujours les mêmes coups, mais problème, ça peut se révéler être perdant si l'adversaire change. Il faut donc toujours exploité de nouvelles combinaisons car l'environnement change toujours. Ainsi, on explorera le plus possible et ensuite on exploitera.

Imaginons une machine à sous avec deux bras. On voudrait savoir quel bras est le meilleur, sachant que quand on tire un bras on reçoit de l'argent qui varie. Ainsi, on tire souvent sur les deux bras et quand on a assez exploité, on peut commencer à explorer. Problème quand a-t-on assez exploité?

4.2 Q-learning

Le Q-learning est une méthode d'apprentissage par renforcement présenté par Watkins en 1989. Beaucoup de variantes ont été créées dont le $Q(\lambda)$ -learning présenté à la section suivante. Pour représenter les évolutions possibles du système (robots qui se déplace, choix du pion à jouer, ...), l'on utilise une chaîne de Markov. La propriété de Markov signifie que la probabilité d'une évolution ne dépend que de l'état du système et non de son histoire.

Chaque transition d'un état s^t à s^{t+1} de la chaîne correspond à une action précise. Selon que l'action est bénéfique ou non, l'on y attribue une valeur plus ou moins grande, appelée renforcement. Au départ, tous

les renforcements sont égaux et seront modifiés tout au long de l'exploration. Il n'est pas toujours possible de savoir à l'avance si une action est profitable ou non. Ainsi, l'on ne renforce que la dernière action et lors des recherches suivantes, l'on renforcera une action précédant celle déjà renforcé.

Lors de l'exploration, on choisira surtout les actions les moins utilisées. Ceci nous permettra de trouver une éventuelle meilleure solution. Alors, que pour l'exploitation, on utilisera les actions ayant le meilleur renforcement (méthode gloutonne). D'autres techniques sont aussi possibles comme un choix aléatoire des actions ou un choix selon la distribution de Boltzman.

4.3 $Q(\lambda)$ -learning

euh...

Chapitre 5

Stochastique locale search

La recherche stochastique locale sert à résoudre des problèmes combinatoires compliqués.
euh...

Chapitre 6

Intelligence collective

Le but de l'intelligence collective est de créer des algorithmes pas très intelligents qui, s'ils coopèrent, peuvent donner des choses très efficaces. L'intelligence collective est inspirée des insectes et d'animaux sociaux comme les fourmis, les termites, certaines abeilles, guêpes et araignées. Environ 2% des insectes sont sociaux et parmi ces animaux, la moitié sont des fourmis.

6.1 Fourmis

Le nombre de fourmis existantes est estimé à 10^{18} . Elles pèsent entre 1 et 5 mg. ainsi, le poids total des fourmis vaut environ le poids total des êtres humains. Les fourmis sont apparues il y a 100 millions d'années, alors que l'homo sapiens est apparu il n'y a que 50 000 ans. Les fourmis s'entraident pour créer un nid, en faisant des chaînes vivantes, elles se mettent à plusieurs pour transporter des charges lourdes et elles font encore pleins de trucs intéressants. Il existe une sorte de hiérarchie entre fourmis, elle s'organise en société: reine, défenseurs, collecteur de nourriture, nourrices pour larves, nettoyeur du nid, ... Cette structure fait apparaître quelque chose au niveau global par interaction. Personne ne connaît le système global, ils ne font qu'interagir avec eux-mêmes et leurs voisinages.

Pour avoir un système qui s'organise il faut une interaction multiple (de beaucoup de monde), de l'aléatoire (si quelqu'un fait quelque chose alors la probabilité que d'autres fassent de même augmente), un positif et négatif feed-back. Il faut aussi une communication. Il y a deux types de communication: directe (par contact physique, échange de liquides ou de nourritures, ...) et indirecte (par modification de l'environnement).

La coordination des tâches et la construction ne dépendent pas directement des ouvriers mais des constructions elles-mêmes. L'ouvrier ne dirige pas l'œuvre. Par exemple, si on retire des termites qui créent un nid et les remplace par d'autres, les nouvelles vont continuer le nid; donc les termites réagissent à ce qu'ils voient, à leur environnement, indépendamment des ouvriers précédents.

Une fourmi ou une termite, quand elle marche, dépose de la phéromone. La probabilité qu'une autre la suive est ainsi augmentée. Ceci a inspiré les algorithmes de *foraging*.

6.2 Foraging

Le *foraging* est un terme anglais signifiant chercher à manger, ici elle désigne un type d'algorithme basé sur les déplacements des fourmis.

Lorsqu'une fourmi sort du nid, entre autres pour chercher à manger, elle dépose de la phéromone tout au long de son parcours. Ceci leur permet d'optimiser leur déplacement. Si on a deux nids, les fourmis vont, au début, se déplacer de l'un à l'autre de façon plus ou moins aléatoire. Celles qui ont pris le chemin plus court arriveront les premières et donc celles qui partiront dans l'autre sens prendront plutôt le même chemin à cause de phéromones. De ce fait, les fourmis vont augmenter la quantité de phéromone du chemin. Ainsi, il y aura encore plus de fourmis qui l'utiliseront ce chemin. Le chemin pris par les fourmis va converger vers le chemin le plus court, soit la ligne droite entre les deux nids.

Le comportement de la fourmi dépend de la quantité de phéromone. Pour certains insectes, comme la guêpe, le comportement dépend de la qualité de la phéromone.

Imaginons, que les fourmis voyagent dans un tube d'un nid à un autre. Si l'on rajoute un raccourci au tube, les fourmis ne changeront pas leur comportement et continueront sur le plus long chemin à cause de la différence de phéromone. Les fourmis ne sont donc pas capables d'optimiser lors d'un changement soudain d'environnement. Néanmoins ce genre de problème peut être résolu facilement dans un algorithme.

Le *foraging* peut être utilisé pour la recherche de plus court chemin bien qu'on préférera utiliser les algorithmes usuels, plus efficaces pour ce problème. Par contre le *foraging* fonctionne bien avec des problèmes dynamiques comme la recherche de chemins par les routeurs. Malheureusement quand on trouve un plus court chemin on est jamais sûr qu'il n'y en a pas un autre plus court... Enfin, pour résoudre des optimisations combinatoires on peut utiliser le *foraging*.

6.3 Swarp-bot

Swarp-bot est un projet constitué d'une trentaine de robots, capable de faire de l'assemblage. Ces robots peuvent s'attacher l'un à l'autre, ainsi que s'adapter à l'environnement. Pour passer au-dessus d'un trou, les robots créent une chaîne assez longue pour passer ensemble au-dessus du trou. Ils sont aussi capables de se mettre ensemble, pour porter des objets trop lourds pour l'un d'eux. Ces robots sont inspirés des capacités d'assemblage et de transport de fourmis.

Ce projet est motivé par sa robustesse (capable de fonctionner en présence de robots défectueux), sa flexibilité (capable d'interagir à un environnement non prévu) et sa sociabilité (continue à fonctionner correctement quand on augmente le nombre de robots).

Le fonctionnement du système est décentralisé, c'est-à-dire que chaque robot se contrôle lui-même. Ce contrôle est fait par un réseau de neurone. Chaque robot est équipé de senseurs pour détecter les trous et obstacles qu'il franchira grâce à leur coopération. La communication entre robots voisins se fait par de petites lumières de différentes couleurs. Ceux qui veulent qu'on s'attache à eux deviennent rouges. Ceux qui veulent bien s'attacher seront bleus jusqu'à ce qu'il s'attache. Il deviendra alors rouge à leur tour.

La capacité des robots est limitée. Ils ne peuvent distinguer que trois couleurs jusqu'à vingt centimètres. Un robot ne pose pas de phéromone. Pour se déplacer, ils utilisent une autre technique. Le robot avance de 20 centimètres puis devient une base. Ainsi, ils créent une chaîne vers l'objectif. A un moment, un robot peut en avoir marre de faire la base et partir. Il n'est alors plus une base. Quand un robot est une base, il devient vert. Tout comme les fourmis, c'est bordélique, mais ça marche.

Chapitre 7

Réseaux

Un réseau est un graphe qui n'est pas figé dans le temps. Des réseaux, il en existe de toutes les sortes entre personnes, machines, animaux, etc.

Pál Erdős (1913-1996) est un mathématicien hongrois qui a publié quantité d'articles avec d'autres personnes. Ceci conduisit à la définition du « nombre d'Erdős », qui est le degré de collaboration avec Erdős. Erdős a le nombre zéro, ceux qui ont publié avec Erdős ont un, ceux qui ont publié avec un nombre d'Erdős un ont deux, etc.

Milgram, psychologue, qui a effectué l'expérience de soumission à l'autorité entre 1960 et 1963, est le père de l'expérience de la petite planète (1967). Cette dernière se base sur la distance entre personnes. Deux personnes sont de distance un s'ils se connaissent, de distance deux s'ils connaissent quelqu'un qui les connaît, etc. Milgram a montré que la distance maximale entre deux personnes est de six. Pour le prouver, il a demandé à des personnes d'envoyer des lettres à des gens choisis au hasard, sans utiliser un annuaire. La lettre passe donc de main à main. Pour arriver à destination, les lettres sont passées en moyenne par 5 à 6 personnes. Dans ce réseau, on trouve, à la fois, beaucoup de liens entre personnes proches et un petit nombre de liens entre personnes éloignées.

L'on différencie trois types de réseaux. D'abord, les réseaux individualistes qui satisfont chaque personne. Ensuite, les réseaux globaux qui satisfont globalement un utilisateur extérieur. Enfin, les réseaux inutiles qui ne sont ni individualistes ni globaux, mais des propriétés y sont observées. Par exemple, les réseaux d'ordinateurs sont individuels, les réseaux biologiques (neurone, ...) sont globaux et le réseau d'Erdős est inutile, alors que les réseaux terroristes peuvent être des trois types.

L'expérience de Milgram a mis en avant l'existence de connecteur (personne ayant une quantité de relations dépassant de loin la moyenne). Si le monde est « petit », c'est grâce à ces connecteurs. Un autre exemple où l'on a des connecteurs est le réseau fait dans les années septante sur les homosexuels atteint du SIDA dont le but était de découvrir l'origine de l'épidémie. Dans ce réseau, il y avait quelques connecteurs (gros baiseurs) et beaucoup de non-connecteurs (petit baiseurs). Pour la petite histoire, le record est détenu par un steward américain qui eut plus de 3000 conjoints. Dans le cas de la grippe aviaire, qui est un virus qui se déplace librement dans l'air, le réseau de grippe ne contiendrait pas de connecteur et les relations seront faites de façon tout à fait aléatoire. Dans le réseau de collaboration entre acteurs de cinéma, on trouve des connecteurs; principalement les seconds rôles qui sont un touche-à-tout. Le plus gros connecteur dans ce réseau est Kevin Bacon. Quelqu'un de nouveaux dans une relation a beaucoup de chance de se connecter au connecteur (effet boule-de-neige).

Des physiciens ont étudié les réseaux d'après leurs relations. Ils ont montré que si on prenait un réseau structuré auquel on ajoute quelques relations aléatoires on obtient un réseau intimement connecté. Ce genre de réseau est appelé « petit planète » d'après ces ressemblances avec le réseau de Milgram et ces connecteurs. Depuis, de nombreuses recherches ont été faites sur la structure « petit planète » qui a, entre autres, montré que les neurones du cerveau sont organisés selon cette architecture. Pour savoir pourquoi le cerveau n'était pas plutôt organisé de façon aléatoire, des scientifiques ont créé un réseau de neurone avec des liaisons structurées qui passait progressivement à une structure aléatoire. Les résultats montrèrent que c'est avec l'architecture « petit planète » que l'ordinateur a le mieux appris et le plus rapidement. Ceci expliquera pourquoi la sélection naturelle nous a conduit à un cerveau « petit planète. »

Napster a été facile à mettre hors d'état car il avait un serveur central donc un connecteur central qui a été détruit facilement. Sur gnutella, tous les nœuds sont des connecteurs. Si on veut un morceau, on

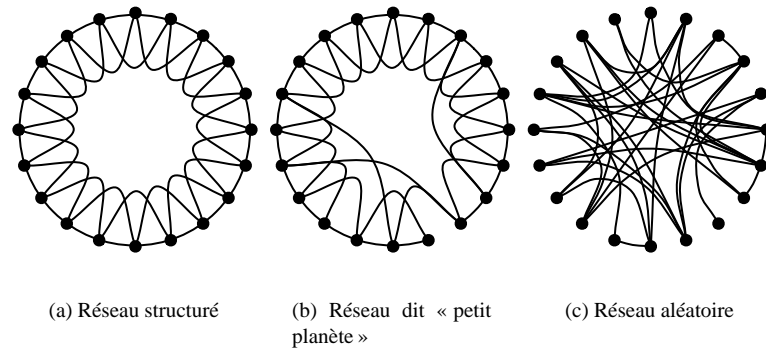


FIG. 7.1 – Comparaison des trois type de réseau

demande à cinq noeuds, qui demanderont à cinq noeuds, ainsi de suite. Cela va moins vite que napster mais va exponentiellement donc c'est quand même pas mal. Kazaa est un compromis. Il a quelque serveur, mais il y a toujours des noeuds connecteurs.

On peut faire un lien avec la politique. Dans le communisme de Lénine, tout est centralisé par une seule personne. Le libéralisme de Bush ne centralise rien. Tony Blair (social-libéralisme) centralise un peu mais pas trop.

7.1 Effets dynamiques

Les réseaux changent dans le temps. En effet, le Web grossi de plus en plus. Les réseaux trophiques montrent la relation mangée être mangé dans un écosystème. Ce réseau ne fait pas que grandir sa constitution change. Détruire une espèce aura la même conséquence qu'une épidémie car les espèces dépendent d'autres. Il y a une influence entre espèces.

La thèse Gaïa spécifie que la terre est vivante et en point fixe (stable dans le temps). Par exemple, si le gaz carbonique augmente, alors il fera plus chaud donc les plantes grandiront plus et absorberont plus de gaz carbonique. Il y a régulation donc point fixe. Cette thèse est fausse, mais permet de présenter ce qu'est un système en point fixe.

Les systèmes avec cycle passent toujours par les mêmes points. Par exemple le cerveau (horloge interne), le calendrier, cycle terre-soleil, etc. Imaginons la situation frustrante suivantes: Simon aime Paul qui n'aime pas Simon. Si Paul est content alors Simon l'est aussi. Comme Simon est content, Paul arrête de l'être. Ainsi, Simon ne le sera plus non plus ce qui rendra Paul heureux et donc Simon aussi. Ce système oscille. La relation proie-prédateur cycle: quand le nombre de proie sera important, le nombre de prédateurs augmentera et fera diminuer le nombre de proie.

Un système chaotique dépend uniquement des conditions initiales. Si on fait deux fois la même chose, on aura le même résultat par contre si on change une condition initiale, on peut avoir quelque chose de tout à fait différent. Le chaos est déterminé et imprévisible à cause de l'impossibilité de mesurer toutes les conditions initiales sans arrondis. Par exemple, les battements de cœur sont chaotiques. En cas d'arythmie cardiaque (maladie du cœur), les battements deviennent cycliques. Pour le cerveau, lors de crise d'épilepsie, les neurones oscillent de façon cyclique et plus chaotique.

Lorsque l'on montre à un lapin n'importe quoi, son électro-encéphalogramme est chaotique. Par contre, si on lui présente des carottes, on observe des cycles. Ceci, a été découvert en 1987 et ce vérifié avec l'être humain. Si on présente à un être humain des tâches, on aura un fonctionnement chaotique du cerveau. Par contre, si on lui présente des tâches qui représentent quelque chose, on fait apparaître des cycles.

Chaque neurone a un rôle: reconnaissance de couleurs, silhouettes, vitesses, etc. Si l'on observe plusieurs choses en même temps, il faut pouvoir associer convenablement les neurones. Pour ce faire, les neurones qui s'occupent d'une même chose auront la même phase d'oscillation. Ce qui fait apparaître des cycles. Les neurones s'occupant du fond oscillent de façon chaotique.

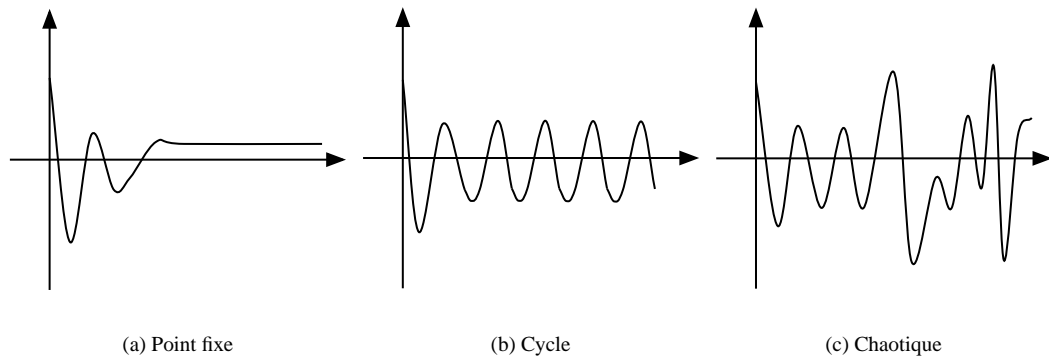


FIG. 7.2 – Représentation des différentes dynamiques

7.2 Réseaux et IRIDIA

L'IRIDIA étudie plusieurs réseaux, surtout des réseaux biologique.
 euh...

Chapitre 8

Algorithmes génétiques

Les algorithmes génétiques sont des algorithmes d'optimisation basée sur la théorie de Darwin, créé par John Holland en 1975. Cet algorithme fait partie de la famille des métaheuristiques, dont le but est d'obtenir une solution approchée lorsqu'il n'existe pas d'algorithme donnant une solution exacte en un temps raisonnable. Le pseudo code de cet algorithme est présenté ci-dessous.

Algorithme 1 : Pseudo code d'un algorithme génétique

Soit $X^{(n)} \subseteq X$, la population à l'étape n ;

répéter

- sélectionner dans $X^{(n)}$ un ensemble de paires de solutions de haute qualité;
- croiser chaque paire de solutions sélectionnées pour produire des solutions « enfants » ;
- remplacer une partie de $X^{(n)}$ formée de solution de basse qualité par des solution « enfants » de haute qualité;
- appliquer un opérateur de mutation aux solutions ainsi obtenues;
- les solutions éventuellement mutées constituent la population $X^{(n+1)}$;

jusqu'à ce que la règle d'arrêt soit satisfaite ;

La sélection ce fait en attribuant une probabilité de sélection à chaque individu. Meilleur est l'individu, plus sa probabilité d'être choisit est grande. Le croisement, parfois appelé crossing-over, ce fait généralement en permutant des séquences entre les deux parents. Il faudra faire attention à ce que la solution ainsi obtenu reste valable. La mutation ce fait plutôt sur les individus de faibles qualités. Le but de la mutation est d'introduire un élément de diversification et d'innovation comme dans la théorie darwinienne.

Chapitre 9

Compléments

9.1 Applets

Voici une liste d'Applets qui illustrent des techniques vues dans les chapitres précédents:

- Michel Casabianca. *Jeu de la vie*. [en ligne]. 2000 [référence de 2006]. Disponible sur Web: <<http://www.sdv.fr/pages/casa/html/vie.applet.html>>.
- Anna Nardella. *Biomorph Breeder*. [en ligne]. 2006 [référence de 2006]. Disponible sur Web: <<http://www.webalice.it/anna.nardella/biomorph.html>>.
- Fahri Tuncer. *Hopfield Applet*. [en ligne]. 2004 [référence de 2006]. Disponible sur Web: <<http://www.eee.metu.edu.tr/~alatan/Courses/Demo/Hopfield.htm>>.
- Frank Vanden Berghen. *The robot Applet*. [en ligne]. [référence de 2006]. Disponible sur Web: <<http://iridia.ulb.ac.be/~fvandenb/qlearning/qlearning.html>>.
- Eli Bachmutsky. *Self-Replication loops in Cellular Space*. [en ligne]. 1999 [référence de 2006]. Disponible sur Web: <<http://neeci.org/postdocs/sayama/sdsr/java/>>.

9.2 Coup de gueule de Bersini

Les mathématiciens font beaucoup de choses très pures qui ne marche pas avec la mise en pratique. Il y a beaucoup de mise en théorie, mais quand il faut faire de la pratique, il n'y a plus personne¹.

Les Japonais sont plus pragmatiques. Ils font et puis se demandent à quoi ça sert. Nous c'est l'inverse donc on fait moins.

9.3 Vidéo

Voici quelques commentaires sur la vidéo projetée lors du cours:

1. Bersini avait plus de cheveux avant;
2. le film date de 1999-2000;
3. le singe fonctionne par Q-learning;
4. l'homme aux insectes qui ne veut pas de software, commence à avouer l'utilité de ce dernier;
5. le chien de Sony a été mit en vente deux ans après le film;
6. le MIT est responsable du projet Cog;
7. l'IA c'est surtout étudiée aux Etats-unis et au Japon;
8. contrairement à ce que dit quelqu'un dans la vidéo, la simulation aide énormément la création de robots;
9. certains estime que d'ici 2050 on aura des équipes de foot robotique meilleur que des humains.

¹N.D.A.: les mathématiques n'ont pas pour objectif d'être mis en pratique.

Anecdote: lors d'un autre reportage, fait par la RTBF, Bersini était interrogé sur l'avenir de l'informatique et devait prédire comme elle serait en 2040. Comme il n'avait pas préparé le sujet et n'avait pas beaucoup d'idées lors de l'interview, il répondit que les ordinateurs on les verraient plus tellement ils seraient petits et présents partout. De plus ils communiqueraient tous ensemble. En 1970, la RTBF avait demandé la même chose pour l'an 2000; et en 1970, on avait répondu la même chose...

9.4 Lectures conseillées

- Jean Hamann. *Une très petite planète*. [en ligne]. 03 mars 2005 [référence du 06 février 2006]. Disponible sur Web: <<http://www.scom.ulaval.ca/Au.fil.des.evenements/2005/03.03/neurobiologie.html>>.
- Wikipédia. *Chaîne de Markov*. In Wikipédia, l'encyclopédie libre. [en ligne]. Mis à jour 04 février 2006. [référence du 04 février 2006]. Disponible sur Web: <http://fr.wikipedia.org/w/index.php?title=Cha%C3%A9ne_de_Markov&oldid=5337747>.
- Wikipédia. *Jeu de la vie*. In Wikipédia, l'encyclopédie libre. [en ligne]. Mis à jour 07 février 2006 [référence du 07 février 2006]. Disponible sur Web: <http://fr.wikipedia.org/w/index.php?title=Jeu_de_la_vie&oldid=5190870>.
- Marco Dorigo et al. *Swarm-bots project*. [en ligne]. Mis à jour 16 janvier 2006 [référence du 26 mai 2006]. Disponible sur Web: <<http://www.swarm-bots.org/>>.
- Hugues Bersini. *De l'intelligence humaine à l'intelligence artificielle*. Edition ellipses, 2006.
- ...

Bibliographie

- [1] Hugues Bersini. *Méthodes numériques d'intelligence artificielle*. [cours oral]. Bruxelles. Université libre de Bruxelles. 2005.
- [2] Hugues Bersini. *Méthodes numériques d'intelligence artificielle*. [en ligne]. [référence de 2006]. Disponible sur Internet: <ftp://iridia.ulb.ac.be/pub/bersini/coursIA1.ppt>
- [3] Hugues Bersini. *Réseaux de neurones*. [en ligne]. [référence de 2006]. Disponible sur Internet: <ftp://iridia.ulb.ac.be/pub/bersini/coursIA2.ppt>
- [4] Hugues Bersini. *Data Mining – How to make islands of knowledge emerging out of oceans of data*. [en ligne]. [référence de 2006]. Disponible sur Internet: <ftp://iridia.ulb.ac.be/pub/bersini/coursIA3.ppt>
- [5] Hugues Bersini. *Introduction to Reinforcement Learning – Q-Learning and evolutions*. [en ligne]. [référence de 2006]. Disponible sur Internet: <ftp://iridia.ulb.ac.be/pub/bersini/coursIA4.ppt>
- [6] Marco Dorigo. *Introduction to Ant Colony Optimization and Swarm Intelligence*. [en ligne]. 2004 [référence de 2006]. Disponible sur Web: <http://iridia.ulb.ac.be/~prasanna/pdfslides/1-Introduction-PPSN2004.pdf>
- [7] Hugues Bersini. *Networks*. [en ligne]. [référence de 2006]. Disponible sur Internet: <ftp://iridia.ulb.ac.be/pub/bersini/coursIA6.ppt>