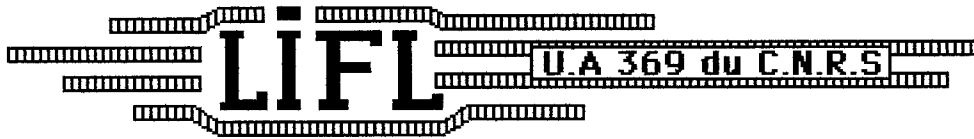
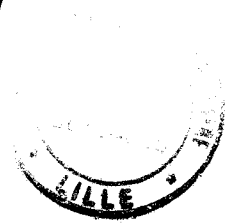


50376
1994
269

50376 2 102 551
1994
269

USTL
FLANDRES ARTOIS



CP

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

Numéro d'ordre :

THÈSE

Nouveau régime

présentée à

l'Université des Sciences et Techniques de Lille Flandres Artois

pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

Michel BINSE

**REPRÉSENTATION DE CONNAISSANCES
ET COMPLEXITÉ DE KOLMOGOROV.
LE CAS DU DESSIN AU TRAIT.**

Soutenue le 24 Novembre 1994 devant la commission d'examen

Membres du Jury

Président : Mr. Jean Paul DELAHAYE, Professeur USTL

Rapporteurs : Mr. Michel DE ROUGEMONT, Professeur ENSTA

Mr. Serge GRIGORIEFF, Professeur Paris VII

Directeur de thèse : Mr. Max DAUCHET, Professeur USTL

Examineur : Mr. Jean SALLANTIN, Directeur de recherches CNRS, LIRMM

Représentation de connaissances et complexité de Kolmogorov. Le cas du dessin au trait.

Résumé :

La ressemblance entre deux objets s'établit à la fois par ce qu'ils ont d'analogue ("information mutuelle") et d'identique ("information commune"). Nous utilisons ici le mot information dans son sens de contenu descriptionnel, ce qui nous place dans le cadre de la complexité de Kolmogorov. Le théorème de Gàcs et Korner (1973) affirme que "l'information commune" est probablement négligeable devant "l'information mutuelle". Pour ne pas nous placer dans le cadre de ce théorème, nous limitons le domaine à des langages restreints, en nous dotant d'un droit logarithmique à négliger. Ce défaut logarithmique est suffisamment large pour permettre de profiter des propriétés énoncées par Levin et Kolmogorov et suffisamment limité pour que les objets distincts inclus (au sens de l'information) dans un objet donné, soient en nombre polynomial. Nous imposons des contraintes à ce langage restreint pour que toute l'information mutuelle soit faite d'information commune. De tels langages seront appelés booléens. En effet, après avoir fixé une axiomatique, nous montrons que l'on peut manipuler les objets comme des ensembles. Nous illustrons par des exemples les résultats théoriques. Nous abordons alors le cas du dessin au trait, proposons un langage de représentation booléen, spécifions des qualités vis à vis d'un espace sémantique et illustrons par des exemples, des contre exemples, et un compromis : une représentation pyramidale. Nous montrons alors comment le traitement booléen permet de rester dans le raisonnable pour construire des bases de connaissance. Une seconde partie présente un vaste panorama des méthodes d'apprentissage et de représentation des connaissances, appliquées au cas du dessin au trait.

Mots clés :

représentation de connaissances, complexité, vision, compression, segmentation, apprentissage

Abstract :

The likeness between two objects, comes from both their analogous (mutual information) and identical (common information) features. Here we use the word information to mean descriptive content, which places us within the scope of Kolmogorov complexity. The Gàcs and Korner theorem (1973) states : "common information is probably far less than mutual information". In order to avoid putting ourselves in the context of this theorem, we limit the field to restricted languages with a logarithmic right to be overlooked. This logarithmic fault is big enough to take advantage of the properties stated by Levin and Kolmogorov and limited enough to have a polynomial number of distinct objects enclosed (in terms of information) in a given object. We lay down constraints to the restricted language so that all mutual information should consist of common information. We name such languages "boolean". Indeed, after stating axiomatic rules we show that objects can be handled like sets. We illustrate the theoretical results with examples. Then we get on to the case of outline drawing, putting forward a boolean language of representation, specifying qualities in relation to a semantic space and illustrating through examples, counter-examples and a compromise : a pyramidal representation. Then we prove how boolean treatment can proceed in a reasonable way to build knowledge bases. A second part presents a wide range of learning and knowledge representation methods applied to the case of outline drawing.

Key words :

knowledge representation complexity, vision, compression, segmentation, learning

REMERCIEMENTS

Je tiens à remercier

- Monsieur DELAHAYE d'avoir accepté de présider le jury de cette thèse.
- Messieurs DE ROUGEMENT et GRIGORIEFF qui m'ont fait l'honneur de bien vouloir en être les rapporteurs.

Je les remercie pour leurs commentaires, leurs critiques et observations qui m'ont permis d'améliorer ce document.

Je remercie également Monsieur SALLANTIN d'avoir participé au Jury et de m'avoir fait part de ses remarques concernant ce travail.

Je remercie tout particulièrement Max DAUCHET qui a dirigé ces travaux. Son expérience, ses remarques pertinentes et ses encouragements, m'ont permis de mener à bien ce travail et ont indéniablement contribué à renforcer mon goût pour la recherche.

Je remercie aussi Michel MERIAUX qui a dirigé avec Max DAUCHET la première partie de ce travail.

Je ne peux pas oublier les personnes avec qui j'ai eu beaucoup de plaisir à travailler.

Que Jérôme DELEU, trouve ici mes sincères remerciements.

Enfin je ne saurais terminer sans avoir remercié ma famille, et tout particulièrement, Monique, mon épouse, qui m'a toujours soutenu et encouragé dans cette entreprise, et à qui je dois beaucoup.

Plan

Introduction générale

où l'on présente notre démarche.

Première partie : Représentation de connaissances : une approche par la complexité de KOLMOGOROV.

0. Introduction.

où l'on aborde le propos.

1. Préliminaires relatifs aux machines.

où l'on fixe le cadre.

2. Complexité restreinte.

où l'on adapte à un point de vue.

3. Langages booléens de représentation.

où l'on cherche à imposer à un langage des contraintes pour que toute l'information mutuelle soit faite d'information commune.

4. Prétraitement numérique et compression. Représentations exactes et approchées.

où l'on introduit la notion de langage de description approximative.

5. Complexité et traitement numérique - symbolique. Une première approche.

où l'on illustre d'exemples et précise des qualités souhaitables.

6. Construction de bases de connaissances.

où l'on montre comment le traitement booléen permet de rester dans le raisonnable.

7. Conclusion

où l'on ouvre des perspectives

Annexes

1. Le dessin au trait.

2. L'approximation d'un dessin au trait par un nombre minimum de segments est NP-dure.

Bibliographie de la première partie.

Seconde partie : Représentation de connaissances et apprentissage : on étudie le cas du dessin au trait.

0. Introduction

où l'on aborde le propos.

1. La machine PASTIS

où l'on s'adapte aux contraintes technologiques.

2. Mécanismes humains de la vision

où l'on survole les modèles neurobiologiques de la vision.

3. Vision et dessins

où l'on présente et discute des mécanismes artificiels.

4. Apprentissage

où l'on classe les modèles.

5. Modélisation des systèmes complexes

où l'on introduit l'approche systémique.

Guide

où l'on résume les textes à aborder comme on regarde un paysage.

Sommaire

Introduction générale

où l'on indique notre démarche.

Cette thèse est séparée en deux parties de style différent, à lire séparément :

1. une approche complexité, de facture classique, est à parcourir de manière linéaire.
2. un voyage à travers l'apprentissage, est à découvrir comme un paysage, en le survolant.

Historiquement, c'est la seconde partie qui a débouché sur la première.

En effet, à l'origine, notre travail s'est placé dans le cadre du projet PASTIS de Lille, qui se proposait, en collaboration avec les équipes de Michel MERIAUX et de Max DAUCHET, de concevoir une machine qui traite rapidement et approximativement des dessins au trait.

Un premier composant, processeur de rotation rapide, a été réalisé et a fait l'objet de la thèse de Jérôme DELEU.

Ainsi, dans la mesure de nos possibilités matérielles, nous avons expérimenté des méthodes d'approximation de dessins par des segments.

Au travers de l'étude des différentes approches de représentations de dessins au trait et d'apprentissage, le souci de trouver un cadre intrinsèque¹, nous a amené au point de vue de la complexité de KOLMOGOROV.

Comme nous le verrons, la notion d'approximation peut être conservée dans ce cadre.

D'autre part, il est bien connu que l'approche de KOLMOGOROV est (hautement) non effective. On pourrait y voir une contradiction avec notre volonté primitive de rapidité. Nous éclairerons ce point dans la première partie.

Première partie : *à parcourir de manière linéaire, elle contient l'apport théorique de la thèse.*

En général, on ne peut pas dire en quoi deux objets sont différents ou se ressemblent (exemple : des signatures). La complexité de KOLMOGOROV a été introduite par des physiciens et des mathématiciens pour réaliser des descriptions universelles. Les seules études sur des restrictions portent sur des ressources bornées en temps ou en espace. Nous adoptons un point de vue d'informaticien qui veut étudier un domaine donné, au travers d'un langage restreint dans son pouvoir de spécification. Nous introduisons la notion de langage booléen qui permet sans ambiguïté de parler de l'information commune et de l'information propre à deux objets. Nous exploitons cette classe de langage pour étudier les liens entre le numérique et le symbolique, "symbolique" étant ici restreint en un sens que nous préciserons et nommerons "booléen". Le cas du dessin au trait est développé. Cette partie est de facture classique et s'adresse en priorité à des informaticiens ou plus généralement à des lecteurs ayant de bonnes bases mathématiques et informatiques.

C'est à l'introduction de cette partie qu'il faut se reporter pour rapidement en cerner le propos.

Seconde partie : *à découvrir en la survolant comme un paysage, elle met l'accent sur les représentations et l'apprentissage.*

Certaines méthodes connues depuis toujours réapparaissent. En effet notre volonté n'est pas de faire original à tout pris, mais d'étudier le domaine dans le plus grand nombre de ses aspects, tout en cherchant à préciser des passerelles entre différentes disciplines (informatique théorique, vision, apprentissage, cognition, architecture). Toutes les notions sont introduites de façon non technique afin d'être comprises des différentes communautés.

Nous présentons :

- la machine PASTIS, comme un mécanisme possible réalisant le projet.
- un survol des modèles neurobiologiques de la vision, comme référence à un objectif à atteindre.
- une étude des mécanismes artificiels de la vision, comme batterie d'outils pour appréhender des dessins.
- une classification des modèles de l'apprentissage, comme ensemble de méthodes pour un itinéraire de compréhension de nouveaux concepts, de nouvelles connaissances.
- une introduction à l'approche systémique, comme stratégie pour rendre intelligible les phénomènes étudiés.

Enfin un **guide**, en fin de document permet une lecture soit en deux pages, soit en résumé au septième.

C'est à ce guide qu'il faut se reporter pour rapidement cerner ce second aspect.

¹ : **intrinsèque** : qui tient de sa nature propre, qui appartient à son essence.

Première partie

Représentation de connaissances

une approche par la complexité de KOLMOGOROV.

0. Introduction

où l'on aborde le propos.

1. Préliminaires relatifs aux machines.

où l'on fixe le cadre.

2. Complexité restreinte.

où l'on adapte à un point de vue.

3. Langages booléens de représentation.

où l'on cherche à imposer à un langage, des contraintes pour que toute l'information mutuelle soit faite d'information commune.

4. Prétraitement numérique et compression. Représentations exactes et approchées.

où l'on introduit la notion de langage de description approximative.

5. Complexité et traitement numérique - symbolique. Une première approche.

où l'on illustre d'exemples et précise des qualités souhaitables.

6. Construction de bases de connaissances.

où l'on montre comment le traitement booléen permet de rester dans le raisonnable.

7. Conclusion

où l'on ouvre des perspectives.

Annexes

1. Le dessin au trait.
2. L'approximation d'un dessin au trait par un nombre minimum de segments est NP-dure.

Bibliographie

Partie 1 - Chapitre 0 .

Introduction

où l'on aborde le propos

Introduisons la problématique à partir d'exemples.

En quoi deux "objets", se ressemblent ? en quoi sont-ils dissemblables ?

Exemple 1 dit booléen: Observons les deux "objets" ci-dessous :

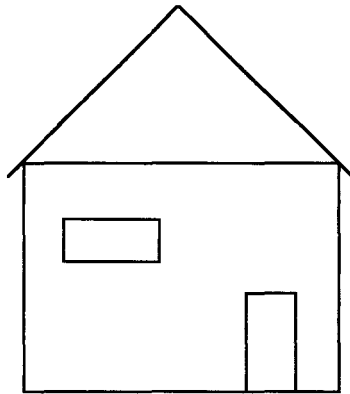


figure S1

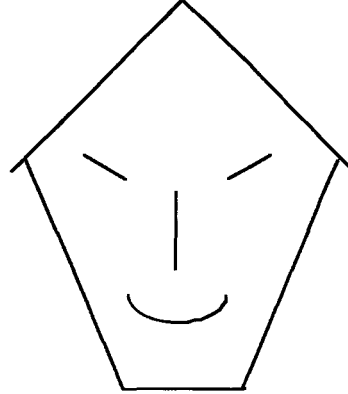
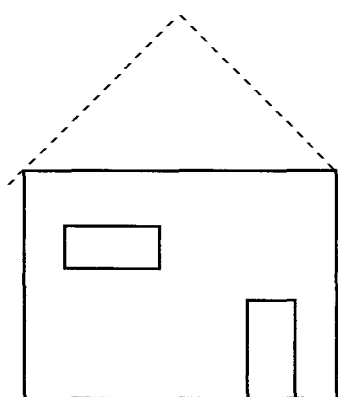


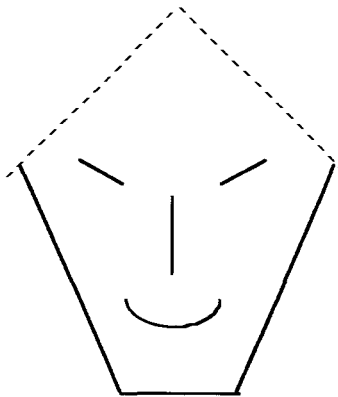
figure S2

Si l'on demande à plusieurs personnes isolées : "*Dites moi ce que ces deux figures ont en commun et ce que chacune d'elle a en propre ?*"

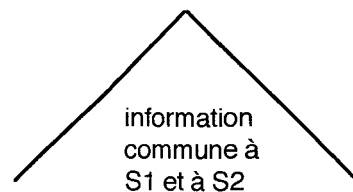
Toutes feront la même réponse, illustrée par les figures suivantes :



information propre à S1



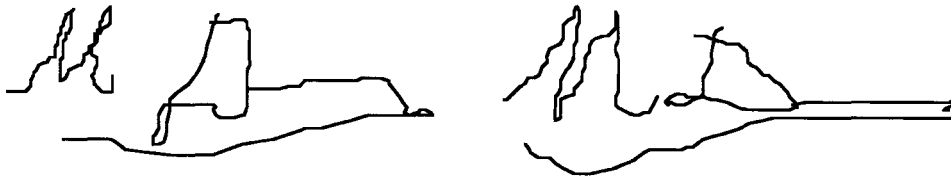
information propre à S2



information
commune à
S1 et à S2

Nous dirons que le langage de référence est "booléen". Dans un tel langage, la transmission des connaissances, la communication se font de façon formelle et non ambiguë.

Exemple 2 dit non booléen: Observons les deux signatures ci-dessous :



Si l'on fait la même expérience avec les deux signatures ci-dessus, les réponses vont varier. On ne saura pas expliciter formellement ce qu'elles ont en commun, ni en quoi elles diffèrent. Nous dirons que nous sommes dans un cas "non booléen".

Nos objectifs, notre approche :

Les constats ci-dessus sont bien connus. Notre but est de les étudier et de les préciser dans le cadre unificateur de la complexité de KOLMOGOROV. Celle-ci a l'avantage de donner une consistance mathématique très précise aux propos informels ci-dessus.

Dans ce cadre, un résultat ancien mais peu connu plante le décor : le théorème de GÀCS (1973) dit que presque toujours "l'information commune (à deux objets) est négligeable par rapport à leur information mutuelle". Autrement dit, dans la nature, on est presque toujours dans le cas des signatures : "Ça se ressemble beaucoup (information mutuelle) mais on ne peut expliciter que très peu de choses communes".

Les récents développements de la complexité de KOLMOGOROV sont portés par des physiciens, ou en tout cas des communautés soucieuses de l'universalité, ce qui exclut d'avoir les propriétés "booléennes" de l'exemple de la maison et de la tête.

Notre point de vue est différent :

Les apports principaux de notre travail :

- On relativise les langages
C'est à dire que l'on relativise délibérément l'approche à des domaines restreints de connaissance. (idée qui convient bien à la démarche "représentation de connaissances").
- On introduit la notion de langage booléen
C'est à dire que l'on se place dans un domaine dans lequel on peut représenter les connaissances en termes booléens pour faire des opérations booléennes sur objets.
- On éclaire (partiellement) le débat symbolique numérique.
- On ouvre à des problèmes.

Donnons quelques clés de l'approche que nous proposons :

1. La complexité de KOLMOGOROV.

Elle a été largement promue récemment par LI et VITANYI. (Voir aussi les articles de Jean Paul DELAHAYE). On retiendra pour le moment que :

La complexité de KOLMOGOROV notée $K(x)$ d'un objet (codé en une séquence x de 0 et de 1) est la longueur d'un plus petit programme qui engendre x . On définit de même la complexité relative $K(x / y)$, qui est la longueur minimale, étant donné y , d'un programme engendrant x .

- Cette notion est *robuste* : tous les résultats sont indépendants de la machine, pourvu que celle-ci modélise raisonnablement un ordinateur.
- Tous les résultats sont en fait *vrais "à un log près"* (à $O(\log(K(x)))$ près). La propagation de ces écarts lors des calculs peut engendrer des "*aspects gloutons*".
- Une séquence tirée à pile ou face est en général *incompressible*. (En fait, compresser nécessite de tirer parti d'une information structurelle).
- $E(x, y) = K(x / y) + K(y / x)$ définit une distance (ZUREK)
- On dispose d'une égalité entre les séquences

Quand on parlera de $x = y$, ce sera toujours "à un log près". Ainsi, $x = y$ signifie que l'on passe de x à y par un programme de longueur négligeable, c'est à dire en $O(\log)$.

En abrégé, nous parlerons de programme court, la longueur se réfère à la complexité de la séquence à laquelle il s'applique.

Notons - c'est important - que, pour une borne en $O(\log)$ donnée (ce qui est le cas pour une machine donnée), *le nombre de ces programmes courts est polynomialement borné par $K(x)$* .

- On démontre que s^* le plus court programme engendrant s est unique (au sens de nos conventions, c'est à dire à une transformation près par un programme court).

Notons que cette *explication s^* la plus courte de s* sera privilégiée au sens d' OCCAM (ce point de vue est développé largement par SOLOMONOFF et LEVIN, puis LI et VITÀNÝI).

- $I(x : y) = K(x) - K(x / y) = K(y) - K(y / x)$ est appelé *information mutuelle* de x et y .
- La complexité s'impose comme une notion très pénétrante, elle fait maintenant l'objet de travaux de physiciens (ou de travaux joints de physiciens et d'informaticiens) dans le but d'éclairer les notions d'entropie (CHAITIN, BENNET, GÀCS, LI, VITÀNÝI, et surtout ZUREK).

Elle fournit un cadre de pensée robuste, elle semble toucher à la "nature des choses", mais la contrepartie est que rien n'y est effectif.

2. Information mutuelle et information commune

Common information is far less than mutual information

GÀCS, KORNER, 1973

Nous touchons ici un point capital, à notre sens trop peu connu, résumé par le titre évocateur du papier de GÀCS et KORNER, cité en exergue. Nous allons introduire et illustrer le problème (on trouvera plus loin des définitions plus précises)

Étant données deux séquences on ne peut pas parler en général de leur information commune et de l'information propre à chacune.

La "tentation symbolique" :

Considérons trois séquences s, s', s'' *indépendantes*, c'est à dire telles que la donnée de deux d'entre elles ne donne aucune information sur la troisième :

$$K(s / (s', s'')) = K(s) \quad ; \quad K(s' / (s'', s)) = K(s') \quad \text{et} \quad K(s'') = K(s'' / (s, s'))$$

Pour simplifier, on les considérera de longueur n . On peut imaginer qu'elles sont les fruits de trois séries de n tirages à pile ou face.

Considérons $x = ss''$ et $x' = s''s'$;

On obtient $K(x) = K(x') = 2n$ (les séquences sont incompressibles);

$$K(x / x') = K(s) = n \quad \text{et donc} \quad I(x : x') = n.$$

On obtient la *décomposition booléenne* suivante :

s'' , de complexité $I(x : x')$, est l'information commune à x et x' ,

s est l'information propre à x ,

s' est l'information propre à x' .

Ici, on peut donc dire toute l'information mutuelle est "faite d'information commune".

Nous dirons que nous sommes dans le cas booléen si cette décomposition existe et est unique.

On posera $x \cap x' = s''$, $x - x' = s$ et $x' - x = s'$.

Remarquons que rien ne change si x et x' sont compressibles. Imaginons que, pour une machine de TURING universelle fixée, la donnée x fournisse le résultat y et x' fournisse y' . On a toujours $K(y) = K(y') = 2n$; $y \cap y' = s''$, $y - y' = s$ et $y' - y = s'$.

Par exemple, imaginons que x et x' soient les plans, condensés au maximum, de deux maisons y et y' . Les maisons sont le résultat d'un calcul, mais leur complexité est celle de leurs plans (exemple de DELAHAYE dans "Pour la Science"). En terme d'information, ce que les maisons ont en commun est ce que leurs plans ont en commun (4 chambres, garage, piscine, etc....).

Rien ne change non plus si x'' est réparti par un calcul (pas trop complexe) dans la séquence. Prenons par exemple z défini par : (z^i désigne le i ème bit de z , idem pour x , x' ...)

pour i impair non premier.... $z^i = s^{(i-1)/2}$

pour i pair..... $z^i = s''^i / 2$

pour i impair premier $z^i = 1$ si et seulement si $s^{(i-1)/2} = 0$

On passe de z à x et de x à z par des programmes de longueur négligeable, on a donc $z = x$ et rien n'est changé. (Notons que l'on est ainsi amené à identifier deux maisons apparemment fort différentes, mais fondamentalement de même conception - il suffit de savoir brièvement décrire l'autre en s'appuyant sur celle que l'on a sous les yeux).

s et s'' se déduisant par programme court de x (ou même de z), nous dirons que s et s'' sont des parties de x (ou informations partielles). z tout comme x se découpe donc en deux informations indépendantes s et s'' (nous noterons $z = x = [s, s'']$).

Dans cet exemple, x et x' ont une information commune s'' de complexité égale à leur information mutuelle.

En général, il n'en est rien, puisque GÀCS, KORNER ont mis en évidence il y a vingt ans que "l'information commune est (en général) bien moindre que l'information mutuelle".

En fait, l'information commune est en moyenne négligeable (en log). Remercions au passage VITÀNÝI et GÀCS qui nous ont directement informé par mail de ce résultat.

Que signifie intuitivement ce résultat?

Soient deux séquences x et x' . Nous voudrions les représenter en une "base de connaissance" la plus comprimée et structurée possible.

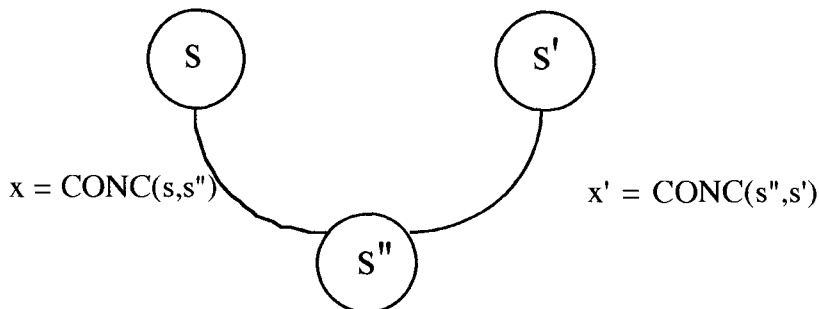
La complexité de cette base est $K(x, x') = K(x) + K(x') - I(x : x')$.

Dans le cas booléen, on a la solution (non effective) représentée par la figure page suivante. Elle fait intervenir trois séquences de base, (et deux programmes courts, de complexité négligeable)

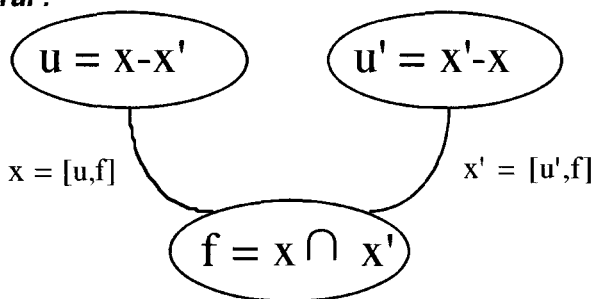
Le cas booléen :

Un exemple :

x et x' sont obtenus par des programmes courts à partir de s , s' et s'' qui jouent le rôle d'intersection et de différence, et tels que $K(s) + K(s') + K(s'') = K(x, x')$ et $K(s'') = I(x : x')$.



Le cas booléen général :



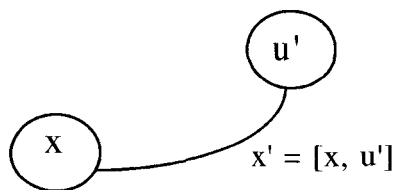
Représentation optimale et unique de (x, x')
 (à de petits programmes de transformation près,
 c'est à dire "au vocabulaire près")

Dans le cas général, une représentation optimale ne peut être que de la forme (x, u') , où u' est un plus court programme de x' à partir de x .

Dans les deux cas, la donnée d'un objet permet de raccourcir la plus courte description de l'autre de $I(x : x')$, mais dans le cas booléen seulement, on peut "expliquer pourquoi" : il existe une information commune correspondante dans les deux objets. Dans le cas général, l'information mutuelle est inexprimable.

Le cas général :

x et x' sont obtenus à partir de x et u' (ou de x' et u)
 u' est un plus court programme engendrant x' à partir de x .



On a $K(u') = K(x') - I(x : x')$ et $K(x, x') = K(x) + K(u')$

(x' est obtenu en appliquant u' à x)
 Représentation optimale de (x, x')
 (elle n'est pas unique)

• Remarque : L'exemple suivant montre que, même si il existe une information commune de complexité égale à l'information mutuelle, la décomposition booléenne n'est pas unique.

• Exemple :

Soit $x = ss'$ et $x' = s \oplus s'$ (s et s' sont incompressibles le longueur n comme précédemment). Alors $K(x) = 2n$, $K(x') = n$, $I(x : x') = n$. x' est l'information commune à x et x' de complexité x' , mais x admet deux décompositions (au moins) : $[s, s \oplus s']$ et $[s', s \oplus s']$.

Ainsi, en général, on peut constater que des objets sont corrélés, *mais on ne peut pas exprimer en quoi, leur information mutuelle ne peut pas être explicitée.*

Abordons maintenant directement le contenu technique de la première partie.

Les généralités sur les machines de TURING (1. **Préliminaires relatifs aux machines** (de TURING ou ordinateurs)) peuvent être sautées en première lecture.

La notion de langage restreint (2. Complexité restreinte)

Nous introduisons ensuite la notion de langage restreint.

Les langages que nous nous définissons sont restreints en ce sens que le jeu d'instructions et de contrôles qu'ils comportent ne permet pas de faire tout calcul. On peut imaginer un langage restreint comme un sous – Pascal, avec des modules de haut niveau, mais non universels.

Les définitions du paragraphe semblent techniques, mais elles ne font que traduire le fait que *la restriction doit préserver les bonnes propriétés de KOLMOGOROV dont jouit une machine de TURING universelle.* Toutes les égalités tiennent, comme dans le cas général, à un défaut logarithmique près.

Le fait que des restrictions conservent les bonnes propriétés ne va pas de soi, et c'est ce qui fait l'intérêt de la démarche.

Nous étudions ensuite quelles propriétés "minimales" doivent satisfaire des langages restreints pour être qualifiables de booléens (3. **Langages booléens de représentation**). L'idée a été illustrée ci-dessus : il s'agit de définir des langages tels que l'information mutuelle de deux objets manipulés soit descriptible par un objet. C'est de manière informelle cette propriété que nous qualifions de booléenne. Nous avons vu qu'elle nécessite une restriction du domaine de travail.

Plus précisément, nous posons des axiomes qui permettent de manipuler union, intersection, et différence d'informations.

Nous donnons de nombreux exemples et contre-exemples. Des calculs intuitivement "symboliques" comme l'unification le sont bien au sens booléen que nous proposons, mais de manière moins évidente que l'on pourrait le croire. Faire entrer l'unification dans notre cadre booléen oblige à une réflexion sur ce que l'on fait, et à expliciter des propriétés (pour la simplicité, nous ne traitons que le cas des mots).

Remarque : L'indépendance de deux informations se traduit de manière booléenne par la vacuité de leur intersection. On a donc dans le cas booléen l'indépendance deux à deux qui implique l'indépendance globale (la complexité de la réunion est la somme des complexités). Notons qu'en général, l' "effet ou exclusif" met en défaut cette propriété (considérer s , s' et $s \oplus s'$).

Deux importantes propriétés émergent de notre approche booléenne. Dans cette introduction informelle, nous les appellerons "aspect polynomial" et "limites du droit de négliger".

"Limites du droit de négliger"

Il est intéressant de noter que le "défaut logarithmique" est indispensable : il n'y a pas en remplacement de théorie "exacte" consistante. Ainsi, dans le cas de langages booléens, les informations peuvent être manipulées comme des ensembles dans une certaine mesure (les informations peuvent être arbitrairement complexes, mais le nombre de séquences manipulées ne doit pas être trop grand, sinon les défauts logarithmiques s'accumulent).

"Aspect polynomial"

Nous développons cet aspect dans le chapitre 6 (**6. Construction de bases de connaissance**).

De façon grossière, on peut dire que tout tient au fait que *dans le calcul booléen, l'ensemble des "morceaux" ou "parties" est polynomialement lié à la complexité* (alors que dans les ensembles, le nombre de parties est de manière exponentielle lié à la cardinalité). Par conséquent, lors de l'établissement ou de la consultation d'une base de connaissances, il n'y a pas d'explosion combinatoire : tout reste polynomial.

Ainsi sont liées naturellement approche booléenne et non explosion des explorations.

Notons que cette propriété est une propriété des couches supérieures des manipulations, elle ne dit rien de la complexité en temps des opérations élémentaires.

Mais nous esquissons aussi dans le chapitre 6 comment, la non effectivité, omniprésente dans l'approche générale de KOLMOGOROV, n'est plus, dans le cadre booléen, un obstacle rédhibitoire à la mise en oeuvre.

Les représentations exactes et approchées (4. Prétraitement numérique et compression.

Représentations exactes et approchées)

L'exemple du dessin au trait illustre bien deux problèmes :

- la difficulté de décrire dans un langage booléen des informations informelles.
- le traitement du bruit et de l'approximation.

Nous illustrons dans le chapitre 4 comment notre approche peut éclairer le connexionnisme et la logique floue, et les concilier avec la programmation modulaire et la logique bivaluée. Une large présentation informelle est faite au sein du paragraphe, lisible immédiatement et sans support technique. Nous en arrêtons donc là l'introduction et y renvoyons directement.

Le traitement numérique et symbolique dans un langage restreint (5. Complexité et traitement numérique - symbolique)

Nous illustrons une démarche par un exemple de classification de silhouettes de véhicules en quatre catégories. Tout d'abord un traitement numérique pour une approximation des silhouettes par segments, puis une analyse qui ne retient que des éléments pertinents enfin une représentation dans un langage restreint booléen pour réaliser des traitements booléens.

Nous nous replaçons dans le cas du dessin au trait en général pour proposer différents langages de représentation possédant à des degrés divers les qualités suivantes : la précision, l'analogie, la différenciation, l'épaisseur.

Les conclusions et perspectives de notre travail.

Cette première partie est née, répétons le, de notre désir de participer aux fondations de l'apprentissage et surtout des problèmes de représentation de connaissances. Nous pensons que notre approche (qui n'est qu'un modeste prolongement des idées d'auteurs comme KOLMOGOROV, CHAITIN, BENNET, GÀCS, ZUREK, LI et VITÀNYI) mérite d'être approfondie. Chaque chapitre n'est qu'une esquisse.

Le développement des concepts et résultats des chapitres 3, 4, et 5, l'étude des questions laissées en suspend, en particulier, devraient beaucoup apporter aux rapports symbolique – numérique, bivalué – flou.

Des expérimentations doivent être réalisées à partir de langages très restreints mais pertinents (dans le cadre du "dessin au trait").

Nous n'avons pas explicité dans cette première partie les problèmes d'apprentissage, qui sont pourtant notre but. Mais ils sont liés aux problèmes de constructions de bases de connaissances ici évoqués. Dans le domaine de l'apprentissage, des efforts théoriques ont été faits, dans la ligne de VALIANT par exemple, mais surtout de LI et VITÀNYI. Ces auteurs dressent une incontournable synthèse dans leur papier "Inductive Reasoning and KOLMOGOROV Complexity" qui s'appuie sur les résultats probabilistes profonds de SOLOMONOFF et LEVIN. Les résultats, qualitativement très éclairants, ont le défaut de n'être qu'un cadre non effectif - (encore que dans certains cas, ils retrouvent l'approche "presque toujours polynomiale" de VALIANT). Notre approche devrait, en contrepartie de son caractère restrictif, y apporter plus de praticabilité (au sens informel et formel).

Nous avons voulu, modestement, participer aux premiers pas sur la complexité de KOLMOGOROV utilisée dans la problématique de représentation des connaissances.

Partie 1 - Chapitre 1.

Préliminaires relatifs aux machines (de TURING ou ordinateurs)

où l'on fixe le cadre

1 Optimalité :

Dans l'approche classique de la complexité de KOLMOGOROV, on considère des machines de TURING universelles et optimales. Cette optimalité va tellement de soi qu'elle est souvent passée sous silence. Elle revient à dire que les programmes sont exprimables (potentiellement, pas effectivement) de façon compressée. Par exemple, une machine universelle où les séquences seraient exprimées en unaire serait non optimale !

Concrètement, une machine M universelle est optimale si

Pour toute machine M' , il existe une constante $C_{M, M'}$ telle que $\forall s, K_M(s) \leq K_{M'}(s) + C_{M, M'}$

2. Minimalité

Une machine de TURING universelle et optimale est dite minimale si la longueur de sa spécification est minimale et si, pour toute M' , on a $M(s; M') = M'(s)$ ¹. On peut évidemment donner des variantes (minimiser le nombre de règles par exemple). Intuitivement, ceci revient à considérer des machines sans procédures pré définies. Curieusement, cette notion n'est pas

¹ On peut toujours supposer qu'une donnée est repérée par son premier bit comme étant de type séquence ou de type machine. Cela n'influe que de 1 sur la complexité ! La définition signifie que l'on prend une machine, la plus simple possible qui simule le plus simplement possible les autres.

évoquée dans la littérature, alors qu'elle est importante : quelle que soit la séquence donnée, on peut construire une machine universelle optimale qui donne à cette séquence une complexité presque nulle en intégrant dans la définition de la machine une procédure pré définie de calcul de la machine; ceci biaise la complexité pour un nombre fini de cas seulement, donc n'affecte aucun résultat obtenu à une constante près, mais ceci est gênant quand on étudie concrètement des séquences de "basse" complexité. L'étude de la minimalité est néanmoins abordée dans le livre de LI & VITÁNYI. D'un autre côté, on peut comprendre la réticence à introduire la minimalité : d'abord, nous l'avons dit, elle est asymptotiquement non nécessaire; ensuite, contrairement à l'universalité et l'optimalité, elle est non effective, car on ne connaît pas la longueur minimale. De ce fait, dans la suite, nous nous contenterons de considérer des machines "assez courtes".

A retenir :

Pour notre part, nous considérerons des machines universelles, minimales et "presque" optimales. Concrètement, ceci revient à considérer un ordinateur usuel muni d'un langage de programmation élémentaire (sans procédure spéciale pré définie – il se pose par exemple le problème de la pré définition de l'arithmétique; on supposera que l'on ne dispose d'aucune opération arithmétique élémentaire). Nous allons raisonner en termes de machines de TURING, nous préciserons quand nécessaire, l'influence du choix de machine. Mais, en gros, les résultats tiennent pour ce que nous appellerons "un calculateur élémentaire" quelconque donné. On pourra sans dommage se soutenir l'intuition en référence à l'informatique de tous les jours.

Partie 1 - Chapitre 2.

Complexité restreinte

où l'on adapte à un point de vue.

La liste suivante de définitions est longue mais "naturelle". Seules les conditions de cohérence méritent attention. Elles vont de soi dans le cas de la complexité de KOLMOGOROV proprement dite. Elles soulignent des propriétés nécessaires à la fécondité de l'approche restreinte.

1. Définition :

On se fixe

– Un *domaine*¹ D , constitué par un ensemble récursif de séquences.

– Un *langage restreint* (LR) L (que l'on identifiera par la suite à une partie de D). Ce langage ne comporte pas d'autres entrées - sorties que les IN et OUT pré définis. Il est muni d'une syntaxe (utilisant les parenthèses) et d'une sémantique bien définies. A chaque programme p_i sont associés un certain entier n (positif ou nul) et une fonction partielle f_i de D^n dans $(\{0,1\}^*)^m$ (le résultat n'appartient pas nécessairement à D^m , il peut par exemple coder un booléen). La classe des programmes est *close par composition*, au sens des fonctions associées (sous respect des sortes : D , booléens, etc.).

On note $p_i(s_1, \dots, s_n)$ le résultat (si il existe) du programme p_i avec s_1, \dots, s_n comme données. Si $n = 0$, on note $p_i()$ le résultat.

¹ L'utilité de cette restriction ne s'impose pas, puisque les fonctions calculées par les programmes sont partielles. Mais nous souhaitons pouvoir ultérieurement distinguer l'appartenance d'une séquence à D , des opérations effectuées avec le langage L sur des données de D .

– On définit la *complexité d'une séquence s (ou d'une suite finie de séquences) de D restreinte à L*, notée $K_L(s)$

$$K_L(s) = \inf \{ |p| : p \in L \text{ \& } p() = s \}$$

– On définit de même la *complexité, relative à une suite finie ¹ s_1, \dots, s_n de séquences de D, d'une séquence s de D restreinte à L*,

$$K_L(s / s_1, \dots, s_n) = \inf \{ |p| : p \in L \text{ \& } p(s_1, \dots, s_n) = s \}$$

Notations : Nous noterons s / s' tout programme minimal transformant s en s' .

(Quand il n'y a pas d'ambiguïté, nous omettrons de préciser le langage en indice).

De plus, les **conditions de cohérence** suivantes doivent être satisfaites :

2. Conditions de cohérence.

– Cohérence ensembliste :

Pour tout n et tout n' , il existe un entier $E(v)$ tel que,

pour $n \leq v$ et $n' \leq v$, pour toutes suites de séquences $s = s_1, \dots, s_n$ et $t = t_1, \dots, t_{n'}$

pour toutes permutations σ de $\{1, \dots, n\}$ et σ' de $\{1, \dots, n'\}$, on ait

$$K_L(s_1, \dots, s_n / t_1, \dots, t_{n'}) \leq K_L(s_{\sigma(1)}, \dots, s_{\sigma(n)} / s_{\sigma'(1)}, \dots, s_{\sigma'(n')}) + E(v).$$

On supposera de même que $|K_L((s, s'), s'') - K_L(s, s', s'')| \leq E(3)$, formule qui se généralise de façon évidente à toute partition de l'ensemble.

De même (à un E près), on aura $K((u, u) / u) = 0$ et $K((u, v) / u) = 0$.

La fonction E est le *défaut de cohérence ensembliste*. Elle majore l'erreur commise en parlant de complexité d'ensembles finis en termes d'ensembles plutôt que de suites.

– Cohérence de l'information relative :

Il existe une fonction de défaut d'indépendance IND en $O(\log)$ telle que

$$K_L(s, s') = K_L(s / s') + K_L(s') \text{ (à } \text{IND}(K_L(s, s')) \text{ près)}$$

Cette propriété, que nous appelons cohérence de l'information relative, est vérifiée pour la complexité non restreinte de KOLMOGOROV, et en est une des pierres angulaires.

Les définitions classiques s'étendent naturellement :

– Soit une fonction f (à valeurs entières).

Une séquence s de D est $L-f$ -incompressible *si et seulement si* $K_L(s) \geq |s| - f(K_L(s))$.

– Un programme de L est dit *incompressible* si il est de longueur minimale parmi ceux de L qui lui sont équivalents.

En général, on dit que s est *incompressible* si f donnée est en $O(\log)$. Ce résultat vaut asymptotiquement, et le \log est dû au fait que l'on considère souvent les programmes comme étant auto délimités et pas les séquences.

¹ A priori, l'ordre de présentation des séquences influe.

Nous essaierons de préciser pour chaque L une telle fonction f , qui sera souvent constante ou en log de la complexité.

– Deux séquences sont L – indépendantes *si et seulement si*

$$K_L(s, s') \geq K_L(s) + K_L(s') - \text{IND}(K_L(s, s'))$$

Plus généralement, un ensemble de séquences (s_1, \dots, s_n) est L – indépendant si

$$K_L(s_1, \dots, s_n) \geq K_L(s_1) + \dots + K_L(s_n) - n(E(n) + \text{IND}(K_L(s_1, \dots, s_n)))$$

Un ensemble fini S de séquences est dit L - libre si toute partie finie est L – indépendante.

Toutes ces hypothèses impliquent que l'on peut identifier multi - ensembles et ensembles. Par exemple, si le programme "p;print" génère u, le programme "p;print;print" génère (u, u). Inversement, le programme "read;print" ne lira que le premier u de (u, u), qu'il restituera. Ainsi, par les seules entrées - sorties, on a bien $(u, u) = u$ (aux défauts près).

Remarque : Un ensemble fini de séquences est identifié à une séquence qui est la concaténée de ces séquences, supposées auto délimitées, mais il est nécessaire de se fixer l'ordre d'utilisation des données (sauf en cas d'opérateurs commutatifs), donc il est nécessaire d'utiliser des permutations d'éléments.

Comme il est imposé que la permutation soit une opération "négligeable", c'est à dire en $O(\log)$, on sera amené à considérer des sous-ensembles de cardinalité bornée en fonction de la complexité des ensembles considérés.

Remarque : D'autres approches seraient possibles

– Munir l'ensemble des séquences d'un ordre canonique (on peut prendre l'ordre lexicographique). La notion d'ensemble est remplacée par celle de suite ordonnée. La complexité de l'algorithme dépend alors de la complexité des permutations nécessaires sur les suites triées.

– Sortir de la borne logarithmique pour les défauts (laisser "flotter" les défauts).

– Considérer la notion de complexité d'ensemble; plusieurs notions existent.

Notation : Une famille libre sera notée entre crochets [].

Information mutuelle de deux séquences

Posons $I_L(s : s') = K_L(s) - K_L(s / s')$. D'après ce qui précède, cette notion est définie symétriquement à $\text{IM}(v) \leq 2 \text{IR}(v) + E(2)$ près. IM sera appelé *défaut d'information mutuelle*.

Notations et fonctions de défaut. Dans la suite, sauf besoin explicite, on omettra la référence en indice à L, et on fera les calculs en négligeant les fonctions de défaut. On dira que l'on fait les calculs *aux défauts près* (adp). De même, on négligera les problèmes auto délimitation et on parlera de *la séquence vide* λ .

Deux programmes déduits l'un de l'autre par des programmes courts (en log de la complexité) seront identifiés. En ce sens, l'égalité n'est donc transitive que dans la mesure où il existe de tels programmes de complexité majorée par la fonction de défaut.

3. Propriétés restreintes élémentaires :

(R-1) Associativité de [] : On a $[s, [s', s'']] = [s, s', s'']$

Preuve : $K([s, [s', s'']]) = K(s) + K([s', s'']) = K(s) + K(s') + K(s'')$.

(R0) Si $K(s / s') = 0$ et $K(s) = K(s')$ alors $s = s'$ (adp)

Preuve : de la cohérence de l'information relative, on déduit $K(s' / s) = 0$ donc $s = s'$

(ce qui signifie que l'on passe de s à s' par un programme au plus en $O(\log(|s|))$)

(R1) Si $K(s / s') = 0$, alors $(s, s') = s'$

Preuve : $K(s, s') = K(s / s') + K(s') = K(s')$ et $K(s / (s, s')) = 0$.

Donc R1 est une instance de R0.

(R2) La cohérence de la complexité relative assure le maintien, dans le cas restreint, de la situation classique, à savoir que :

pour tout s et s' , un programme v de longueur minimale engendrant s à partir de s' , est indépendant de s' , de complexité $K(s, s') - K(s')$, et vérifie $(s, s') = (s', v)$.

Un tel v se caractérise donc par $(s, s') = [v, s']$.

En effet, la formule $K(s, s') = K(s / s') + K(s')$ implique par définition l'existence d'un v de complexité $K(s, s') - K(s')$ tel que (s, s') s'obtient trivialement à partir de (s', v) .

Comme $K(s', v) = K(s') + K(v) = K(s, s')$, on a bien s' et v indépendants, et, d'après (R1), $(s, s') = (s', v)$. En considérant les complexités, on vérifie facilement que tout v' de longueur minimale engendrant s à partir de s' a ces propriétés.

(R3) Si $K(s / s') = 0$ et $K(s'' / s') = K(s'')$, alors $K(s'' / s) = K(s'')$

Preuve : de (R1) on tire $(s, s') = s'$. Supposons qu'il existe un v engendrant s'' à partir de s tel que $K(v) < K(s'')$. Comme $s' = (s, s')$, v engendre immédiatement s'' à partir de s' , ce qui contredit $K(s'' / s') = K(s'')$.

Propriétés élémentaires d'une complexité restreinte K_L

Il existe une constante C telle que l'on ait les propriétés suivantes.

– La longueur majore la complexité :

L'appartenance à L d'un "programme identité" (programme OUT, ou encore "print") nous assure que $K_L(s) \leq |s| + C$

– Cohérence de la composition et de l'auto référence :

Il existe un programme qui réalise la fonction identité.

Pour toutes séquences s, s', s'' , $K_L(s / s'') \leq K_L(s' / s'') + K_L(s / s') + C$ et $K_L(s / s) \leq C$

La preuve est immédiate par utilisation des procédures IN et OUT

– Identification entre programme et séquence :

A toute séquence donnée s de D correspond dans L le programme $OUT(s)$ qui calcule s ; à tout programme sans entrée on associe sa sortie.

Remarque : La cohérence de ces définitions est due au fait qu'un programme p incompressible l'est aussi en tant que séquence (à C près), comme on le vérifie sans peine. (idée : si la séquence p est résultat d'un programme p' plus court, alors p' fournit par deux exécutions successives un programme plus court pour la sortie de p).

Nous ferons un usage implicite de cette correspondance.

– Distance en Information Restreinte :

(La notion de distance en information fut introduite par ZUREK.)

Posons $E_L(s, s') = K_L(s / s') + K_L(s' / s)$

E_L définit une distance (aux fonctions de défauts près).

Preuve : On obtient immédiatement :

$$E_L(s, s) \leq 2C; E_L(s, s') = E_L(s', s); E_L(s, s'') \leq E_L(s, s') + E_L(s', s'') + 2C$$

– Deux séquences proches ont des complexités proches :

$$|K_L(s) - K_L(s')| \leq E_L(s, s') + C$$

Preuve : immédiate.

– Unicité des causes :

Il n'existe qu'une forme incompressible d'une séquence donnée (aux défauts près). Le programme ainsi associé à une séquence s sera noté s^* . Intuitivement, il sera, d'après la règle d' OCCAM, la cause la plus probable de s .

Plus précisément, ce résultat sera la conséquence du lemme suivant :

– Lemme de continuité des causes :

Deux séquences proches ont des causes proches. Plus précisément, soient p et p' deux programmes incompressibles sans entrée de résultats respectifs s et s' . Identifions les programmes à des données. (L étant donné, on l'omettra en indice.)

$$\text{On a } E(p, p') \leq E(p, s) + E(s, s') + E(s', p') + 4C$$

Comme p est incompressible et a pour sortie s , on a $K(s) = K(p)$ (à C près).

De cette égalité, de $K((s, p)) = K(s / p) + K(p) = K(p / s) + K(s)$ (à \mathbb{R} près) et de :

$K(s / p) = 0$ (à C près), on déduit que $K(p / s) = 0$ (à $C + \mathbb{R}$ près), donc que

$$E(s, p) = 0 \text{ (à } 2C + \mathbb{R} \text{ près). Finalement, on obtient } E(p, p') \leq E(s, s') \text{ (à } 2\mathbb{R} + 8C \text{ près)}$$

– Non unicité des causes relatives : Le phénomène "ou exclusif".

En général – comme dans le cas non restreint –, on peut avoir des programmes minimaux indépendants engendrant s' à partir de s .

Exemple : soient s et s' indépendants de même longueur, et $u = s \oplus s'$ leur "ou exclusif" : u et $u' = s'$ conviennent.

Partie 1 - Chapitre 3.

Langages booléens de représentation.

où l'on cherche à imposer à un langage des contraintes pour que toute l'information mutuelle soit faite d'information commune.

1. L'idée de base:

Information mutuelle et information commune.

Le point d'ancrage de l'aspect complexité de notre travail est un théorème dans un article de GÀCS et KORNER (dans Problems of Control and Inf. Th. 2 : 149-162, 1973) dont le titre est explicite: "l'information commune (à deux objets) est bien moindre que leur information mutuelle" ("Common information is far less than mutual information"). Précisons comment ce fait se traduit dans notre cadre, et comment il nous a amené à poser la notion de langage booléens.

Soient deux objets représentés par deux séquences s et s' .

On sait que l'information mutuelle de s et s' est la longueur dont la donnée de l'un raccourcit une plus courte explication de l'autre: $I(s:s') = K(s) - K(s/s') = K(s') - K(s'/s)$.

Nous allons dans la suite considérer comme partie d'une séquence u toute séquence u' qui se déduit de u par un programme court (de complexité en $\log(K(u))$). Autrement dit, avec le sens du signe = adopté précédemment, u' est une partie de u si et seulement si $K(u'/u)=0$.

Nous qualifierons de booléen tout langage dans lequel union, intersection et différence de séquences seront définis et satisferont aux axiomes des algèbres de BOOLE¹.

Ceci implique pour tout couple (s, s') , les identifications $s = (s \cap s', s - s')$ et $s' = (s \cap s', s' - s)$.

$s \cap s'$ sera l'information commune de s et s' .

En général, un langage n'est pas booléen. En effet, la définition implique l'unicité de la décomposition et implique l'indépendance des trois séquences de la décomposition. On en déduit immédiatement que $K(s \cap s') = \mathbf{I}(s:s')$. Or, le théorème de GÀCS et KORNER dit qu'en général, pour toute partie u (en notre sens) commune à s et s' , $K(u)$ est bien plus petit que $\mathbf{I}(s:s')$. Autrement dit, on ne peut pas en général dire qu'une séquence s se décompose en une séquence d'information commune à s et s' , et une séquence qui n'apporte aucune information à s' . Notons que même si on a une séquence u telle $K(u) = \mathbf{I}(s:s')$, l'exemple suivant montre que la décomposition de s en $(s \cap s', s - s')$ n'est pas unique: si on pose $s = u \oplus u'$, et $s' = u \oplus u''$, avec u et u' indépendants, $(u \oplus u', u)$ et $(u \oplus u', u')$ conviennent.

2. Axiomes booléens.

Un langage restreint L est dit *booléen* (LRB) si :

il existe un fonction de *défaut booléenne* S en log telle que l'on ait les propriétés suivantes

Axiome de la différence : pour toutes séquences s et s' , parmi les séquences u de complexité minimale engendrant s à partir de s' , une et une seule vérifie $K(u / s) = 0$ (adp).

Cette séquence sera notée $s - s'$.

Axiome d'indépendance : si $[s, s']$, $[s, s'']$ et $[s', s'']$ alors $[s, s', s'']$.

La complexité restreinte associée sera également dite *booléenne*.

Notons que dans le cas général, si u et u' indépendants et de même longueur, u , u' et $u \oplus u'$ fournissent un contre-exemple aux deux axiomes.

3. Propriétés booléennes élémentaires :

(S1) $s - s'$ est caractérisée par $(s, s') = [s - s', s']$ et $K(s - s' / s) = 0$

Preuve : traduction des définitions.

(S2) $[s, s'] = [s, s'']$ implique $s' = s''$.

Preuve : on a $s' = s'' = [s, s'] - s$

¹Nous ne considérons pas ici d'opérateur complément mais seulement les compléments relatifs.

(S3) Existence et unicité de l'intersection : pour toutes séquences s et s' , il existe une et une seule séquence f , telle que $K(f / s) = K(f / s') = 0$ et $K(f) = \mathbf{I}(s : s')$ (adp). f sera notée $s \cap s'$.

De plus, on a $[s \cap s', s' - s] = s'$ et $[s - s', s' - s, s \cap s'] = (s, s')$.

on appelle $[s - s', s' - s, s \cap s']$ la décomposition booléenne de (s, s')

Preuve :

existence : considérons $u = s - s'$ et $f = s - u$. On a $[f, u] = (s, u) = s$ et donc $K(f / s) = 0$. On considère de même $u' = s' - s$ et $f' = s' - u'$. On a $[f', u'] = s'$ et $(s, s') = [u, [f', u']]$. On en déduit $[[u, u'], f] = [[u, u'], f']$ et donc $f = f'$. La décomposition booléenne est assurée.

Enfin, $\mathbf{I}(s : s') = (K(f) + K(u)) + (K(f) + K(u')) - (K(u) + K(u') + K(f)) = K(f)$.

unicité : Soit g tel que $K(g / s) = K(g / s') = 0$ et $K(g) = \mathbf{I}(s : s')$. Considérons v de complexité minimale engendrant s sachant g . On sait qu'alors $[g, v] = (g, s)$. Ici, $(g, s) = s$. Considérant de même s' sachant g , on obtient l'existence de v et v' tels que $[g, v] = s$ et $[g, v'] = s'$. On obtient $(s, s') = (g, v, v')$. Mais comme $K(g) = \mathbf{I}(s : s')$, on a $K(g) + K(v) + K(v') = (K(g) + K(v)) + (K(g) + K(v')) - K(g) = K(s) + K(s') - K(g) = K(s, s')$. Donc $(s, s') = [g, v, v']$. En particulier, $[s, v'] = (s, s')$ et $K(v' / s') = 0$. Par unicité de la différence, on en déduit que $v' = u'$.

Finalement, on obtient $[(u, u'), g] = [(u, u'), f]$ donc $g = f$.

(S4) $[s, s']$ implique $s \cap s' = \lambda$.

la preuve est évidente.

(S5) $K(s / s') = K(s / s'') = K(s)$ implique $K(s / (s', s'')) = K(s)$.

Remarquons que ceci s'écrit encore $[s, s']$ et $[s, s'']$ implique $[s, (s', s'')]$

Preuve : soit $[u', u'', f]$ la décomposition booléenne de (s', s'') . D'après (R3), on a $[s, u']$.

Comme $[u', s'']$, l'axiome d'indépendance assure $[s, u', s'']$.

Pour conclure, on remarque que $(u', s'') = (s', s'')$.

(S6) $(s - s'', s' - s'') = (s, s') - s''$.

Preuve : posons $u = s - s''$ et $u' = s' - s''$. On a $K(u / s) = K(u' / s') = 0$

donc $K((u, u') / (s, s')) = 0$. D'autre part, $(s'', s) = [u, s'']$ et $(s'', s') = [u', s'']$.

D'après, S5, on obtient $[(u', u), s''] = (s'', (s, s'))$, ce qui assure le résultat.

(S7) $K((s, s') / s'') = K(s - s'', s' - s'')$.

C'est un corollaire de (S6).

(S8) Si les axiomes booléens sont vrais sur D , ils sont vrais sur les parties finies de D .

Preuve : (E, E', \dots) désignent des ensembles).

a/ Preuve que l'axiome de la différence est vrai sur les parties finies de D .

On se ramène au cas des séquences en considérant les programmes minimaux s et s' de E et E' .

b/ Preuve que l'axiome d'indépendance est vrai sur les parties finies de D .

Attention, la notation $[E, E']$ signifie seulement $K(E, E') = K(E) + K(E')$. C'est pourquoi nous prenons ci dessous la notation en extension.

Pour montrer que si des s_i sont indépendants deux à deux, alors $[s_1, \dots, s_n]$, on procède par récurrence sur n : soit p le plus court programme de (s_1, \dots, s_{n-1}) . Par

hypothèse, on a $[p, s_n]$, $[p, s_{n+1}]$ et $[s_n, s_{n+1}]$. On en déduit $[p, s_n, s_{n+1}]$. Comme $[s_1, \dots, s_{n-1}]$, on déduit le résultat de l'associativité des crochets.

(S9) Distributivité de l'intersection par rapport à l'appariement

pour toutes séquences s, s', s'' , on a $(s \cap s', s \cap s'') = s \cap (s', s'')$

Preuve :

a / Supposons s' et s'' indépendants.

Soit $[u, u', f]$ la décomposition booléenne de (s, s') et $[v, v'', g]$ celle de (s, s'') .

Nous allons montrer que $(f, g) = s \cap (s', s'')$.

On a immédiatement $K((f, g) / s) = 0$ et $K((f, g) / (s', s'')) = 0$.

Il reste à établir $K(f, g) = \mathbf{I}(s : (s', s''))$.

$$\mathbf{I}(s : (s', s'')) = K(s', s'') + K(s) - K(s, s', s'')$$

$$\mathbf{I}(s : (s', s'')) = K(s') + K(s'') + K(s) - K((s', s'') / s) - K(s)$$

$$\mathbf{I}(s : (s', s'')) = K(s') + K(s'') - K(s' / s) - K(s'' / (s', s))$$

On a $K(s' / s) = K(u')$ et $K(s'' / (s', s)) = K(v'' / (s', s)) = K(v'')$ compte tenu de S5.

Donc $\mathbf{I}(s : (s', s'')) = K(s') - K(u') + K(s'') - K(v'') = K(f) + K(g) = K(f, g)$ car, d'après R3, f et g sont indépendants si s' et s'' le sont.

b / Cas général.

On considère la décomposition booléenne $[w', w'', h]$ de (s', s'') . D'après a /, on a :

$$(s \cap s', s \cap s'') = (s \cap [w', h], s \cap [w'', h]) = (s \cap w', s \cap w'', s \cap h) = (s \cap (w', w''), s \cap h) = s \cap ((w', w''), h) = s \cap (s', s'')$$

(S10) Associativité de l'intersection $(s \cap s') \cap s'' = s \cap (s' \cap s'')$

Preuve :

a / Si $[s', s'']$ alors $s \cap (s' \cap s'') = (s \cap s') \cap s'' = \lambda$

Preuve : immédiat d'après S4 et R3

b / $(s \cap s') \cap s' = s \cap s'$

Preuve : soit $[u, u', f]$ la décomposition booléenne de (s, s') .

$$(s \cap s') \cap s' = f \cap [u', f] = (f \cap u', f \cap f) = (\lambda, f) = f = s \cap s'$$

c / le cas général. Soit $[w', w'', h]$ la décomposition booléenne de (s', s'') . On va appliquer plusieurs fois la distributivité

$$(s \cap s') \cap s'' = (s \cap (w', h)) \cap (w'', h) = (s \cap w', s \cap h) \cap (w'', h) =$$

$$((s \cap w', s \cap h) \cap w''), (s \cap w', s \cap h) \cap h) =$$

$$((s \cap w') \cap w'', (s \cap h) \cap w''), (s \cap w') \cap h, (s \cap h) \cap h)$$

compte tenu de a / et b /, on obtient $s \cap h$, qui vaut bien $s \cap (s' \cap s'')$

(S11) Distributivité de l'appariement $(s, s' \cap s'') = (s, s') \cap (s, s'')$

On applique la distributivité de l'intersection

$$(s, s') \cap (s, s'') = (s \cap (s, s''), s' \cap (s, s'')) = (s \cap s, s \cap s'', s' \cap s, s' \cap s'')$$

donc $(s, s') \cap (s, s'') = (s, s' \cap s'', (s \cap s'', s' \cap s))$ mais on a $K(s \cap s'' / s) = K(s \cap s' / s) = 0$ d'après la définition de l'intersection., d'où le résultat.

(S12) La cohérence de l'information relative découle de l'axiome de la différence.

La preuve est immédiate.

Les notations booléennes :

Nous poserons dorénavant

$$(s, s') = s \cup s'$$

On pose aussi classiquement

$$s \subseteq s' \text{ si et seulement si } s \cup s' = s$$

On dira que s est *incluse* dans s' , ou est *une partie* de s' .

Pour nous, ceci équivaut immédiatement à

$$s \subseteq s' \text{ si et seulement si } K(s / s') = 0$$

*Les propriétés établies permettent, sous les deux axiomes booléens, de manipuler sans précaution – mais, aux défauts près – les symboles \cap , \cup et $-$ comme les symboles booléens. L'opérateur \cup peut évidemment être dérivé de \cap et $-$, mais par tradition nous garderons les trois opérateurs, que nous appellerons *les opérateurs booléens*.*

Notons que l'on peut majorer les différentes fonctions de défaut et les défauts des formules obtenues par une seule fonction de défaut Δ qui est en $O(\log)$.

Remarque sur les parties

Le nombre de parties est polynomialement lié à la complexité de l'ensemble.

En effet, soit $s \subseteq s'$.

Un programme qui déduit s de s' est d'une longueur majorée par $C \cdot \log(K(s'))$ (pour une certaine constante C fixée par la fonction de défaut). Il y a donc un nombre polynomial de tels programmes donc de parties. Par exemple, pour les mots, on peut prendre des réunions bornées de facteurs (on retrouve la nécessité de borner en fonction des défauts tolérés).

4. Exemples :

Nous illustrons brièvement comment la structure de pile et l'unification, qui sont des archétypes de traitement symbolique au sens usuel, rentrent dans le cadre booléen à notre sens.

Exemple 1. : Les mots et la structure de pile

On va considérer un alphabet fini donné et on suppose le fond de pile à gauche (on empile de gauche à droite). En fait, on considère le ruban d'entrées – sorties comme une pile. On souhaite que tout mot soit incompressible et formaliser l'idée que la transformation la plus simple qui fait passer du mot fau au mot fbu est "dépiler jusqu'à la case lf ; empiler(bu)" (f , u et u' sont des mots, a et b des lettres différentes). L'opération qui consiste à dépiler jusqu'à la première lettre qui fait la différence est indiquée par l'écriture binaire de lf , et est donc de complexité négligeable (en \log par rapport à la complexité des mots, c'est à dire ici leur longueur).

Une impasse

Une idée naturelle est de considérer les instructions de base.

dépiler(k) qui dépile jusqu'à la case k , de complexité (environ) $\log(k)$

empiler(m), de complexité lml .

Une première analyse sur la commutation des instructions (que nous passons), amène plutôt à considérer les instructions *empiler*(k, m), qui signifie que l'on empile à partir de k .

L'exemple suivant illustre l'échec :

Soit les mots fau et fbu . On voudrait obtenir $fau - fbu = au$, qui correspondrait au programme

(P) "dépiler(lf); empiler(k, au)" appliqué à la donnée fbu .

de complexité, au \log près, $K(P) = lbul$

Or, par identification des programmes à des données, comme :

fau est obtenu par *empiler*(lfa, u) appliqué à fa ,

on a $fau = (\text{empiler}(lfa, u), fa)$. Par projection ensembliste, on en déduit la nécessité de considérer les séquences de la forme *empiler*(lvi, u), qui représentent des fragments de piles, comme parties de vu ; c'est à dire que l'on considère des instructions *fragment*(k, k') qui considèrent le fragment correspondant d'une donnée. On identifie aussi le programme *empiler*(k, u) à la séquence (k, u) .

Considérons alors le programme (P') " *fragment*($lfb, lfbu$); dépiler(lf); empiler(lf, a)"

P' transforme fbu en (lfb, u), ($0, f$), (lf, a) qui est une partition de fau , et P' est de complexité indépendante de u , ce qui montre que l'on a pas formalisé ce que l'on voulait.

Remarque : Notons que l'on pourrait étudier si, partant de cette base, on obtient un langage booléen. En première analyse, $m - m'$ serait constitué des occurrences de lettres dans m et non dans m' . Par exemple, on aurait $abaabab - abbaaa = (a,3), (b,5), (b,7)$.

Une solution

Afin d'éviter les "effets de bords" ensemblistes, on va associer à chaque occurrence de lettre dans un mot le préfixe du mot qui la précède. On ne considérera comme séquences que les mots ainsi bien construits. Ceci amènera à considérer un mot comme l'ensemble de ses lettres.

Ainsi, on ne considérera pas le mot $m = \text{abbabba}$ et son facteur $m' = \text{bab}$, mais le mot $m = (a, \lambda)(b, a)(b, ab)(a, abb)(b, abba)(b, abbab)(b, abbabb)(a, abbabb)$ et son facteur : $m' = (b, ab)(a, abb)(b, abba)$. Une lettre dans la $k^{\text{ième}}$ case de la pile a donc la complexité k .

Les parties d'un mot sont les réunions de facteurs (on verra en commentaire les limitations en cardinalité), en particulier les réunions de lettres, comme annoncé.

Une lettre a est désormais un couple (a, m) , qui ne peut être décomposé par aucune instruction du langage.

On considère le langage d'instructions élémentaires

$\text{empiler}(m)$, $\text{fragment}(k, k')$, où m est un facteur au sens précédent et k et k' des entiers.

La séquence m peut être identifiée au programme $\text{empiler}(m)$. En fait, $\text{empiler}(m)$ revient à ajouter les lettres de m à l'ensemble des données, et $\text{fragment}(k, k')$ revient à supprimer toutes les lettres indicées par des préfixes de longueur supérieure à k' ou inférieure à k ($\text{dépiler}(k)$ s'identifie à $\text{fragment}(0, k)$).

Propriétés

On vérifie successivement :

- Deux lettres différentes sont indépendantes (évident).
- Tout mot est incompressible, car il n'a pas deux lettres identiques.
- Le programme le plus court transformant fau en fbu' est " $\text{dépiler}(l f l)$; $\text{empiler}(\text{bu}')$ ", qui est de complexité $l \text{ bu}' l$ (à un log près).
- On peut associer à tout ensemble fini de mots une décomposition booléenne unique, conformément aux résultats généraux. C'est à dire que l'on partitionne un ensemble de mots par clôture ensembliste.

Exemple : On part de $m = \text{uavaw}$; $m' = \text{uavbw}'$ et $m'' = \text{ubv}''$. On voudrait décomposer :

$\{m, m', m''\}$ comme suit : $m = (u, av, aw)$; $m' = (u, av, bw')$; $m'' = (u, bv'')$, avec :

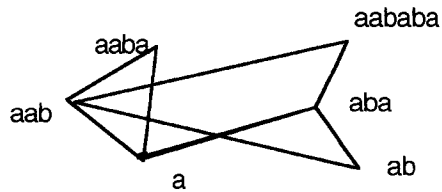
$u = m \cap m' \cap m''$; $av = m \cap m' - m''$; $aw = m - m'$, etc....

Ceci revient à écrire l'arbre partageant les préfixes et à considérer comme décomposition les facteurs entre les noeuds.

Le programme transformant au en bu est court, car a est identifié à (a, λ) , qui est de complexité indépendante de u . Pour palier ce fait, on suppose que tous les mots sont précédés d'un même nombre de symboles de fond de pile, assez grand (par exemple le sup des longueurs des mots considérés). Les fonctions de défaut sont données uniformément, et pour tout ensemble fini considéré, on peut se fixer un fond de pile "assez grand".

On a un exemple où tout ensemble fini d'objets (les mots) s'organise de façon unique (à des transformations simples près) en une "base de connaissances" où chaque élément peut être

considéré comme atomique. Ici, on peut donner des noms symboliques aux facteurs entre les noeuds et les manipuler comme des symboles.



Remarque sur l'implémentation

Le formalisme adopté ci-dessus est lourd et non opératoire, mais son utilité est d'établir des propriétés, pas d'être implémenté tel quel. On obtient comme résultat qu'en considérant le ruban d'entrée comme une pile que l'on décompose et compose en fragments, on a une manipulation booléenne des mots en posant $uav - ubv' = av$, l'intersection étant alors u . L'implémentation directe de ces propriétés suffit.

Exemple 2. : Le filtrage et l'unification dans les arbres.

Se traite comme le cas des mots. Un développement soigneux resterait à faire.

Exemple 3. : Les ensembles

On écrit la fonction de défaut sous la forme $A \log(k) + B$.

Dans ce qui suit, on considère un langage qui manipule formellement les éléments et les parties finies d'un ensemble, fini ou infini. Plus précisément, tout élément a un nom, et toute partie finie est donnée en extension (la complexité d'un sous ensemble à n éléments est donc au moins n). Les noms sont formels et deux éléments donc indépendants. D'autre part, on considérera pour tout ensemble des sous-ensembles de parties qui s'énumèrent par un procédé simple, ce qui permet d'extraire une partie par un programme dont la longueur est de l'ordre du log du nombre des parties à énumérer.

Le cas fini. Nous allons voir qu'on retrouve artificiellement la situation classique de l'algèbre de BOOLE des éléments d'un ensemble. L'aspect artificiel provient de la nécessité de considérer une constante A fonction exponentielle de la cardinalité de l'ensemble.

Considérons un ensemble de n éléments. Le seul problème qui se pose est de faire coïncider la notion usuelle de partie avec celle que nous avons définie. Ceci implique de pouvoir extraire - donc nommer - les parties d'un ensemble par un programme court. Supposons pour simplifier que k est la complexité commune de tous les éléments; comme un sous ensemble de p éléments a 2^p parties, ceci impose que A soit exponentiel en tout p , donc en n .

Restrictions finies d'un ensemble infini. La donnée de la fonction de défaut fixe un n tel que toute partie finie puisse être considérée comme dans l'exemple précédent.

On pourrait ainsi dire que tout langage est *finement booléen*, en ce sens que toute restriction de cardinalité suffisamment petite peut être manipulée booléennement.

Dans cette approche, on a considéré toute la combinatoire d'ensembles réduits de manière drastique, puisque la cardinalité est en $\log A$. L'approche suivante est plus riche, elle considère une partie de la combinatoire de tout l'ensemble.

Parties à nombre borné d'éléments. Des instructions de longueur de l'ordre de $A \log n$ suffisent à extraire toute partie de moins de A éléments d'un ensemble à n éléments, puisque le nombre de ces parties est de l'ordre de n^A , et que la complexité d'un ensemble à n éléments est au moins n (car comme le langage L considère les éléments comme indépendants, tout sous-ensemble est donné en extension).

Ainsi, on peut toujours manipuler booléennement un ensemble infini d'ensembles ayant un nombre borné d'éléments, ce qui revient à considérer un nombre quelconque d'objets ayant entre eux des liens simples.

On peut trouver une situation analogue dans le cas non restreint de KOLMOGOROV, en considérant un ensemble de séquences ayant entre elles des liens simples. Nous détaillons cet aspect ci-dessous. Il est analogue au cas précédent.

Combinaisons de séquences incompressibles.

Soit un ensemble E de n séquences de longueur n , tel que deux parties disjointes de E soient indépendantes (pour établir l'existence, couper en n facteurs une séquence incompressible de longueur n^2 et prendre $A > 2$). On considère l'ensemble F des concaténées (dans l'ordre lexicographique) de moins de A séquences de E . On se retrouve dans le cas précédent, dans le cadre non restreint de la complexité de KOLMOGOROV, mais en restreignant la base d'objets.

On illustre ici que des ensembles non bornés d'objets peuvent être traités dans le cadre général de façon booléenne, si les relations qu'ils ont entre eux sont particulières.

Remarque sur le lien booléen ensembliste.

L'idée de l'axiomatique des langages booléens que nous définissons est de manipuler chaque objet non pas comme un programme, mais comme une information, et de décomposer cette information en sous informations comme on décompose un ensemble en sous ensembles. On construit ainsi les relations entre objets comme on comparait des parties d'un ensemble donné, chaque élément étant "atomique", sans lien avec les autres. Mais il y a une différence importante avec l'axiomatique ensembliste, qui a trait à l'acception réduite que nous donnons à la notion de partie. nous avons d'ailleurs hésité à conserver ce mot, hésitant à introduire le

Langages booléens de représentation.

néologisme "portion" ou "morceau". Le nombre des parties d'un ensemble est classiquement exponentiel en la cardinalité de l'ensemble, alors qu'il est pour nous polynomial. Le point de départ est qu'une partie d'un objet doit, pour nous, être extraite par un programme simple (court) de l'objet; la borne logarithmique de la longueur de ces programmes, que nous nous imposons par cohérence avec les résultats généraux de la théorie de la complexité, borne polynomialement l'ensemble des parties d'un objet.

Cette approche, qui peut paraître choquante, nous semble défendable. Dans la pratique, on ne considère jamais les 2^n parties d'un ensemble à n élément dans un contexte de représentation et de classification de connaissances.

Partie 1 - Chapitre 4.

Prétraitement numérique et compression. Représentations exactes et approchées.

où l'on introduit la notion de langage de description approximative

Dans ce paragraphe, nous introduisons la notion de langage de description approximative. Nous avons comme arrière pensée d'introduire de tels langages qui soient booléens. Ce dernier point fera l'objet du paragraphe suivant. Nous ne nous préoccupons pas ici de cette propriété.

1. Généralités

Une donnée subit un premier traitement de bas niveau (niveau physique, traitement du signal), qui l'approxime par une donnée plus simple (un réseau neuronal peut être vu comme un algorithme qui approxime un objet par un attracteur, qui est un objet de complexité localement minimal). Cette approximation doit conserver les propriétés sémantiques des objets. On concrétise ici cette proximité sémantique par l'introduction d'une distance sémantique.

Ces problèmes sont largement étudiés en traitement du signal. Nous les esquissons en partie dans notre cadre, en privilégiant le langage de description des approximations.

Nous concrétisons dans ce qui suit plusieurs situations.

La plus simple et la plus courante est celle où on approxime un objet par un objet plus simple, décrit dans un langage qui permettra son traitement au niveau sémantique, sans s'imposer aucune sorte d'unicité ou d'optimalité de l'approximant.

Nous donnons ensuite des conditions traduisant diverses notions de minimalité et d'unicité pour les approximations.

Représentation par approximation dans un langage L

1. Définitions:

Soient:

- un ensemble S des éléments que l'on veut représenter, muni d'une *distance sémantique* d.
- un *langage* restreint L (on omettra l'indice L dans la suite) de *représentation* des approximations, muni de la distance E computationnelle de ZUREK (restreinte à L).

(S,d) sera l'*espace (sémantique)* représenté par L. (S,d) sera *r - Représentable par Approximation dans L* (rRAP-L, ou plus simplement rRAP) si pour tout s, il existe s' représentable dans L (i.e. décrit par un programme de L), tel que $d(s,s') \leq r$.

On dira que la représentation est *effective* (rERAP-L) si on dispose d'un algorithme qui, pour tout s, associe un tel s'. Notons que dans ce cas, s' est forcément plus simple (au sens absolu) que s, à la longueur du programme d'approximation près.

Exemple

Sémantique du calcul: Si on prend pour sémantique les calculs eux-mêmes, c'est à dire si on prend pour d la distance de ZUREK non restreinte, alors on a la propriété RAP pour tout r. En un mot, dans ce cas dégénéré, la propriété est triviale.

2. Continuité, minimalité et effectivité de la représentation

On appelle *module de continuité* d'une r - représentation une fonction $\Gamma(r, d, k)$ telle que, pour tout $r' \geq r$, toute r' - représentation s' de s, t' de t, on a $E(s',t') \leq \Gamma(r', d(s,t), K(s',t'))$. (K est relative à L. Si on ne s'intéresse qu'à $r=r'$, on notera $\Gamma(d(s,t), K(s',t'))$).

Γ existe toujours. Il suffit de prendre $\Gamma(r', d(s,t), K(s',t')) = K(s',t')$. On aura donc toujours $\Gamma(r', d(s,t), K(s',t')) \leq K(s',t')$. Mais plus Γ sera petite, plus deux objets sémantiquement proches seront proches au sens de ZUREK dans L.

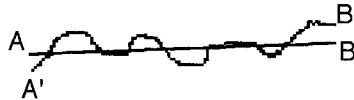
Une r - représentation s' de s sera *minimale* si, pour toute r - représentation s'' de s, $K(s') < K(s'') + O(\log K(s''))$. (On notera aussi *r - minimal*).

Une telle représentation existe évidemment. Nous étendons ici la notion de minimalité, qui s'entend à un log près. Quand on considère des représentations minimales, on peut considérer le *module de continuité pour représentation optimale*. De même, si on prend une représentation effective, on considère le *module de continuité pour représentation effective*.

Exemple

Sémantique du calcul: (suite).

L'identité fournit une représentation pour la quelle on peut prendre $\Gamma(r', d(s,t), K(s',t')) = d(s,t)$. Les représentations minimales sont non effectives. Les segments sont des minima. Ceci repose sur le fait que joindre deux points par un segment est de complexité négligeable. Le segment AB a donc la même complexité que le couple (A,B), dont la complexité minore toute figure ayant des "points extrêmes" A' et B' proches de A et B.



2. Étude de cas: dessins au trait sur une rétine

On suppose la rétine potentiellement infinie. On peut se contenter d'une conception intuitive du dessin au trait. Nous renvoyons à l'annexe 1 de cette partie pour plus de précisions.

On prend pour d la distance de HAUSDORFF entre les ensembles, associée à la distance Euclidienne entre les points (sauf indication contraire).

Ce choix signifie que l'on considère comme sémantiquement proches les figures visuellement proches.

Nous allons passer en revue les propriétés pour divers langages de représentation.

1. Deux exemples de langages de représentation

1. Le cas non restreint:

On prend pour L tous les programmes possibles. La propriété RAP est trivialement vérifiée : pour tout r , tout figure s'approxime par elle même. La représentation minimale n'est pas effective. On se trouve avec un module de continuité de l'ordre de grandeur le plus élevé possible, c'est à dire dans un cas où des figures très proches ont des approximaux minimaux très éloignés. Plus précisément, il existe $a > 0$ tel que $\Gamma(r', d(s,t), K(s',t')) > aK(s',t')$, comme le montre l'exemple suivant (cette inégalité est à rapprocher de l'inégalité $\Gamma(r', d(s,t), K(s',t')) \leq K(s',t')$, toujours vérifiée).

(les figures V et W sont présentées séparément mais V est incluse dans W sur la rétine).

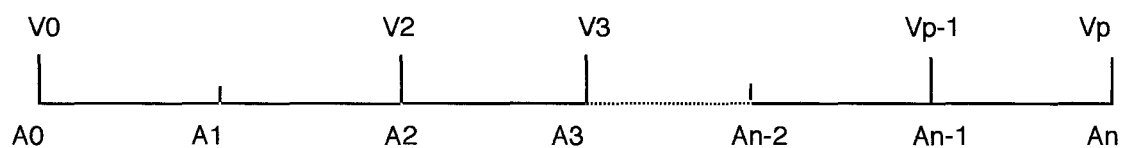


Figure V: le segment A_0A_n a la longueur $4nr$. Les A_i sont espacés de $4r$. Pour chaque i , V_i est à distance $2r$ de A_i . Pour tout i , on "tire à pile ou face" si V_i figure ou non (la suite des occurrences est donc incompressible).

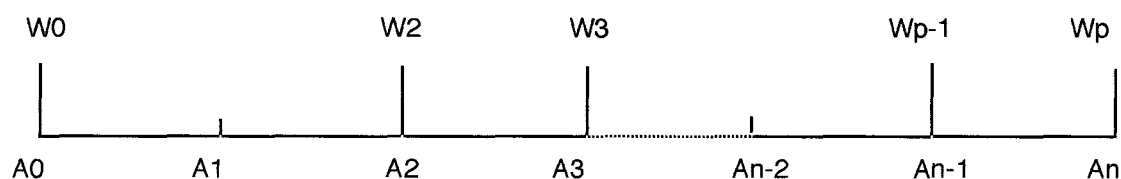


Figure W: elle est déduite de V en rallongeant les segments A_iV_i . W_i est à distance $3r$ de A_i .

Soit B le milieu de A_0V_0 et C le milieu de A_nV_p . Parmi les figures à distance au plus r de V, BC est de complexité minimale. La complexité de BC est de l'ordre de $\log n$, alors que celle de V et celle de W sont de l'ordre de p , c'est à dire de $n/8r$ (on prend r petit par rapport à n). Étant donnée la distance des W_i au segment A_0A_n , il est facile de vérifier que toute figure à distance au plus r de W "conserve l'information sur la répartition aléatoire des W_i ", et a donc une complexité au moins de l'ordre de $n/8r$. En résumé, on a $d(V,W) = r$, V et W ont une complexité en n , BC fournit un r approximant r -minimal V' de V, de complexité en $\log n$, alors que tout approximant r -minimal de W a une complexité de l'ordre de n . D'où le résultat.

2. Approximation par segments

Nous nous intéressons ici à l'approximation de dessins au trait par des segments, ce qui correspond à une pratique courante. La notion de distance considérée est toujours celle de HAUSDORFF.

Précisons le langage L de représentation choisi. Les instructions spécifiques sont

$A := \text{Point}(x,y)$: repère le point A de coordonnées x, y .

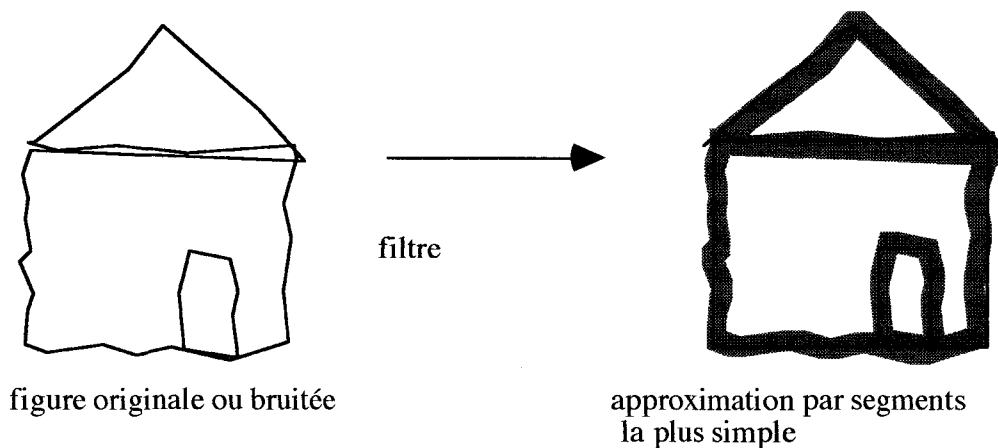
$\text{Translation}(A, x,y)$: translate le point A de (x,y) .

$\text{Segment}(A,B)$: trace le segment.

La complexité des instructions est la longueur qu'il faut usuellement pour les écrire, sans compression sur la représentation des nombres. C'est à dire que la complexité de Point et Translation est en $\log x + \log y$. Une figure représentable est donc faite de points et de segments.

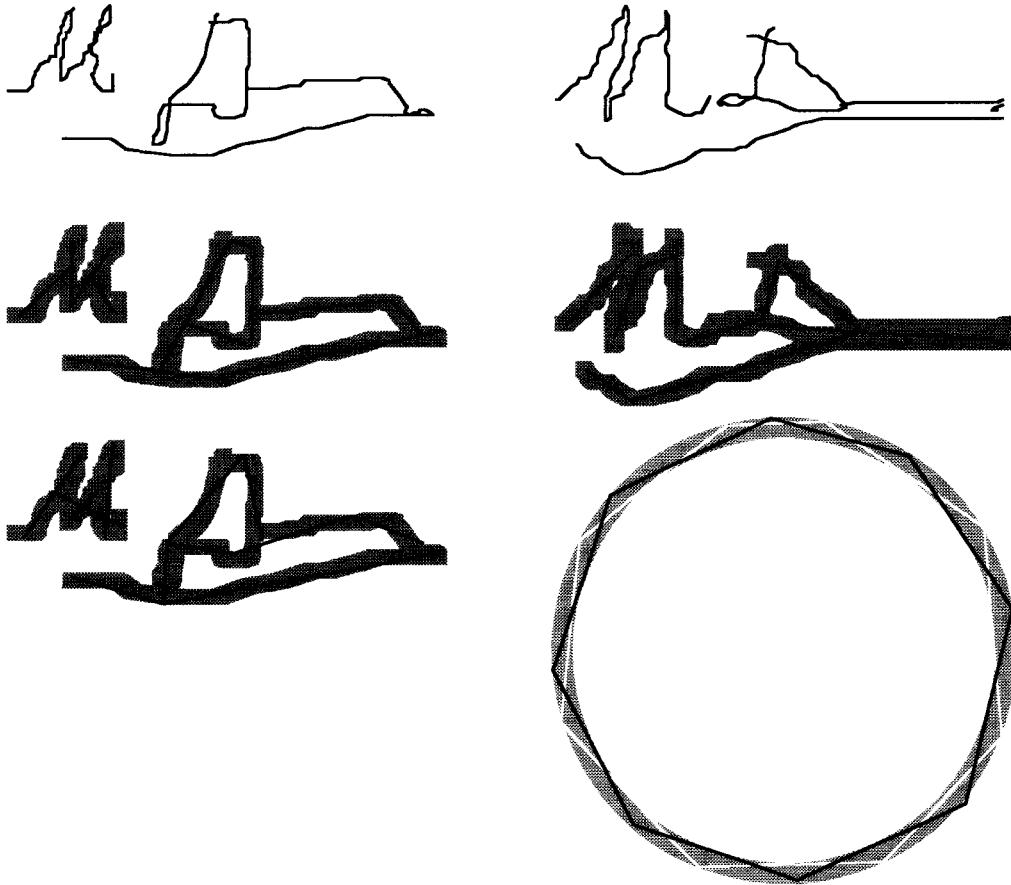
Tout dessin de la rétine est représentable, même exactement, au besoin point par point; on a donc pour tout r la propriété rRAP. Pour tout r , la recherche d'un approximant minimal est effective, car tous les programmes de L s'arrêtent. Par contre, même dans des cas simples de dessins au trait, elle est complexe en temps. On montre par exemple en annexe que la détermination du nombre minimum de segments est NP-complète. Dans la pratique, on considère des heuristiques d'approximation par segment (voir chapitre 6.3 de la thèse). Quelque soit le procédé choisi, le problème de la continuité est pratiquement très important: il s'agit d'avoir des représentations proches au sens du calcul dans L d'objets proches dans le point de vue adopté. On peut considérer le problème de l'unicité de représentation comme un cas limite de la continuité, où le module de continuité est très petit. Il est de même important de disposer d'une forme canonique.

La figure ci dessous illustre la situation idéale: un dessin admet une approximation minimale qui est unique (à de petits déplacements près des extrémités).



Malheureusement, cette situation est exceptionnelle, ce qui est empiriquement prévisible: on touche au problème de la robustesse de représentation de connaissances.

L'exemple des signatures illustre la difficulté: une même signature a plusieurs approximations minimales qui ne se déduisent pas l'une de l'autre par de petits déplacements des extrémités, deux variantes de signatures s'approximent par des lignes brisées très différentes. Plus simplement, la multiplicité des "quadratures du cercle" est une autre illustration.



En restreignant suffisamment l'espace des objets considérés, on peut se retrouver dans la situation idéale de la maison. Prenons l'exemple très simple de la famille des triangles et des quadrilatères. Chaque triangle, même si il est déformé (tremblement, parasites) à r près, sera approximé par trois segments et aura la complexité de son triplet de sommets. Si on déforme un peu le triangle, les sommets sont translatés par des transformations courtes, donc les représentations dans L sont proches en termes de calcul. il en est de même pour les quadrilatères. Quand un quadrilatère dégénère en triangle, les deux sommets proches deviennent proches par le calcul, et la continuité est conservée.

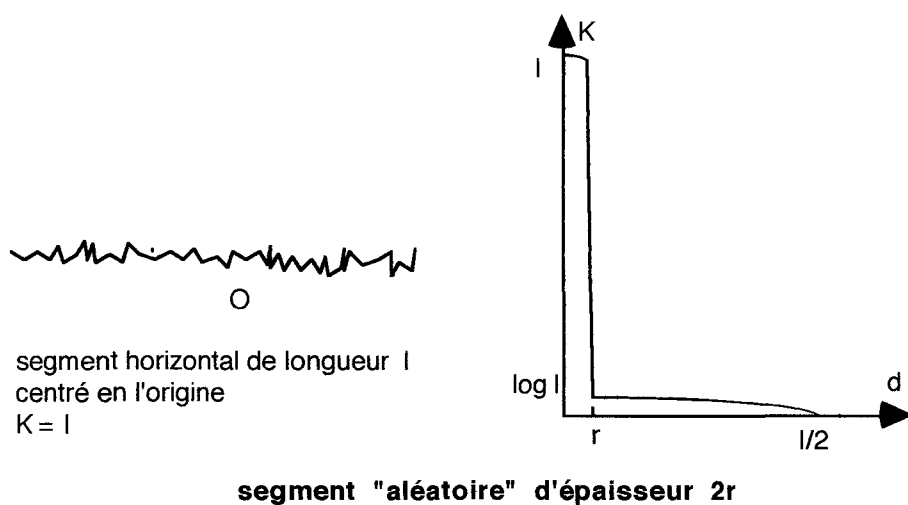
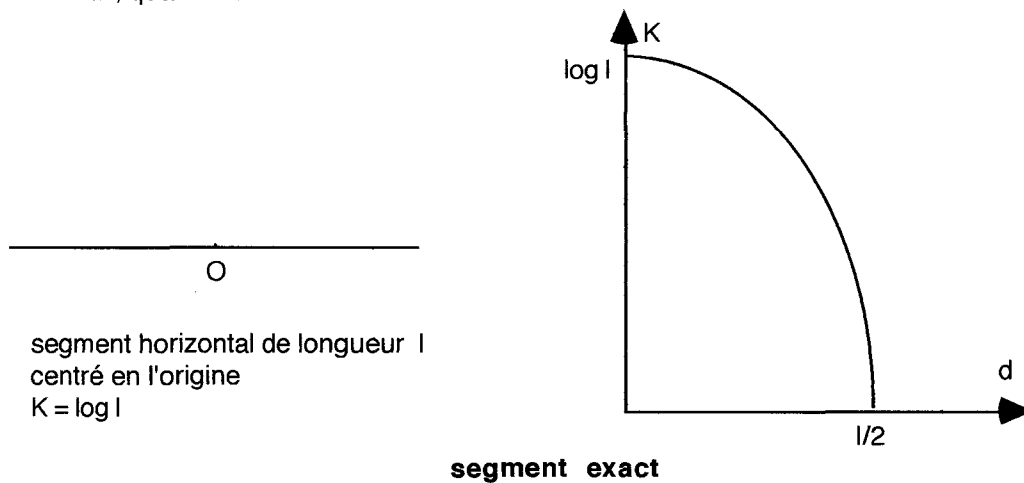
On peut prendre des restrictions moins drastiques. On peut considérer toutes les figures obtenues par perturbation (à r près) d'ensembles de segments, en s'imposant des contraintes géométriques sur les croisements, les angles et les bifurcations, afin d'éviter le genre de situation rencontré par exemple dans la construction de l'annexe. Il suffit que les angles soient "assez droits" ou les segments "assez longs". Nous ne détaillons pas ici l'étude géométrique, qui permet de bien résoudre des cas "simples" mais par exemple pas le problème de la comparaison de signatures.

2. Niveaux d'organisation d'un dessin au trait.

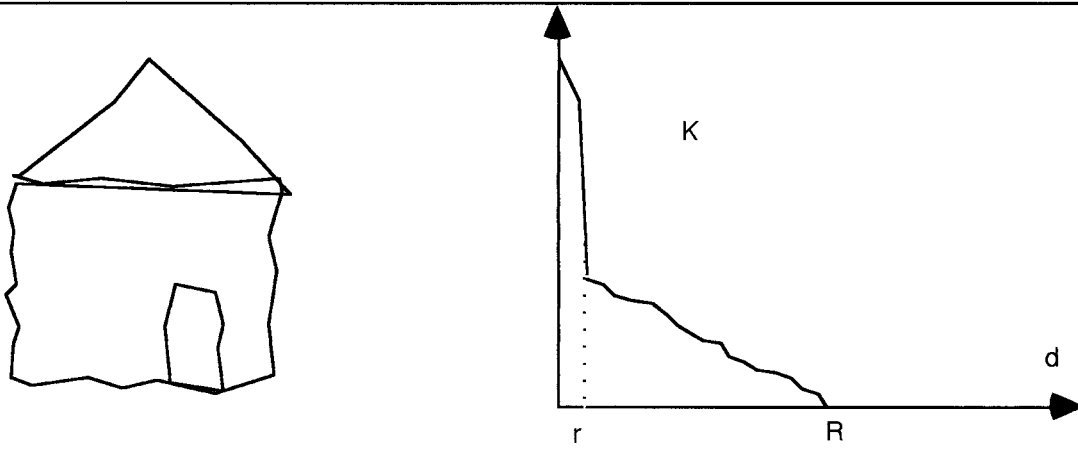
Nous esquissons ici à partir d'exemples ce qui peut aider dans un dessin au trait à fixer le "bon niveau d'analyse", un peu comme on règle la distance focale d'une caméra.

Pour tout procédé de représentation approximative dans un langage L, on peut associer à chaque figure une fonction qui, à toute distance d associe la complexité minimale d'un approximant de la figure à d près (ou de l'approximant à d près défini par l'heuristique). Les niveaux d'organisation du dessin sont les d pour lesquels la complexité s'effondre.

Cette approche procède un peu de la même idée que celle de CHAITIN, qui a proposé une autre notion de niveau d'organisation d'objets de dimensions 2 (ou 3). CHAITIN procède comme suit: pour tout quadrillage de la rétine (carrée) par des $n \times n$ carrés, il définit la complexité K_n d'un objet comme la somme des complexités des portions d'objets dans chaque carré. K_n croît avec n . Par exemple, si un objet est constitué de la réplique dans chaque cadran d'un objet élémentaire, $K_2 = 4K_1$. On voit que notre notion et celle de CHAITIN sont incomparables. Pour du dessin au trait, la définition de CHAITIN fait tendre K_n vers la longueur totale du trait, quand n croît.

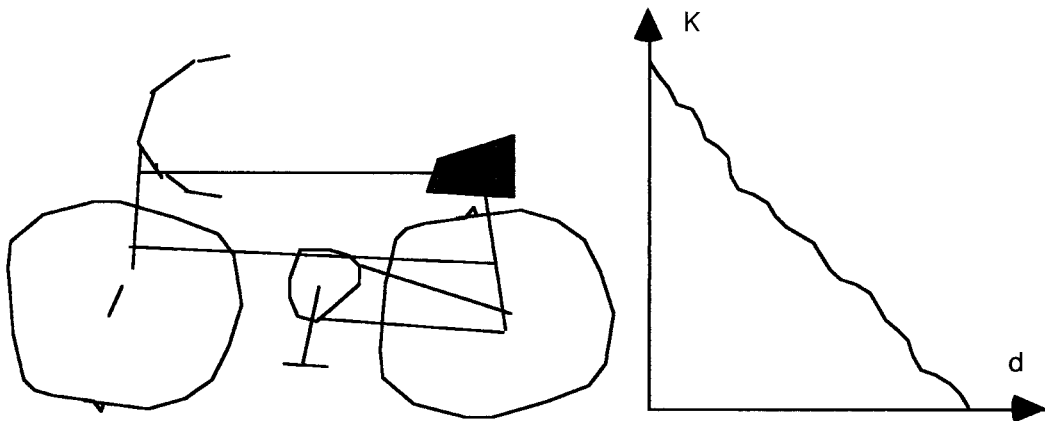


Un segment exact n'a pas de niveau d'organisation particulier. Sa forme est invariante par approximation. Un segment perturbé dans une amplitude r a ce seuil r comme niveau d'organisation.

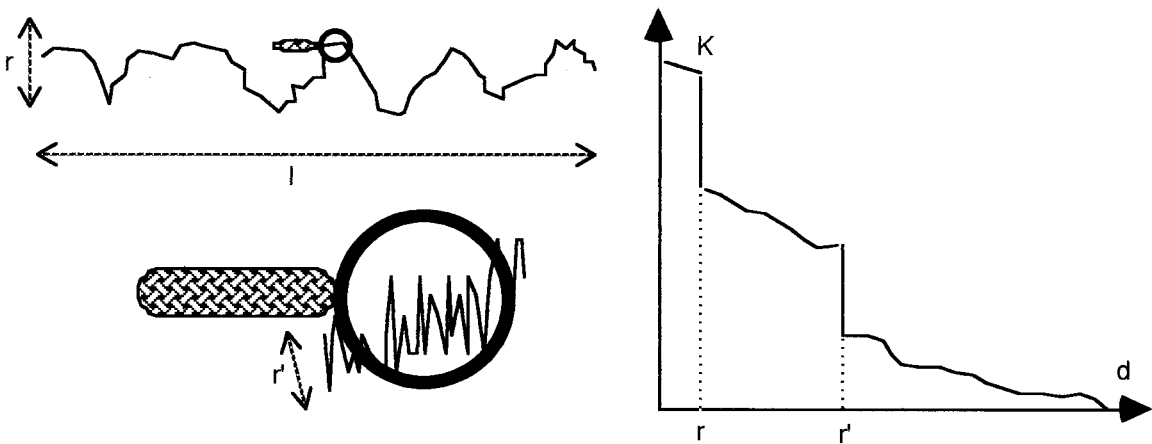


Maison de diamètre R, bruitée sur une épaisseur r.

Comme pour le segment, c'est ici l'amplitude du bruit qui détermine le niveau d'organisation.



Un vélo n'a pas une échelle de structure marquée.



un segment présentant deux niveaux de perturbations.

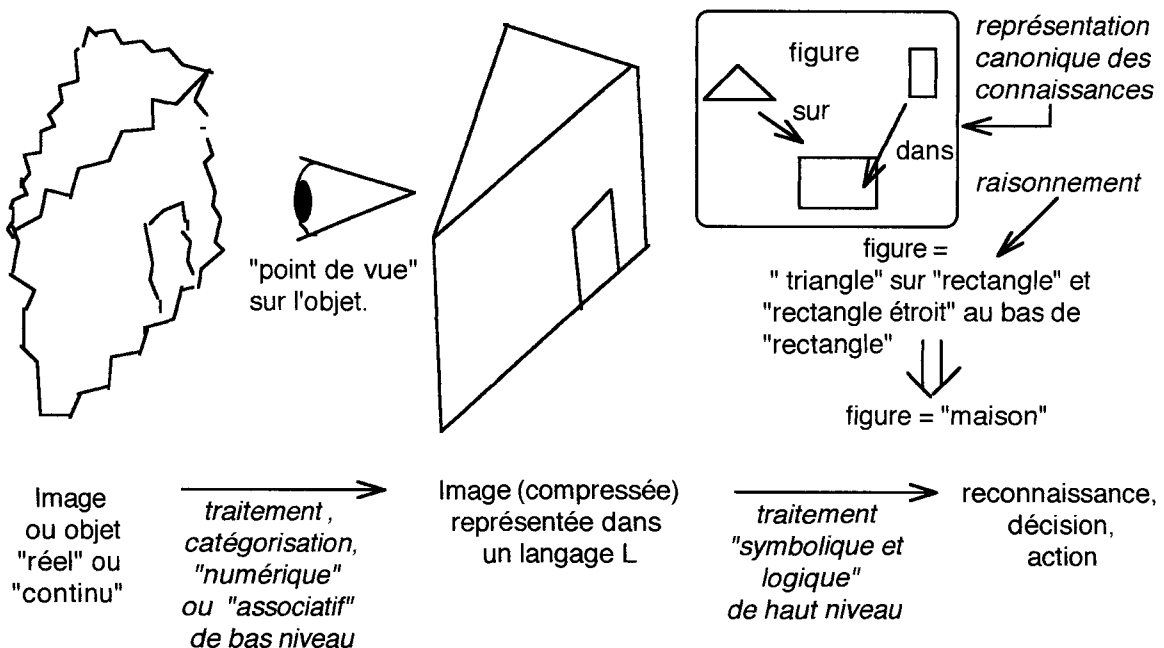
Partie 1 - Chapitre 5. Complexité et traitement numérique - symbolique.

Une première approche.

où l'on illustre d'exemples et précise des qualités souhaitables.

1. Généralités

La figure suivante, illustre, dans le cas du dessin au trait, une idée - simple ou simpliste - que l'on peut se faire d'un traitement numérique - symbolique. Nous nous plaçons dans le cas simple, séquentiel, où il n'y a pas de rétroaction entre les deux étapes.



Nous précisons la figure dans le cadre de notre approche par la complexité, ce qui illustrera les points essentiels de notre démarche. Rappelons les prémisses de notre travail, afin de préciser ce que nous entendons par "numérique", "symbolique", "logique".

1. Traitement numérique, traitement symbolique et booléen.

On s'accorde à donner un sens qui semble clair à la dichotomie numérique - symbolique. Le numérique traite du continu, manipule des nombres, le symbolique traite de symboles, manipule des piles. Ou encore, on associe traitement de bas niveau et numérique, traitement de haut niveau et symbolique. Comme le remarquait Jean-Paul DELAHAYE, on a envie de dire que le numérique est ce qui se fait en Fortran et le symbolique en langage à objets. Quand on associe un module neuronal de pré traitement d'images à un module à base de règles, on parle de système hybride, on qualifie le premier traitement de numérique et le second de symbolique.

On sent bien derrière ces approches que la frontière est plus floue qu'il n'y paraît. En effet, on peut dire aussi - et on le dit - que tout calcul par ordinateur est symbolique, en ce sens qu'il se ramène à des manipulations de symboles (ultimement des bits d'information). Mais on peut aussi dire que tout est numérique, puisqu'on traite des 0 et des 1. Reprenons l'exemple d'images. Supposons que l'on ait des photographies à classer entre portrait, paysage, puis plus finement portrait d'homme, femme ou enfant, paysage champêtre ou citadin, puis plus finement encore selon les époques etc. On va d'abord digitaliser les images, opération numérique (on passe du "réel", du "continu" au discret). On finit par des catégorisations symboliques, quand il s'agit de reconnaître ce que représente une photo. Entre les deux, les choses sont moins claires: attribuer un sexe à une silhouette humaine peut se faire tantôt au niveau traitement numérique ou neuronal, tantôt au niveau symbolique (reconnaissance de traits caractéristiques ou d'accessoires vestimentaires).

Après tout, une claire catégorisation numérique - symbolique n'est pas un impératif, son absence n'empêche pas de "faire". Nous ne prétendons d'ailleurs nullement résoudre ce problème. Nous éclairons seulement une partie du problème sous l'angle de la complexité de KOLMOGOROV, et proposons une approche qui peut aider à comprendre en quel sens, dans quels domaines, il est légitime de faire des raisonnements booléens sur des objets, et quelles sont les limitations de ces domaines.

La complexité de KOLMOGOROV contribue à fonder le traitement algorithmique de l'information dans toute sa généralité, et des mathématiciens, des physiciens, y travaillent. Nous avons vu qu'on ne peut pas espérer dans ce cadre général parler de l'information commune à deux objets, de l'information propre à chacun.

Nous adoptons ici le point de vue de l'informaticien qui n'a qu'un domaine restreint à étudier et classifier. On retrouve les problèmes classiques de représentation de connaissances: quel vocabulaire de description pertinent prendre, quelles classes définir, etc.... Tant qu'on reste dans des problèmes jouets, il y a des solutions triviales. Les vraies difficultés viennent avec les ... "vrais problèmes", c'est à dire les problèmes de grande taille (en biologie, en génétique, en chimie par exemple). Et c'est alors qu'il est important de savoir dans quelle mesure tel type de représentation ou de raisonnement est valide. Est - il biaisé lors de la saisie

de l'information (parasites dans le traitement de bas niveau, erreur de saisie) ou lors de la classification (problèmes de catégorisation, erreurs de raisonnement)?

Dans notre approche, nous introduisons la notion de langage de description "certifié" booléen. Ceci force à bien expliciter ce que l'on fait lors de la définition de ce langage. Déterminer un langage booléen, c'est choisir et restreindre son point de vue. Moyennant quoi le langage fournit une description intermédiaire précieuse du problème : à tout objet "réel" noté o , le traitement numérique associe une représentation $R(o)$ dans le langage L . La représentation $R(o)$ est alors traitée en termes de calcul de prédicats. Toutes les manipulations, inductions, déductions à partir de $R(o)$ sont alors certifiées exactes par rapport au modèle. La conclusion ne peut être entachée d'aucun biais provenant par exemple d'une corrélation cachée entre deux propriétés manipulées comme indépendantes. Ainsi, tout problème d'explication se pose clairement: les problèmes liés au passage non booléen de o à $R(o)$ sont clairement identifiables par rapport aux problèmes liés aux manipulations.

2. Exemple

Nous illustrons cette démarche sur un exemple jouet, trivial comme il se doit. Mais notre approche est générale et permet une puissance d'abstraction pour L que l'on ne peut atteindre sans fil conducteur théorique. L'illustration de cette affirmation se fera dans la seconde partie du chapitre, consacrée au dessin au trait.

On considère un ensemble de silhouettes de véhicules. Notre but est de les classer en 3 catégories : *BE*rline, *MO*nospace, *UT*ilitaire. On se donne les règles:

+de4roues \Rightarrow *UT*ilitaire

FaceGauche-Rectiligne et FaceDroite-Rectiligne et non +de4roues \Rightarrow

*MO*nospace

En abrégé, on écrira

+de4 \Rightarrow *UT* et $FGR \wedge FDR \wedge \neg +de4 \Rightarrow$ *MO*

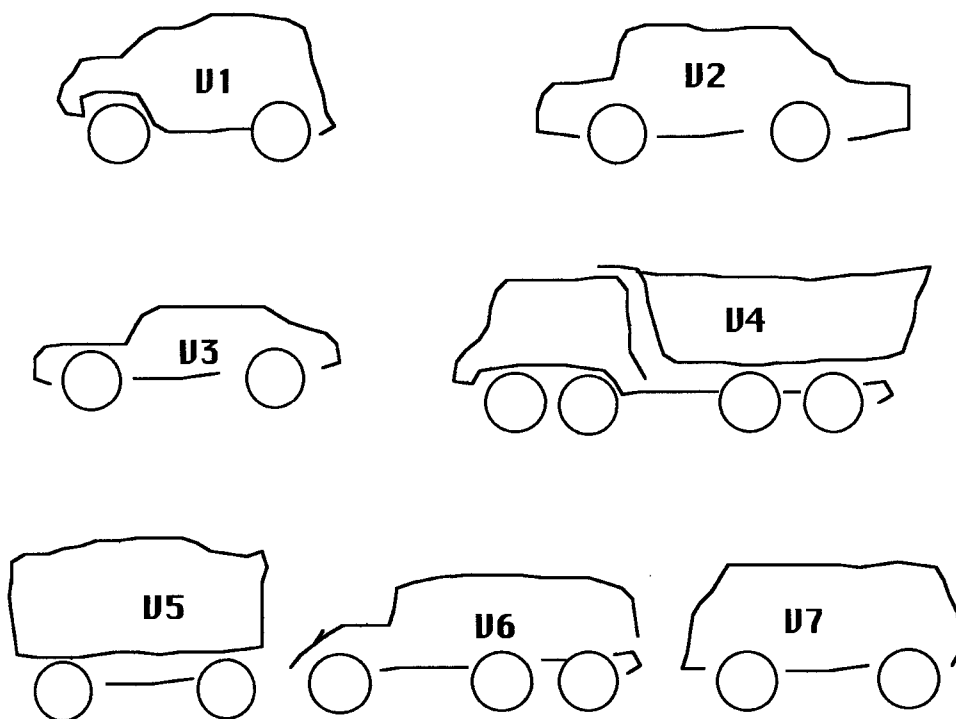
Ce système est cohérent ; on peut le compléter par

$\neg +de4 \wedge (\neg FGR \vee \neg FDR) \Rightarrow$ *BE*

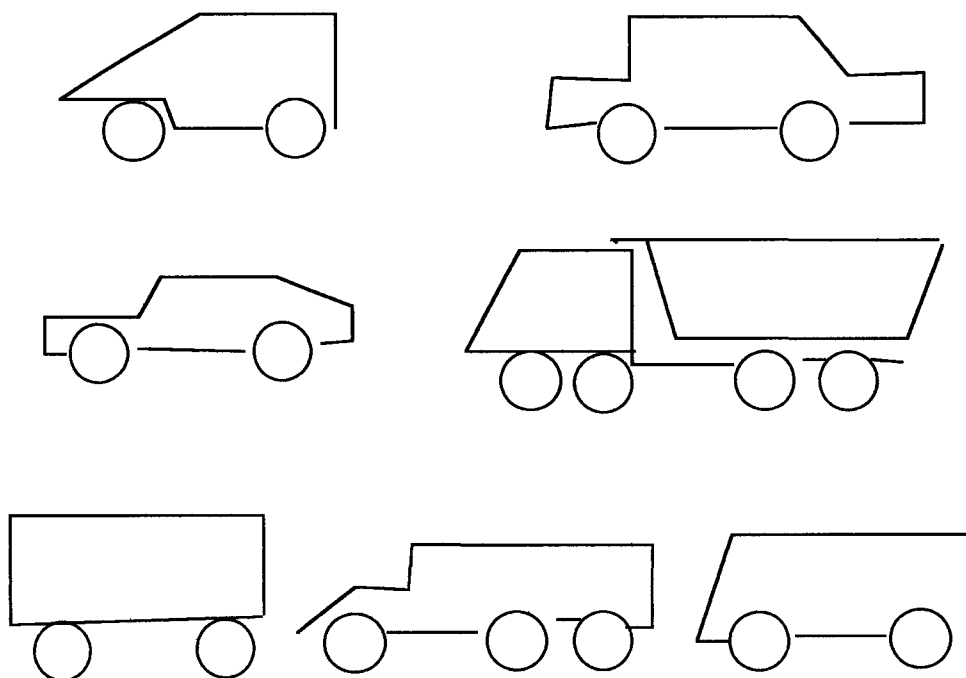
Traitement numérique.

Nous décomposons ici ce traitement en 3 étapes. Selon les points de vue, on pourrait essayer de prendre l'une ou l'autre des descriptions intermédiaires comme langage booléen. Nous disons "essayer", car l'art devient ici de trouver un langage de description qui soit pertinent vis à vis du problème, et qui satisfasse l'axiomatique. Nous reviendrons sur ce point, que nous appellerons "qualités" du langage L .

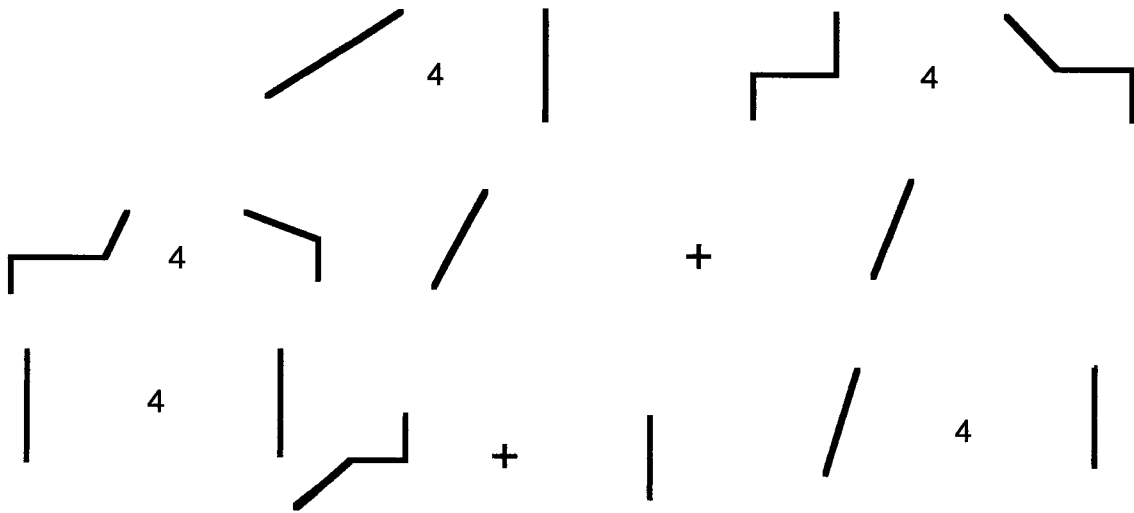
On suppose que l'on dispose de 7 silhouettes de véhicules.



La première étape consiste à approximer la carrosserie par une ligne brisée simple. (Il se pose ici des problèmes d'approximation, discutés dans le chapitre précédent.)



La deuxième étape analyse le nombre de roues et ne retient que les "face gauche" et "face droite". (Cette opération n'est pas sans poser de problème, que nous ne détaillerons pas ici ; c'est le genre de problème que pose la définition d'un langage de description).



La dernière étape associe enfin à chaque véhicule V_i sa représentation $R(V_i)$ dans un langage L qui est simplement celui des triplets de booléens $(FGR(V_i), FDR(V_i), +de4(V_i))$. On obtient trivialement pour L une structure booléenne en identifiant les prédicats à des ensembles.

Traitement booléen

On obtient $R(V1) = \{ FGR, FDR \}$, que nous noterons encore $R(V1) = (1,0,1)$;
 de même $R(V2) = (0,0,0)$; $R(V3) = (0,0,0)$; $R(V4) = (1,1,1)$;
 $R(V5) = (1,0,1)$; $R(V6) = (0,1,1)$; $R(V7) = (1,0,1)$.

D'où finalement $UT = \{V4, V6\}$ $MO = \{V1, V5, V7\}$ $BE = \{V2, V3\}$

Si on confronte dans l'exemple les résultats de la reconnaissance aux silhouettes données, deux cas sont sujets à caution : $V1$ et $V5$. Il faut d'abord remarquer que les silhouettes ne sont pas les véhicules, et que dans les deux cas la restriction aux silhouettes est source d'ambiguïté. Mais le niveau où le problème se pose diffère d'un cas à l'autre. Pour $V1$, le problème se pose dès la première étape du traitement numérique, lors de l'approximation de la face gauche. La question qui se pose est de savoir dans quelle mesure le profil $V1$ a été perturbé. Dans le cas de $V5$, il n'y a pas de faille dans le traitement. Lors de la définition du langage, donc de l'expertise, on a oublié le cas des remorques.

3. Qualités d'un langage de représentation booléenne relativement à un espace sémantique (E,d)

La qualité d'un langage de représentation qui nous intéresse tout particulièrement est évidemment d'être booléen. Cette contrainte implique des choix, et l'adéquation d'un langage booléen à un problème dépend de l'expertise. Si nous ne pouvons mettre en forme cette adéquation, d'autres qualités importantes peuvent être mises en lumière:

- La précision.

Dans les définitions des langages de représentation, plus r est petit, plus on peut représenter dans le langage un objet proche de l'objet réel au sens de d , c'est à dire du point de vue sémantique qui définit le problème. Si $r=0$, on a une représentation *exacte*.

-La (capacité d') analogie

Nous avons vu que plus le module de continuité est petit, plus des objets sémantiquement proches ont des représentants proches au sens de ZUREK dans L . Quand L est booléen, la distance de ZUREK entre deux représentations s' et t' est donnée par la complexité de la différence symétrique $s' \Delta t'$.

$$\Gamma(r', d(s,t), K(s',t')) = K(s' \Delta t') = K(s',t') - K(s' \cap t')$$

La fonction $A(r', d(s,t), K(s',t')) = K(s' \cap t')$ sera appelée (*capacité d'*) analogie de L .

Si, pour $s' \neq t'$, $K(s' \cap t') = 0$, la capacité d'analogie est nulle.

-La (capacité de) différentiation

Si on a toujours $K(s',t') = K(s' \cap t')$, la capacité d'analogie est maximale, mais le cas est complètement fallacieux: il signifie que tous les objets ont la même représentation, donc qu'ils sont tous identifiés. Ce cas n'est pas exclu par nos définitions, antérieures, mais il correspond à une (*capacité de*) différentiation de L nulle. Aussi appellerons nous (*capacité de*) différentiation la fonction $D(s',t') = K(s' \Delta t')$. On a évidemment

$$D(s',t') + A(s',t') = K(s',t').$$

On recherchera un langage L réalisant un bon compromis entre les deux.

-L'épaisseur

En linguistique, du point de vue des grammaires transformationnelles, on parle de structure superficielle et de structure profonde. Nous préférons ici le terme épaisseur, afin d'éviter la confusion avec la profondeur de BENNET en complexité. L'épaisseur prendra, de plus, un sens très concret dans le cas du dessin au trait.

D'une façon générale, Le langage L est défini comme un langage de programmation, c'est à dire à partir d'un ensemble fini d'instructions, qui peuvent toujours être les règles d'une machine de TURING, ou être définies à un plus haut niveau, comme nous le faisons dans nos

exemples. En ce sens, il existe un plus petit sous langage L' de L , dans lequel le domaine sémantique est représenté. L' sera appelé le *langage superficiel*, celui qui suffit à décrire les objets de départ.

Se placer dans le cas booléen est supposer que L est booléen, mais L' n'est pas nécessairement booléen. On pourrait appeler *épaisseur de L* le nombre de règles ajoutées à L' pour obtenir L . Si $L=L'$, le langage est donc sans épaisseur.

Cette notion d'épaisseur désigne l'enrichissement du problème de départ qu'on est amené à introduire pour réaliser un traitement booléen. En pratique, on pourrait dire qu'on essaie d'abord de trouver un langage de représentation L' , à partir de propriétés apparentes de l'univers sémantique, puis qu'on enrichit le langage de nouveaux concepts qui permettent un traitement booléen.

Nous illustrons ces qualités dans le cas du dessin au trait.

2. Le dessin au trait

Nous allons essayer de déterminer un langage booléen qui permette de parler d'information commune à deux dessins au trait, de différence symétrique... tout en ayant un bon compromis entre capacité de différenciation et d'analogie. Nous allons voir que ceci nous amène à "donner de l'épaisseur" au langage.

1. L'étape numérique : approximation par segments.

Dans la suite, nous supposons fixée une heuristique d'approximation par segment, dont nous supposons seulement qu'elle approxime par un petit nombre de segments (mais pas forcément le minimum). On sait qu'il existe des algorithmes très rapides pour cela. Nous y associerons un langage de représentation qui jouera le rôle du langage superficiel L' , et que nous épaissirons au besoin pour en faire un langage booléen "de qualité".

On peut se demander quel est l'intérêt de cette phase numérique. Pourquoi approximer? et de combien?

Certains arguments de réponse sont évidentes:

- La compression: l'expression obtenue est une compression de l'image d'origine, qui en change peu le sens, puisque l'espace sémantique est muni de la distance de HAUSDORFF.
- Le rayon d'approximation correspond au niveau (ou à un niveau) d'organisation vu plus haut.

Il importe de souligner un troisième argument, propre à notre problématique :

- Le fait de compresser, de représenter un objet par "un petit nombre" de segments, permettra de faire des manipulations booléennes sur un petit nombre de segments, donc d'accumuler moins de défauts en log, ou - ce qui est une autre façon de dire la même chose - de considérer des bases de connaissances booléennes d'ensembles plus nombreux d'objets.

2. Un langage booléen sans capacité d'analogie

La représentation par pixels

On prend le langage L' défini par les instructions

$A := \text{point}(x,y)$ de complexité $c + \log x + \log y$

$\text{segment}(A, B)$ de complexité constante c . (Cette instruction ne s'applique qu'à des identificateurs).

L' suffit à représenter tout objet par des segments. Il semble "naturel". L' ainsi défini est un langage booléen ; le caractère booléen s'identifie avec les opérations ensemblistes sur les pixels (l'intersection au sens du langage de description est l'intersection des pixels, etc.). La complexité d'un approximant est (aux constantes près), la somme des complexités des points qui interviennent (comme points ou extrémités de segments). Mais deux points distincts sont considérés comme indépendants, ce qui fait que la capacité d'analogie est nulle : on a $K(s' \cap t') = 0$ pour deux figures disjointes distantes de 1.

3. Un langage booléen sans capacité de différentiation

La représentation unaire

$P_{x,y}$ qui désigne le point (x,y) . On pose $K(P_{x,y}) = c + x + y$

$T((x,y), (p,q))$ est la translation qui associe au point $P_{x,y}$ de coordonnées (x,y) , le point

$P_{x+p,y+q}$. On pose $K(T((x,y), (p,q))) = c + p + q + \log x + \log y$

Les complexités posées signifient que tout se passe comme si on écrivait les coordonnées d'un point en unaire. Pour les translations, les coordonnées du point de départ sont là écrites en binaire classique, mais le vecteur translation est en unaire.

$T((0,0), (x,y))$ s'identifie donc à $P_{x,y}$ et $T((x,y), (p,q))$ à $T(P_{x,y}, P_{x+p,y+q})$

On considère aussi la composition de translations

$I(T(P_{x,y}, P_{x+p,y+q}), T(P_{x+p,y+q}, P_{x+p+r,y+q+s})) = T(P_{x,y}, P_{x+p+r,y+q+s})$

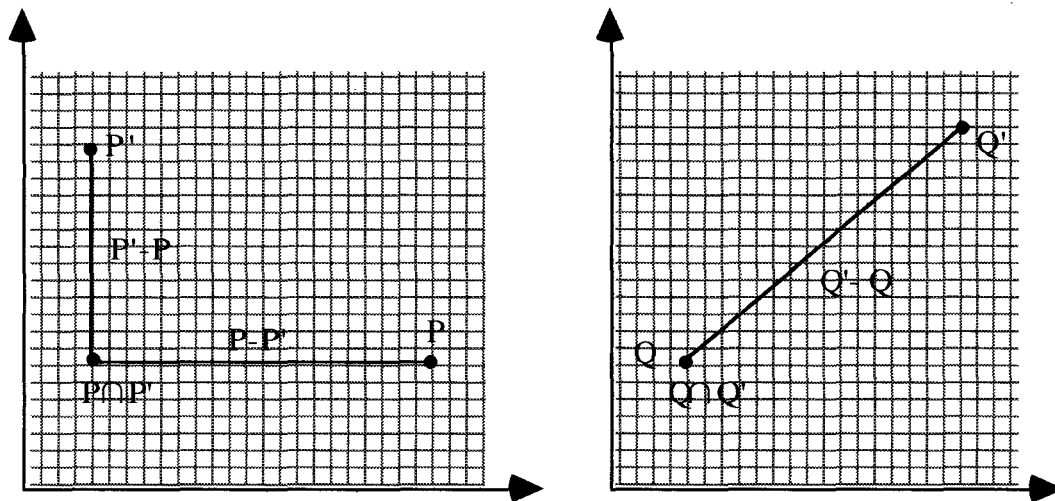
et, afin de satisfaire à la cohérence de l'information relative, les décompositions de translation :

$J_{r,s}(T(P_{x,y}, P_{x+p+r,y+q+s})) = \{T(P_{x,y}, P_{x+p,y+q}), T(P_{x+p,y+q}, P_{x+p+r,y+q+s})\}$

les indices r et s seront écrits en binaire, ce qui permettra de négliger la complexité de la décomposition relativement à la translation où elle s'applique.

Il est facile de vérifier que l'on a un langage booléen. L'idée est d'utiliser le fait que les notations binaires sont d'une complexité compatible avec la fonction de défaut, donc négligeables, par rapport aux notations unaires.

La figure suivante illustre la décomposition d'un ensemble de deux points P et P' en ($P \cap P'$, $P - P'$, $P' - P$). Nous écrivons en unaire avec des bâtons I .

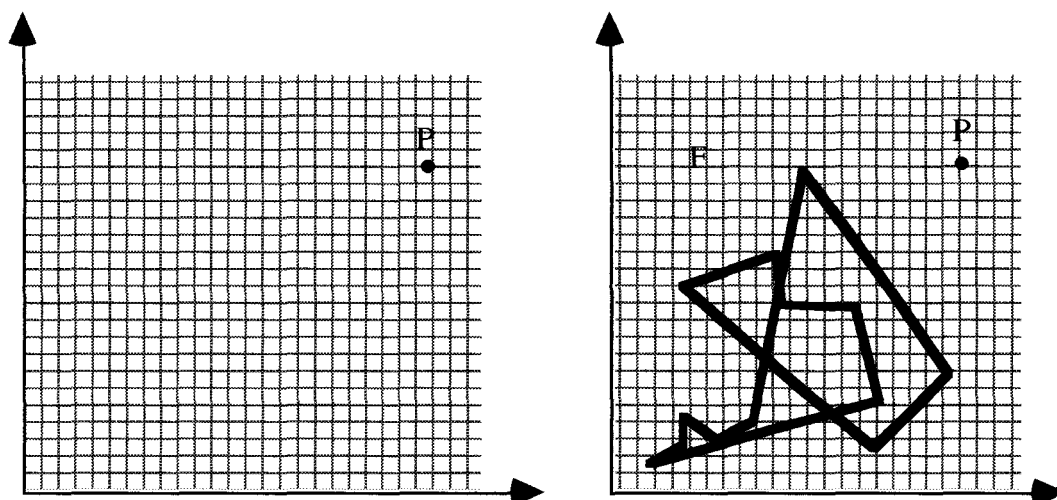


$P = P$	$ $	$ $	$ $	$P' = P$	$ $	$ $	$ $	$Q = P$	$ $	$Q' = P$	$ $	$ $	$ $
$P \cap P' = P$	$ $	$ $		$Q \cap Q' = Q$									
$P - P' = T((3,6), (,))$				$Q - Q' = T((3,6), (,))$	(négligeable)								
$P' - P = T((3,6), (,))$				$Q' - Q = T((3,6), (,))$									

On remarque que les opérations ensemblistes dans L n'ont pas d'interprétation allant de soi en termes de pixels.

La représentation par pixel était trop discriminante, en ce sens que deux figures très proches au sens de d peuvent avoir une partie commune vide. A l'inverse, la représentation unaire n'est pas assez discriminante, car nous allons voir que des figures très différentes peuvent être confondues au sens de E.

En effet, posons $P_{x,y} \leq P_{x',y'}$ ssi $x \leq x'$ et $y \leq y'$. Les décompositions permettent d'identifier un point P à toute paire {P,Q} avec $Q \leq P$. En itérant le procédé, un point P majorant une figure F se confond à $F \cap P$ (avec la restriction habituelle que la somme des complexités des petites transformations reste négligeable, donc qu'on en fasse pas un trop grand nombre).



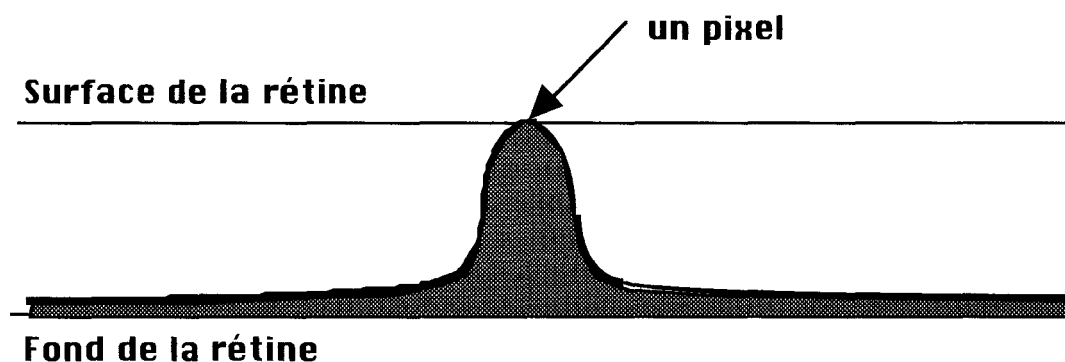
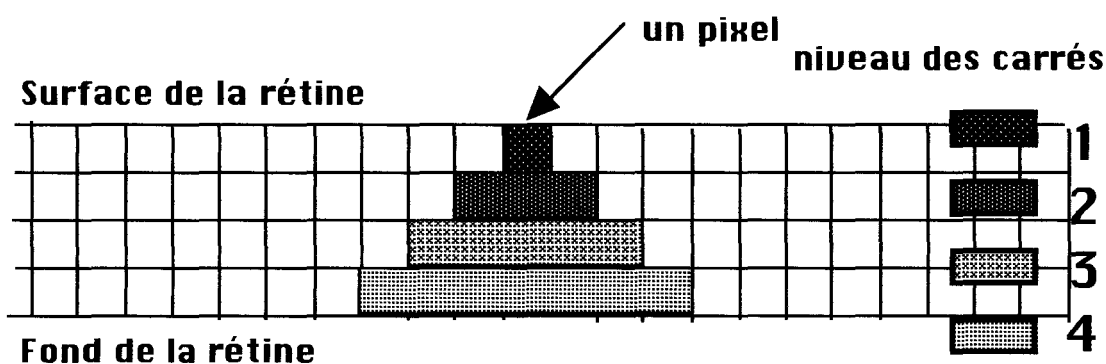
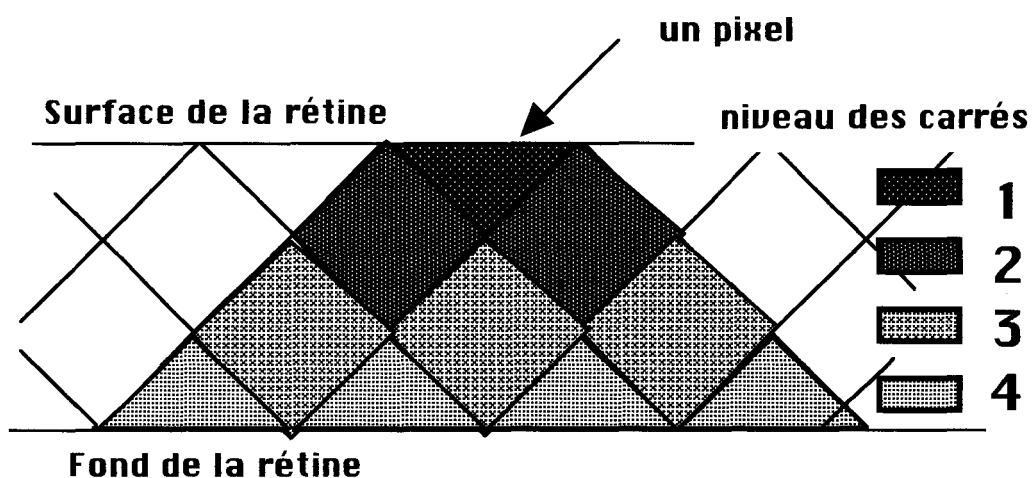
On voit qu'on obtient une très mauvaise différenciation: on a des objets à distance arbitrairement grande qui sont confondus.

4. Un compromis entre capacités d'analogie et de différenciation : représentation pyramidale.

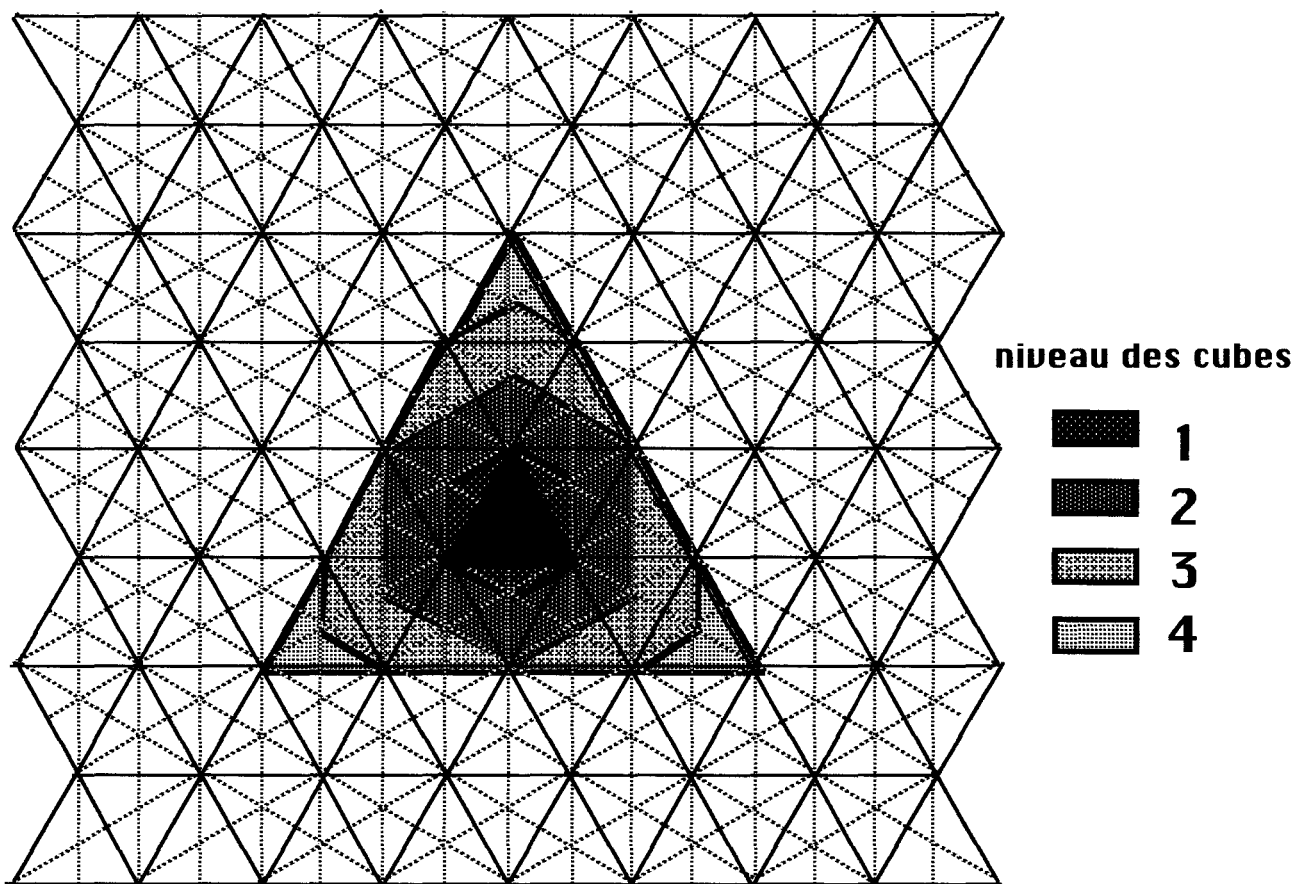
Nous esquissons ici ce que pourrait être un langage de représentation booléen qui réalise un bon compromis entre analogie et différenciation.

L'idée est la suivante : on considère une rétine ayant une certaine épaisseur faite de plusieurs couches de pixels. Le dessin est représenté par un sous ensemble de pixels de la couche superficielle, que l'on peut appeler surface de la rétine. Pour une rétine classique, il n'y a que cette couche. Le langage superficiel L' décrit les dessins de cette couche. Mais à chaque pixel on va associer aussi un ensemble de pixels des couches profondes, qui sera décrit par un "épaississement" du langage L' en le langage L . Le langage L sera basé sur la manipulation de ces ensembles pixels comme des ensembles usuels, ce qui fondera le caractère booléen du langage. On travaille maintenant en dimension trois et on peut associer un volume à chaque pixel, et prendre ce volume comme unité. En première approximation (i.e. en faisant l'impasse sur les problèmes de noms de pixels), la complexité sera mesurée par le volume. L'ensemble des pixels de toutes couches que l'on convient d'associer à un pixel M de la couche superficielle aura une forme plus ou moins pyramidale, pour des raisons que nous donnerons ensuite. Appelons *Pyramide* (M) cet ensemble. On associe à toute figure F superficielle l'ensemble *Pyramide* (F) fait de la réunion des pyramides de ses points. La complexité de la différence symétrique de deux figures, qui les différencie, sera ainsi le volume de la différence symétrique des deux pyramides. La complexité de l'intersection, qui mesure l'analogie, sera le volume de l'intersection. Le fait de prendre des formes pyramidales fixe un compromis entre différenciation et analogie. *Il importe de remarquer que l'information commune ou propre se traite dans un langage épaissi par rapport au langage superficiel. Par exemple, pour deux*

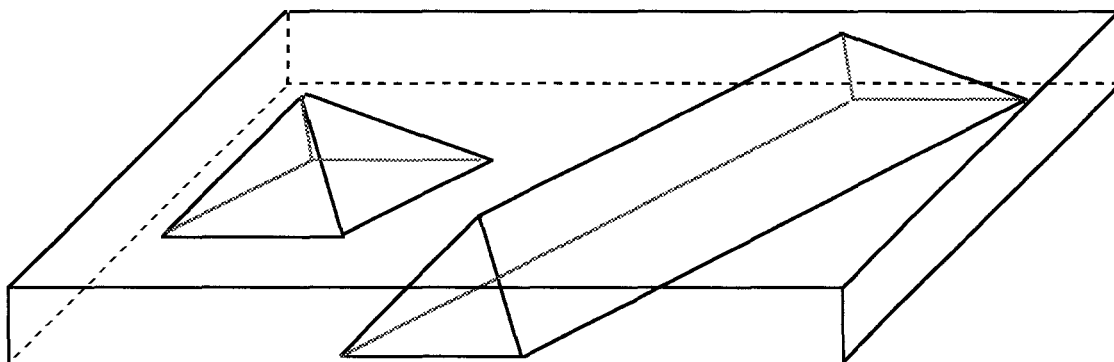
On peut considérer chaque pixel comme un cube unité. Ces cubes pavent l'épaisseur de la rétine. On peut considérer la surface de la rétine comme pavée de triangles ou de carrés, selon que l'on coupe selon des faces ou selon des plans "diagonaux". Les figures qui suivent illustrent l'idée pour une rétine de dimension 1. En prenant un pavage suffisamment fin (et un nombre de couches assez grand) on peut supposer qu'on passe au continu.



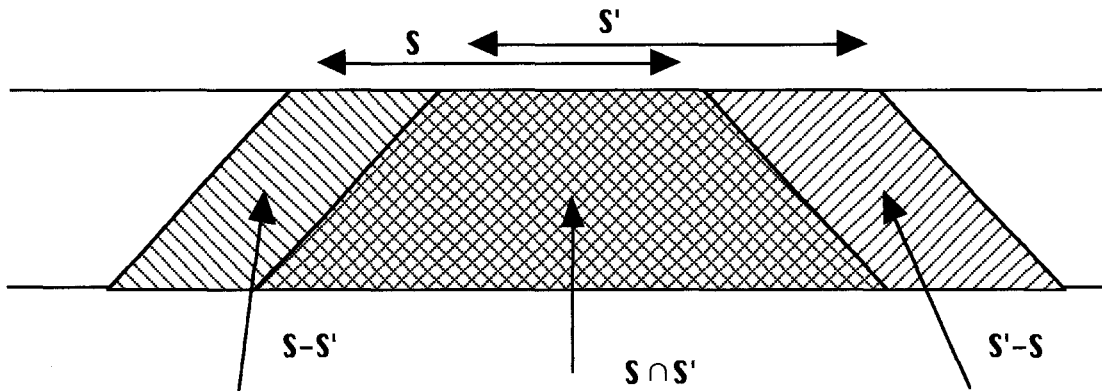
La figure qui suit montre un pixel, vu de la surface d'une rétine de dimension deux, dans le cas d'un pavage triangulaire.



On voit en perspective les volumes pyramides associées à un point et un segment. Dans ce cas, les deux figures, éloignées, n'ont pas d'information en commun.



Dans le cas d'une rétine de dimension 1, nous illustrons en coupe l'intersection et la différence symétrique de deux segments.



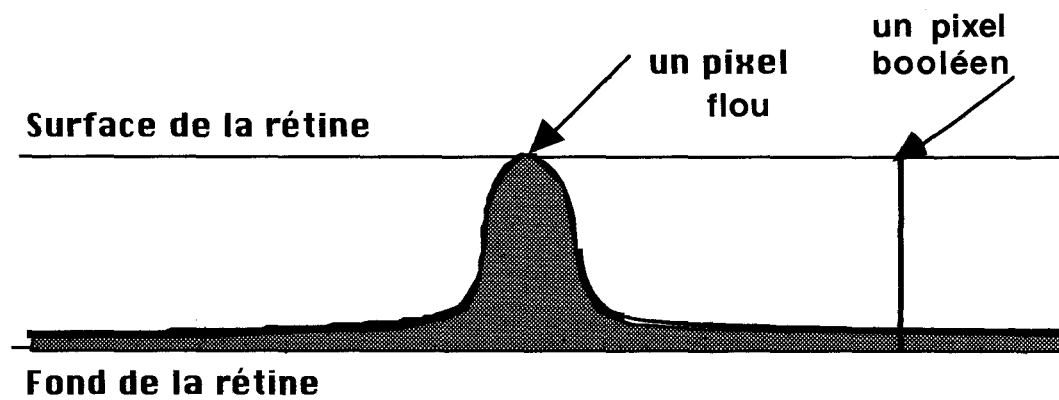
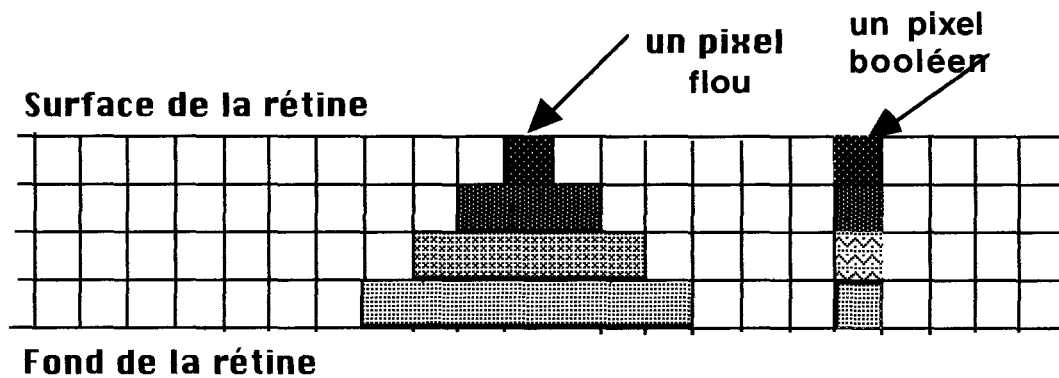
Un pixel donné est donc dénommé par un triplet : sa profondeur p , qui indique la couche, et deux coordonnées x et y qui le repèrent dans la couche. La profondeur est bornée par l'épaisseur de la rétine. La complexité d'un pixel est donc de l'ordre de $\log x + \log y$. Nous considérerons en fait que tous les pixels ont la même complexité, considérée comme unitaire. Ceci revient à se borner la taille de la rétine a priori, ce qui dans la pratique est toujours le cas. On ajuste les constantes si on veut agrandir la rétine.

Nous avons dit que ce qui fonde le caractère booléen du langage, c'est considérer chaque dessin comme un ensemble de pixels. Mais nous avons les restrictions habituelles sur les parties, qui doivent être obtenues par des programmes courts. Tout ce que nous avons dit sur l'exemple des ensembles (voir 3.3) tient ici. On se limite à un nombre polynomial de parties. Ici, on se fixera un choix canonique défini comme suit, basé sur le fait qu'on considère des approximations par segments.

Pour tout segment donné s , l'ensemble des $s \cap s'$ et des $s - s'$, pour tous les s' possibles, dépend polynomialement du nombre de pixels de s , donc de $K(s)$. Ces parties sont indicées par le nombre de s' dont l'intersection avec s est non vide. De même pour tout k donné, l'ensemble des parties de s définies par intersection ou complémentarité avec k segments quelconques est polynomialement lié à $K(s)$ (la puissance du polynôme est de l'ordre de k). La donnée de la fonction de défaut fixe ce k . Comme déjà discuté précédemment, cette restriction se recoupe avec le fait que pratiquement, on ne manipule pas de grands ensembles d'objets sans structure, c'est à dire définissant entre eux des parties en nombre non polynomial.

Remarque d'implémentation Il est évident que les complexités que nous introduisons, tout comme l'épaisseur de la rétine, peuvent être considérées comme fictives. On peut mimer efficacement un langage booléen dans un surlangage non booléen, et ne respectant que fictivement les complexités, sans rien retirer à l'intérêt des représentations introduites.

Représentation pyramidale et logique floue Nous ne développerons pas ce point, mais il nous semble qu'il serait intéressant de l'étudier. L'approche binaire classique correspond à des pyramides qui sont en fait réduites à la colonne des pixels sous la couche superficielle. On fait ainsi rentrer dans le même cadre booléen logique floue et logique classique. Les coefficients de vraisemblance deviennent des mesures de volume. Le point de vue booléen est le cas où la capacité d'analogie est nulle.



Partie 1 - Chapitre 6.

Construction de bases de connaissances

où l'on montre comment le traitement booléen permet de rester dans le raisonnable.

Dans un langage booléen, - et dans la limite de la fonction de défaut - l'itération de la décomposition $(s, s') = (s \cap s', s - s', s' - s)$ fournit une représentation unique de tout ensemble d'objets. Nous nous intéressons ici à la complexité en temps de la construction.

Il se peut que la construction soit effective en théorie (c'est le cas si tous les programmes s'arrêtent) mais qu'elle reste dans la pratique impraticable en temps.

Nous présentons ci-dessous deux aspects de la complexité de construction des bases de connaissances, et esquissons comment les hiérarchiser. L'aspect "glouton", qualitatif, est mineur et donné en annexe. Le point le plus important concerne les aspects polynomiaux. Plus précisément, nous montrons que sous l'hypothèse booléenne, il n'y a pas d'explosion combinatoire. Plus précisément, si on ne tient pas compte du temps d'exécution des programmes extraits des séquences données, mais si on compte seulement le nombre de leurs appels (que l'on pourrait appeler requêtes), le nombre de ces requêtes est polynomialement borné en fonction de la complexité des données. Ainsi, il n'y a pas d'explosion de l'arbre des choix ; au niveau macroscopique (en considérant chaque séquence de la base comme atomique) on reste dans le polynomial. Ceci est très important, car il concerne le niveau stratégique. Si chaque opération élémentaire de la base est elle aussi polynomiale en temps, on reste dans le praticable.

Remarquons que ceci recouvre la pratique, et découle de la borne polynomiale des partitions :

Dans des bases de connaissance booléennes, une donnée ne peut pas se décomposer en un nombre arbitrairement grand de composants, tout comme une séquence perd tout sens si on la réduit à l'ensemble de ses bits. Une figure, un paysage, ne se décompose pas à l'infini mais en un petit nombre d'éléments simples (pour un arbre, on considère "le feuillage" et pas les feuilles une à une). Dans les jeux qui consistent à chercher un objet caché, il y a justement explosion parce que les figures qui ont un sens apparent peuvent se décomposer n'importe comment pour constituer l'objet caché (un bout de carotte, de menton, de raton laveur, de casserole, peuvent dessiner un rasoir).

Aspect polynomial de la construction de la base :

On se donne une fonction de défauts, que l'on majore par une fonction de la forme $A \log() + B$

Deux faits concomitants sont à la base de ce paragraphe

- le nombre polynomial seulement de parties d'un ensemble.
- le rôle privilégié des "programmes court".

• L'oracle "programme court".

a / effectivité : On va supposer que l'on dispose d'un algorithme qui, pour toute donnée s et tout programme P , décide de l'arrêt de $P(s)$ si $|P| < A \log |s| + B$.

(ou plus simplement tous les programmes que l'on considère s'arrêtent)

b / complexité : Quand on se placera sous l'hypothèse de cet oracle, les calculs de complexité en temps ne compteront que les appels de $P(s)$ et non leur temps de calcul (la réponse sera supposée instantanée).

Sous l'hypothèse de cet oracle, on a les résultats suivants (les procédures considérées peuvent sortir du langage restreint donné) :

– On dispose d'une procédure polynomiale $REC(s)$ qui énumère tous les recouvrements partiels d'une séquence donnée.

(un recouvrement partiel $\{s_1, \dots, s_p\}$ de s est un ensemble de parties s_1, \dots, s_p de s qui s'identifie (au sens des programmes courts) à une partie de s).

Preuve : par définition, il n'y a qu'un nombre polynomial de programmes qui engendrent les parties.

– On peut tester polynomialement l'inclusion de s dans s' :

Preuve : On énumère toutes les parties de s' . (le nombre de parties est polynomialement lié à la complexité de l'ensemble)

- La compression optimale est polynomiale : (pourvu qu'on ait l'arrêt)

Preuve : Soit p un programme incompressible de s . Le programme p s'identifie à une partie de s . Il suffit donc d'énumérer les parties de s , et de les exécuter comme programmes. La partie de s (ou l'une des parties de s) de longueur minimale engendrant s est la forme incompressible. Comme l'exécution d'un programme est un programme court, les hypothèses nous assurent de l'effectivité et du caractère polynomial du nombre d'appels de programmes (et non de leur temps de calcul).

- On peut tester polynomialement l'indépendance :

Preuve : Soient s et s' . On énumère leurs parties. On garde les parties communes, on les compresse. On conclut selon la longueur maximale obtenue.

- On peut générer polynomialement les partitions d'une séquence.

Preuve : On génère les recouvrements partiels. Par test de l'inclusion, on ne garde que les recouvrements. Pour chacun, on teste l'indépendance deux à deux des parties. Comme le nombre de parties d'un recouvrement est aussi borné polynomialement, on obtient un algorithme polynomial.

- La décomposition de (s, s') en le triplet $(s \cap s', s - s', s' - s)$ est polynomiale :

Preuve : On énumère les partitions en deux parties de s et s' . On les trie, et on constitue en triplets les paires de partitions ayant une partie en commun. On sélectionne les triplets qui sont des partitions. On compresse les parties communes et on garde un triplet dont la partie commune est de complexité maximale.

- La construction et la mise à jour d'une base sont polynomiales :

Preuve : Soit $\pi(s)$ une borne polynomiale pour le nombre de partitions de s et pour le nombre de parties qui les composent. Soient s_1, \dots, s_n les informations à constituer en une base B_n . Supposons B_{n-1} constituée; elle a au plus $\pi(s_1) + \dots + \pi(s_{n-1})$ éléments (qui sont indépendants deux à deux). Considérons UB_{n-1} , la réunion des éléments de B_{n-1} ; sa complexité est majorée par la somme des complexités des s_j .

On décompose (s_n, UB_{n-1}) en triplet. On énumère les partitions de $s_n \cap UB_{n-1}$ jusqu'à obtenir des partitions qui soient composées d'éléments de B_{n-1} .

On obtient bien un algorithme de construction de B_n à partir de s_1, \dots, s_n qui est polynomial en la somme des tailles des s_j (en la somme des complexités si les données ont été préalablement comprimées).

Le cas de la suppression se traite trivialement.

ANNEXE :

Aspects gloutons

Les hypothèses booléennes facilitent l'approximation de cette base par "accumulation de connaissances".

Plus précisément, la démarche qui consiste à rechercher des parties communes aux objets (sous séquences communes à des séquences) est complète. (Quand nous parlons de sous- séquence, ce n'est pas nécessairement un facteur mais une information extraite par un programme court, conformément à la définition). Or, cette recherche se pratique uniquement par des programmes courts (c'est la définition), et apparaît donc comme réaliste (le nombre de programmes courts est borné polynomialement par les données, puisqu'il est en longueur en $O(\log)$). Quand on arrive à tout décomposer en parties indépendantes, on a gagné. C'est l'indépendance qui est la plus difficile à tester, mais les connaissances ou réputations d'indépendance peuvent *s'accumuler* au cours des temps et par multiplication des expérimentateurs (par utilisation des propriétés gloutonnes).

Plus formellement :

Propriétés gloutonnes :

pour l'indépendance

$$(s \subseteq s' \text{ et } [s, s'']) \Rightarrow K(s' / s'') \geq K(s)$$

pour l'inclusion

$$(s \subseteq s' \text{ et } s \subseteq s'') \Rightarrow K(s' / s'') \leq K(s') - K(s)$$

Hiérarchisation. Couches numériques et booléennes.

Le cas exact.

Prenons des SEGMENT(A, B), TRIANGLE(A, B, C) comme langage.

On se ramène à l'exemple 5.

C'est la complexité des points qui est prépondérante, et à ce niveau, les programmes

(P) A :B :C : SEGMENT(A, B); SEGMENT(B, C)

(P') A :B :C : SEGMENT(A, B); SEGMENT(B, C); SEGMENT(A, C)

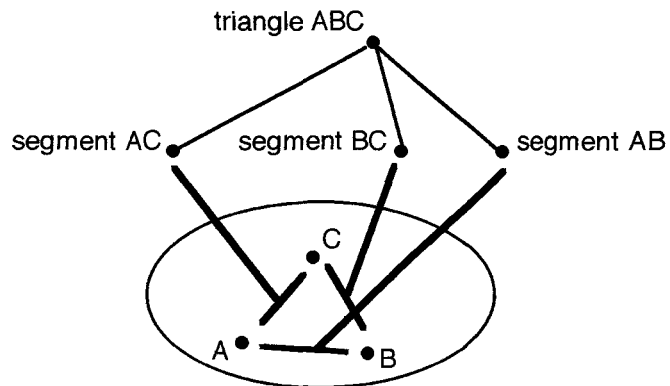
et (P'') A :B :C : TRIANGLE(A, B, C) sont confondus

(on passe de l'un à l'autre par programmes de taille négligeable par rapport à la complexité de A, B, C).

Si on prend maintenant le langage booléen L' faisant fi de la complexité des points, (en les réduisant à des manipulations d'identificateurs) la complexité des programme devient en gros leur longueur, et (P'') sera distingué (et préféré à (P')) si on applique la règle d' OCCAM, car (P'') et (P') définissent la même figure).

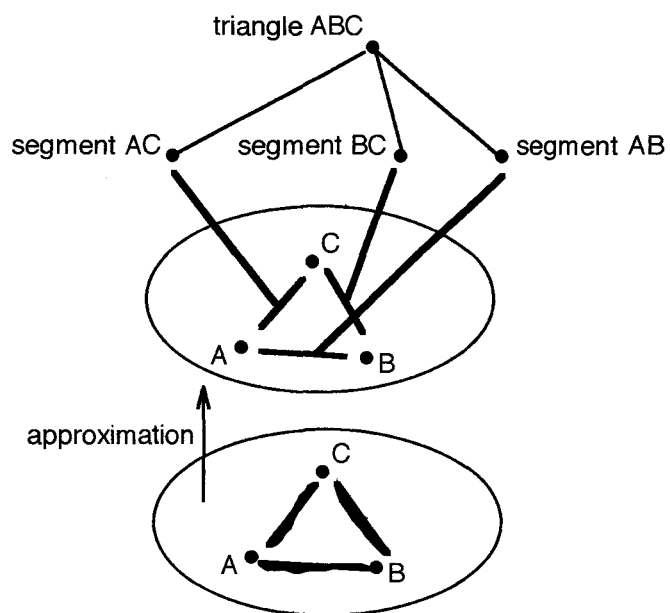
On obtient ainsi deux niveaux de représentation

- le niveau bas des coordonnées des points. On obtient à ce niveau les trois points A, B, C.
- le niveau haut des figures géométriques sur ces points. Il est défini en prenant A, B, C comme des atomes



Le cas approché. (par analogie à l'exemple 6)

Ici, seul le haut niveau est booléen, puisque nous avons vu que nous ne savons pas symboliser les liens de proximités entre points.



Partie 1 - Chapitre 7.

Conclusion

où l'on donne quelques restrictions et perspectives à notre travail

Si l'on veut donner une définition précise à la dichotomie numérique - symbolique dans le cadre de ce travail, c'est en un sens bien particulier. Pour nous, on peut considérer que le symbolique se restreint au booléen et que le non symbolique est le non booléen. Ceci est évidemment trop restrictif mais n'a que la prétention d'être un cas particulier. Les langages booléens fournissent ici un cadre au calcul de prédicats. Un cadre à des calculs sur des relations entre les objets reste à étudier du point de vue de notre approche.

D'autre part, on remarque que la phase numérique de l'exemple des silhouettes de véhicules, aboutit à représenter une silhouette par l'ensemble de ses propriétés qui nous intéressent. Ainsi, $R(V1) = \{FGR, FDR\}$. On pourrait considérer qu'il s'agit là des résultats de trois programmes ("numériques") de prétraitement. Ainsi, un prétraitement peut approximer un dessin par des segments, ou, de manière plus drastique, par un attribut (nombre de roues, forme d'une face, couleur...).

Soulignons ici une autre restriction de notre travail : nous avons restreint les relations sémantiques entre objets à une notion de distance, et nous avons imposé au prétraitement de donner des résultats proches (au sens du calcul dans un langage booléen) pour des objets proches sémantiquement. Cette dernière contrainte est légitime, mais la restriction initiale des relations sémantiques à une distance mériterait d'être enrichie. Par exemple, si les pixels étaient en couleur, cet attribut pourrait être pris en compte par une seconde distance, quitte à ce que cette distance se ramène à une égalité (distance infinie entre couleurs différentes). Mais les relations entre les objets en général restent à étudier.

Conclusion.

Nous pensons que l'approche en termes de complexité devrait aussi être élargie afin d'aborder sous cet angle les liens entre les objets et l'observation de l'objet, les liens entre complexité et types. Esquissons ici les problèmes sur des exemples.

Nous avons toujours identifié objet et codage binaire de l'objet. Pour nous, ceci est moins clair qu'il n'y paraît. Comment représenter un vase? On peut se contenter d'en décrire les apparences, mais on ne pourra alors pas traiter par programme les notions de poids, de porosité, d'opacité aux rayons X, etc.... Si l'on veut ne perdre aucune information sur le vase, il va falloir le décrire au niveau moléculaire. A la limite, on tombe sur un choix de modèle physique de la réalité empirique. On peut se demander si "coder toute l'information d'un objet a vraiment un sens". Même si on limite ses ambitions à un codage qui est d'emblée une représentation partielle, le choix de ce codage influera fortement sur les traitements. Par exemple, si on se donne d'emblée la couleur d'un vase en termes de pixels, la couleur du vase sera facile à déterminer. Il en sera tout autrement si on se donne seulement les structures moléculaires des matières premières du vase et de sa peinture. La complexité (de KOLMOGOROV) du programme "couleur de" peut ne pas être négligeable. Même si celle-ci est vraisemblablement plus courte qu'il n'y paraît, le problème de fond demeure, et on retrouve en terme de complexité que le résultat d'une mesure n'est pas une propriété intrinsèque de l'objet mais du couple (objet, processus d'observation). Pour nous, cette dernière assertion se traduit ainsi : étant donné un objet de programme minimal o et un processus d'observation de programme minimal p , il se peut que $K(p(o)/o)$ ne puisse être assimilée à 0. Dans le présent travail, nous ne nous sommes pas posé ce genre de question.

Soulevons par l'exemple un autre problème : considérons les informations $V3$ et $(V3, FGR)$. On peut supposer que $V3$ est un long trait aléatoirement perturbé et donc incompressible. La séquence codant le symbole FGR a alors une longueur négligeable par rapport à celle codant le dessin $V3$. Dans l'absolu, on peut donc identifier les informations $V3$ et $(V3, FGR)$. La contradiction qui s'appuierait sur la remarque que $V1$ n'a pas la propriété FGR n'est qu'apparente. $(V3, FGR)$ n'est rien d'autre que la juxtaposition du code d'une silhouette et de celui d'un symbole. On voit que c'est le contexte qui fixe le sens d'une séquence. FGR est dans le langage L , comme résultat d'un prétraitement, qui n'est d'ailleurs pas défini sur FGR . On retrouve une problématique de typage en lien avec les calculs qui rejoint la problématique générale des types et qui mériterait d'être déployée (même si ce n'est que pour constater qu'on ne fait que mimer l'approche générale des types en lien avec les preuves ou la fonctionnalité). De même, la notion de point de vue et celle d'objet pourrait être abordée en termes de complexité. Par exemple, un point de vue serait l'information apportée à une classe d'objets par des processus. On pourrait ranger sous cette problématique les rapports entre une théorie et la réalité des objets et liens entre les objets qu'elle modélise.

Les liens booléen - flou, esquissés seulement ici, pourraient être développés également dans le cadre unificateur de la complexité.

Partie 1

Annexes

Annexe I. Dessin au trait.

Le dessin au trait

Nous considérons ici des pixels carrés. Les mêmes considérations valent pour des pixels triangulaires.

Sur une rétine R , nous appelons :

1- dilaté d'une figure F , la figure notée $\Delta_1(F)$ définie par : $\Delta_1(F)$ est la réunion des voisinages de VON NEUMANN de tous les points de F .

Le voisinage de VON NEUMANN d'un point d'une rétine est l'ensemble de ses huit plus proches voisins et de lui-même. C'est le dilaté de F au sens de la morphologie mathématique avec voisinage de VON NEUMANN comme élément structurant.

k dilaté d'une figure F , la figure notée $\Delta_k(F)$ définie par : $\Delta_k(F) = \Delta_1(\Delta_{k-1}(F))$

distance entre deux figures F et G : $d(F, G) = \inf(k / G \subset \Delta_k(F) \text{ et } \Delta_k(G) \subset F)$

C'est la distance classique de HAUSDORFF.

la figure F k approxime la figure G si $d(F, G) \leq k$.

k érodé d'une figure F , la figure notée $E_k(F)$ définie par : $E_k = \overline{\Delta_k(\bar{F})}$ où \bar{F} désigne le complémentaire de F .

On a évidemment

$$\text{saturation } F \subset E_1(\Delta_1(F))$$

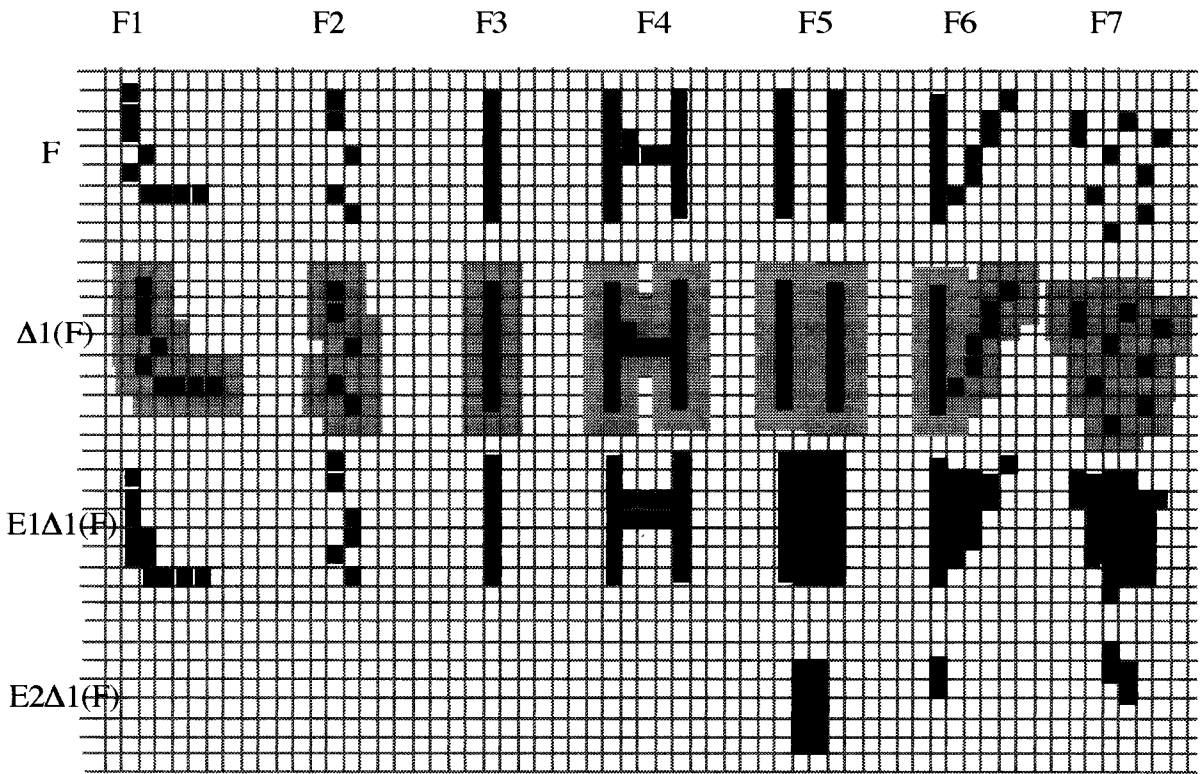
$$\text{idempotence } E_1(\Delta_1(F)) = E_1(\Delta_1(E_1(\Delta_1(F))))$$

Une figure F sera du **dessin au trait à l'échelle r** si et seulement si $E_{2r+1}(\Delta_r(F)) = \emptyset$.

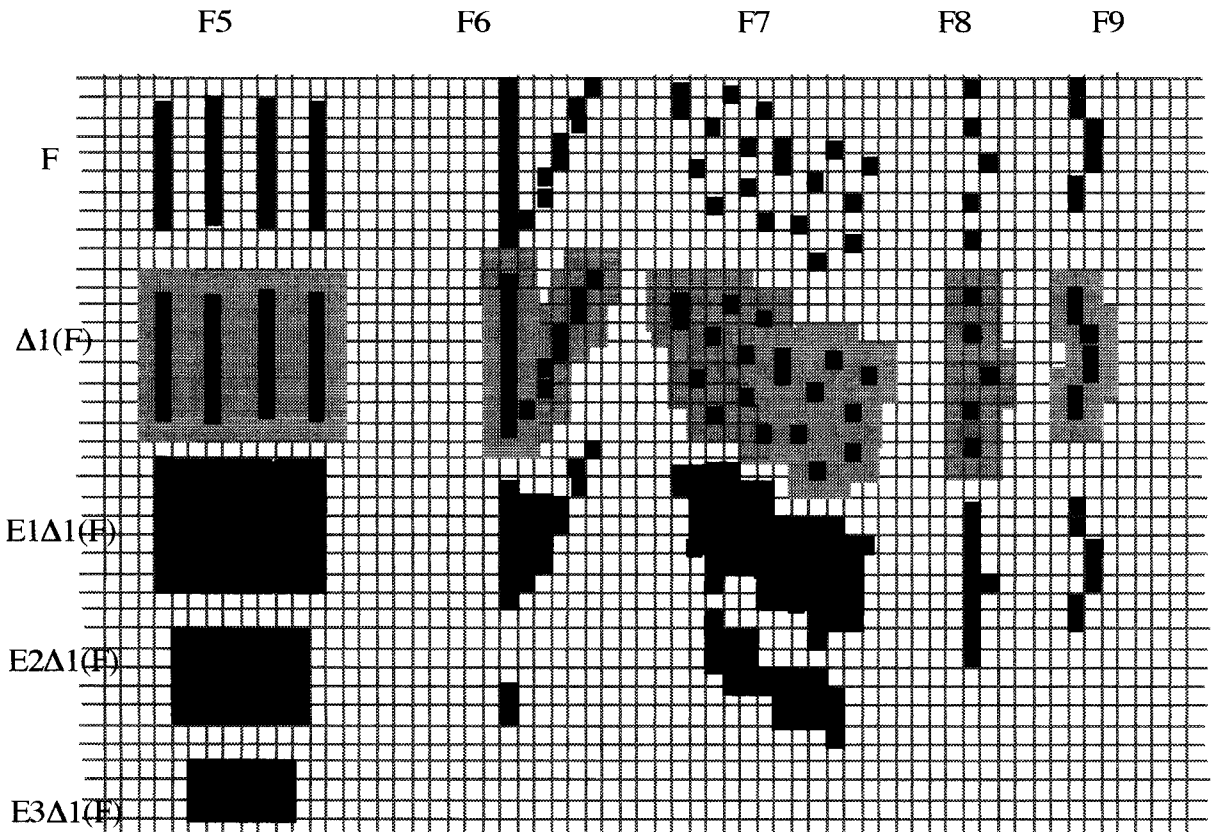
F sera dite **saturée** si $E_1(\Delta_1(F)) = F$.

Notons qu'avec notre définition, un ensemble de points suffisamment espacés est considéré comme du dessin au trait. La définition permet de considérer comme un seul trait, deux traits suffisamment voisins. On considérera en général des dessins saturés.

Exemples :



F1, F2, F3 et F4 sont des dessins au trait. F5, F6 et F7 sont repris comme sous ensembles des F5, F6 et F7 de la figure suivante. F5 n'est pas du dessin au trait (c'est de la texture "lignes verticales". Par contre, l'angle aigu F6, la large ligne pointillée F7, les segments perturbés F8 et F9 en sont.



Dans la figure qui suit (à échelle r donnée), G1 est considéré comme du dessin au trait, approximation d'un même segment. G2 et G4 ne sont pas du dessin au trait mais de la texture, G3 est du dessin au trait.

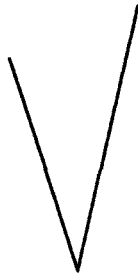
G1



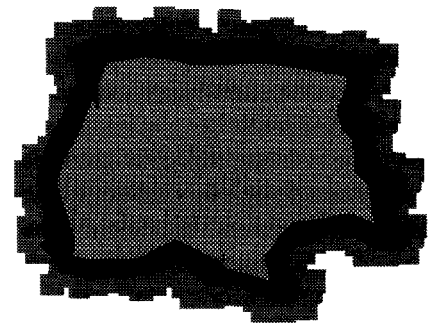
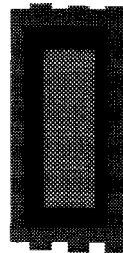
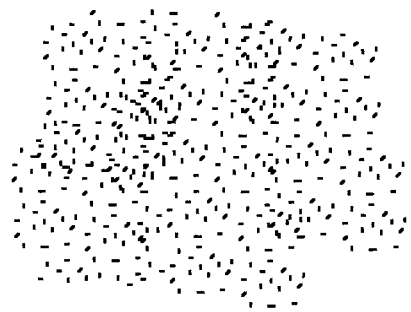
G2



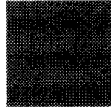
G3



G4



Δr



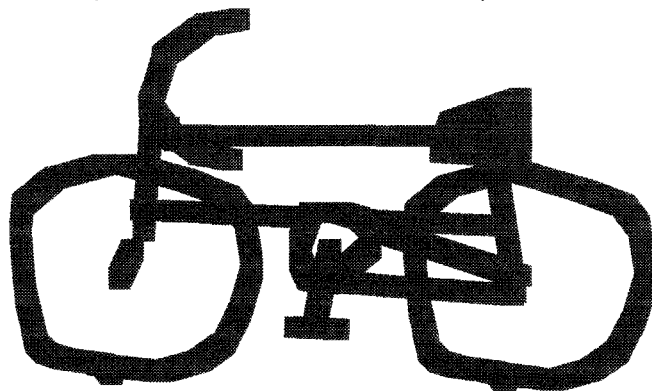
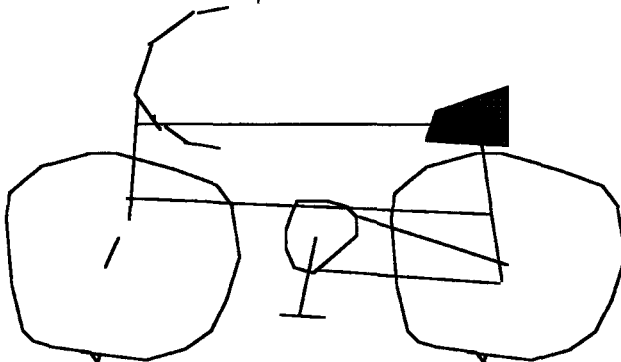
$E_r \Delta r$



$E_{2r+1} \Delta r$



La figure avec les vélos illustre l'importance du facteur d'échelle.



Annexe II. L'approximation d'un dessin au trait par un nombre minimum de segments est NP-dure.

Problème d'optimisation :

Nom : k-ACMiS (k approximer une courbe L sans boucle par un minimum de segments)

Entrée : une courbe L sans boucle et un entier k.

Recherche : une k-approximation de L avec un nombre minimum de segments.

Un premier élément de réponse est que cette recherche est un problème en général NP dur, en un sens que nous allons préciser.

Tout d'abord l'approximation est une k approximation telle que définie précédemment.

Ensuite le problème se pose sur une rétine de pixels. Il est donc discret. Nous donnons pour la clarté de l'exposé une présentation continue de notre construction.

Ce qui suit n'est donc qu'une indication de preuve, la preuve formelle dans le cas discret étant fastidieuse.

De même, on se contentera ici de la "métaphore ferroviaire" afin d'alléger la présentation. C'est en ce sens qu'il faut entendre la notion d'embranchement.

Théorème :

Le problème :

Nom : k-ACMiS (k-approximer, une courbe L sans boucle par un minimum de segments)

Entrées : un entier k et une courbe L sans boucle

Recherche : une k-approximation de L avec un minimum de segments.

- est un problème NP Dur (même si la courbe est sans boucle)
- est un problème polynomial, si la courbe ne comporte ni embranchement, ni croisement

Idée de preuve :

- NP Dur en général

Pour montrer que le problème k-ACMiS est NP Dur, il suffit d'établir que le problème suivant est NP complet :

Problème de décision :

Nom : k-APP(L, p) (existe-t-il une k approximation de L avec p segments ?)

Entrées : un entier k, une courbe L sans boucle, un entier p.

Question : existe-t-il une k approximation de L avec p segments ?

Le fait que le problème k -APP(L, p) soit NP est évident : il suffit de vérifier qu'une ligne brisée de p segments, k approxime L .

Pour montrer que k -APP(L, p) est NP complet, nous allons y réduire le problème de la satisfiabilité des expressions booléennes sous forme 3-CNF, c'est à dire sous forme normale conjonctive de facteurs de trois termes.

Reprenons l'image ferroviaire pour donner l'idée de la réduction.

Pour toute expression E , 3-CNF, nous allons construire un réseau R_E selon le principe suivant :

Si $E = \bigwedge F_i$, notons F_1, \dots, F_n les facteurs. A chaque facteur correspond une gare terminus.

A chaque variable A booléenne présente dans les facteurs, correspond un circuit \mathcal{A} sans boucle, qui se ramifie par des aiguillages en des terminus du type A dans un sens, et du type $\neg A$ dans l'autre sens.

Chaque gare contient un seul quai, où convergent par aiguillage les circuits correspondant aux variables. Le fait de considérer des expressions 3-CNF limite à trois le nombre de voies convergentes par gare, ce qui permet sans problème de simuler dans un cadre discret ce que nous présentons ici en continu.

L'idée de la réduction est que :

R_E peut-être k approximé avec p segments si et seulement si E est satisfiable.

Nous allons construire R_E de façon à ce que :

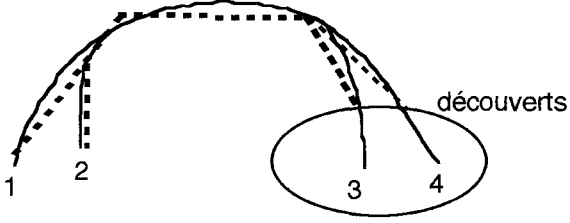
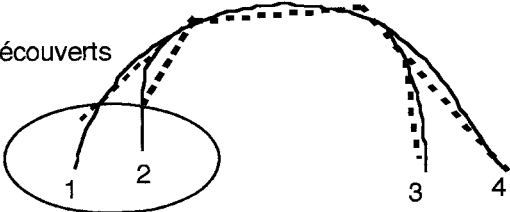
- pour chaque circuit \mathcal{A} il existe un entier P_A qui nous place devant le dilemme suivant : avec P_A segments, je peux k approximer le circuit \mathcal{A} à ceci près que soit les quais A , soit les quais $\neg A$ ne seront pas couverts par l'approximation. Couvrir le quai A revient à affecter la valeur vrai à la variable booléenne A , couvrir le quai $\neg A$ revient à affecter la valeur faux à la variable booléenne A .

Pour couvrir R_E , il faut et il suffit que chaque quai F_i soit couvert par l'approximation d'un des trois circuits qui y aboutit.

On voit alors qu'approximer R_E avec $p = \sum_{A \in \text{VAR}(E)} P_A$ segments équivaut à satisfaire E .

NB : Il resterait à montrer que l'on peut toujours dessiner au trait R_E . La chose est évidente dans le plan réel, mais pas dans $\mathbb{N} \times \mathbb{N}$. Une étude fastidieuse que nous ne faisons pas ici, montre que c'est toujours possible.

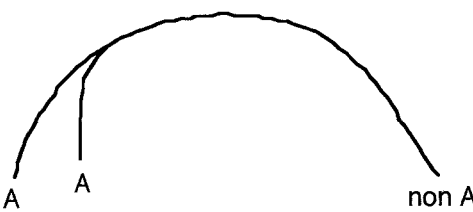
le dilemme :

	<p>Le circuit ci-contre est construit pour illustrer le dilemme qui se pose : avec un minimum de cinq segments si nous "couvrons" les extrémités 1 et 2, alors nous découvrons les extrémités 3 et 4.</p>
	<p>si nous "couvrons" les extrémités 3 et 4, alors nous découvrons les extrémités 1 et 2.</p>

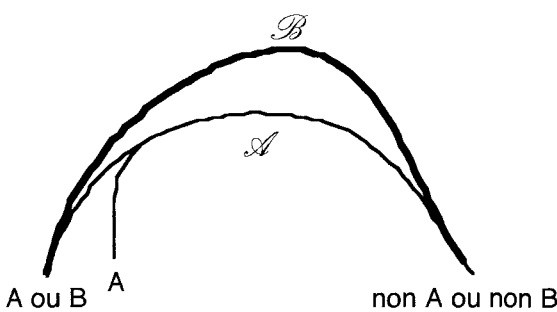
Pour recouvrir de n segments une telle figure, nous sommes obligés de découvrir tous les quais d'une extrémité.

exemple pour un circuit nommé \mathcal{A}

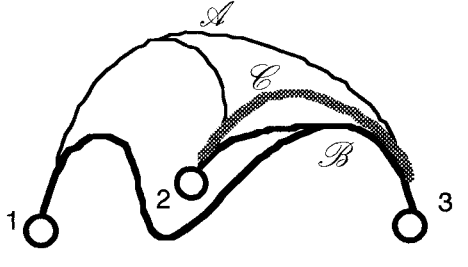
Nous noterons **A** une extrémité et **non A** l'autre extrémité, et nous interprétons A comme une variable booléenne qui prend la valeur vrai si l'extrémité est recouverte, et la valeur faux si l'extrémité n'est pas recouverte.

	<p>Ce circuit correspond à la proposition logique toujours fausse :</p> <p style="text-align: center;">A et A et non A</p> <p>si A est vraie alors nous couvrons les extrémités "A", et découvrons les extrémités "non A" (non A fausse) , la proposition A et non A est donc fausse.</p> <p>inversement si A est fausse alors nous découvrons les extrémités "A" et couvrons les extrémités "non A" (non A vraie) , la proposition A et non A est donc fausse.</p> <p>En effet nous avons vu que nous ne pouvons pas recouvrir à la fois les deux extrémités</p>
---	--

exemple pour deux circuits nommés *A* et *B*


 <p>Les circuits ci-dessus aboutissant à trois gares, s'interprètent par la formule :</p> <p>(A ou B) et (A) et (non A ou non B)</p> <p>si cette formule est vraie alors c'est que chacune des extrémités est couverte, c'est à dire que l'on peut recouvrir les circuits par un nombre de segments égal à la somme des minima de chaque circuit</p>	<p>•1• Si l'un des circuits couvre une extrémité, alors l'autre circuit pourra indifféremment couvrir ou ne pas couvrir cette même extrémité.</p> <p>•2• Couvrir tous les circuits par un minimum de segments revient à couvrir chaque circuit par un minimum de segments. Le minimum pour les circuits étant alors bien évidemment la somme des minima pour chaque circuit.</p> <p>Ainsi nous allons interpréter une extrémité où deux circuits viennent se terminer selon deux courbes tangentes par une formule connectée par " ou "</p> <p>Cette formule sera vraie si au moins l'un des littéraux est vrai c'est à dire si au moins l'un des circuits couvre l'extrémité en question.</p>
--	--

exemple pour trois circuits nommés *A* et *B* et *C*.

	<p>La gare 1 est codée non A ou B La gare 2 est codée A ou B ou non C La gare 3 est codée A ou non B ou C</p> <p>L'expression $E = (A \text{ ou } B) \text{ et } (A \text{ ou } B \text{ ou non } C) \text{ et } (A \text{ ou non } B \text{ ou } C)$ est satisfaite pour A vrai ; B faux et C vrai</p>
---	--

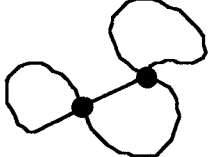
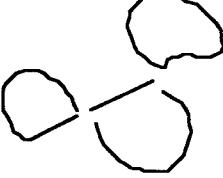
cas où la courbe n'a pas d'embranchement : le problème est polynomial

Nous allons encore indiquer une idée de preuve :

	<p>Chaque point surligné de la figure ci-contre est parcouru un nombre de fois inférieur ou égal à la longueur de la courbe, donc le problème est au pire quadratique.</p>
---	--

Autre acceptation du problème : On s'impose de pointer les points singuliers.

On ne considère que les approximations dont les segments ont leur extrémités aux points singuliers du dessin au trait.

<p>•1• Repérer les points singuliers c'est un algorithme en temps linéaire de suivi de contour</p>	
<p>Nous obtenons une liste de lignes sans boucle et sans embranchement fermées ou non</p>	
<p>•2• Pour chacune de ces lignes "simples", la segmenter avec un minimum de segments.</p>	

On déduit du as précédent que le problème de déterminer sous cette contrainte, une approximation ayant un minimum de segments est un problème polynomial.

Heuristiques (voir aussi partie 2 chapitre 3, section 3.4). Il est très facile de concevoir des algorithmes rapides qui fournissent une solution avec au plus le double de segments du minimum. Par exemple, on peut partir de n'importe quelle solution, et remplacer tant que possible deux segments consécutifs par un seul. On vérifie aisément qu'on arrive au résultat.

Partie 1

Bibliographie

- [BENN 89] BENNET, Ch. H.,
Time / Space Trade-offs for reversible computation,
SIAM J. Comput. Vol 18-4, pp. 766-776, 1989
- [BENN 90] BENNET, Ch. H. ,
How to Define Complexity in Physics, and Why, Complexity, Entropy, and the Physics of information,
SFI Studies in the Sciences of Complexity, vol. VIII, Ed. W.H. ZUREK, Addison-Wesley, pp. 137-148, 1990
- [BENN 93] BENNET, Ch. H., P. GÀCS, M. LI, P.M.B. VITÀNÝI, W. H. ZUREK,
Thermodynamics of Computation and Information Distance,
à paraître, 1993
- [BOUC 92] BOUCHERON, S.,
Théorie de l'apprentissage,
Hermes Ed. 1992
- [CHAT 87] CHAITIN, G.J. ,
Information, Randomness & Incompleteness,
World Scientific ed. , 1987
- [DELA 91] DELAHAYE, J.P.,
Thermodynamique et informatique théorique,
Pour la Science, avril 1991
- [DELA 91] DELAHAYE J.P..
" Articles publiés dans Pour la Science en 1991 "
Publication interne IT n°222 LIFL USTLFA 1991
- [DELA 92] DELAHAYE, J.P.,
La profondeur logique selon BENNET,
Pour la Science, août 1992

[DELA 93] DELAHAYE, J.P.,

Information, complexité et hasard

Hermes, 1993

[DUPR 91] DUPRAT, J, MULLER, J.M.

Ecrire les nombres autrement pour calculer plus vite

TSI 10-3 pp 211-223, 1991

[GACS 73] GÀCS P., J. KORNER,

Common information is far less than mutual information,

Problems of Control and Inf. Th. 2 : 149-162, 1973.

[LEEU 90] Van LEEUWEN J., MEYER A.R., NIVAT M., PATTERSON M.S., PERRIN D.

" Handbook of Theoretical Computer Science : Algorithms and complexity"

MIT press Cambridge Massachusetts 1990.

[LI 89] LI. M & P.M.B. VITÀNÝI,

Inductive Reasoning and KOLMOGOROV Complexity,

Structure in Complexity Theory, 4th annual Conference, Oregon, IEEE Computer Society Press, pp. 165-185, 1989

[LI 90] LI, M. & P.M.B. VITÀNÝI, KOLMOGOROV

Complexity and its Applications,

Handbook of Theoretical Computer Science, Vol A, Elsevier Ed. pp 187-254, 1990

[LI 93] LI, M. & P.M.B. VITÀNÝI,

An Introduction to KOLMOGOROV Complexity and Its Applications,

Springer - Verlag, 1993

[VALI 84] VALIANT, L.G.,

A Theory of the Learnable,

Communications of ACM, Vol 27-11, pp 1134-1142, 1984

[ZURE 89] ZUREK W.H.

" Thermodynamic cost of computation, algorithmic complexity and the information metric"

Nature Vol 341, N° 6238 pp 119-124 14 sept 89

[ZURE 89] ZUREK, W.H.

Algorithmic randomness and physical entropy

Physica Review A, Vol 40-8, pp. 4731-4751, 1989

Seconde partie

Représentation de connaissances et apprentissage

on étudie le cas du dessin au trait.

0. Introduction

où l'on aborde le propos.

1. La machine PASTIS

où l'on s'adapte aux contraintes technologiques.

2. Mécanismes humains de la vision

où l'on survole les modèles neurobiologiques de la vision.

3. Vision et dessins

où l'on présente et discute des mécanismes artificiels.

4. Apprentissage

où l'on classe les modèles.

5. Modélisation des systèmes complexes

où l'on introduit l'approche systémique.

Guide

où l'on résume les textes à aborder comme on regarde un paysage.

Sommaire

Partie 2 - Chapitre 0.

Introduction.

où l'on présente le propos.

Partie 2 - Introduction

Cette seconde partie, est à découvrir, en la survolant comme un paysage, elle met l'accent sur les représentations et l'apprentissage. Pour faciliter ce survol, un guide placé en fin de chapitre permet de cerner rapidement les différents aspects. Cette partie met en place le cadre qui nous a amené à l'approche Kolmogorov développée dans la première partie.

Tout d'abord, nous présentons le projet "PASTIS" dans lequel cette thèse s'enracine. La machine "PASTIS" est une machine de vision de dessins au trait qui se veut efficace, en adoptant une conception modulaire (en ce sens elle reprend les propositions des cognitivistes sur une organisation modulaire des fonctions cognitives) en processeurs spécialisés.

L'hypothèse que nous formulons dans ce projet, est que l'efficacité résultera d'un compromis à chacun des niveaux, entre l'architecture et la complexité, le temps et l'espace, la redondance et l' à peu près. Le comportement de PASTIS s'inscrit dans une durée, qui lui permet un apprentissage. Un oracle indique à PASTIS les noms des concepts que PASTIS a, tout seul, identifiés. Ces exigences conduisent à des recherches à différents niveaux et dans des cadres multi disciplinaires, entre autres :

- la conception de modules architecturaux dédiés à des tâches précises.
(Jérôme DELEU a développé dans ce cadre un processeur de rotation rapide)
- la recherche de méthodes originales de prise en compte de la réalité.

Pour fixer l'environnement, nous survolons alors les modèles neurobiologiques de la vision. Le cerveau, organe de la pensée est souvent associé au tout dernier progrès technique. Ce modèle "mécanique" est toujours faux. Les travaux les plus récents des neurobiologistes permettent d'entrevoir quelques bribes, quelques pistes. Nous essayons dans le chapitre 2 d'exposer, sans doute de manière un peu trop sommaire, quelques unes des dernières découvertes. L'énumération, en conclusion, des notions admises actuellement, est à lire en regard des options décidées pour la réalisation de notre mécanisme de vision. Rappelons que notre but n'est pas d'étudier la cognition (nous ne sommes pas spécialistes), mais d'en utiliser certaines connaissances pour éclairer notre démarche. Il faut savoir que bon nombre de points de vue évoqués de façon lapidaire sont en fait discutables et discutés par les spécialistes.

Nous présentons ensuite le domaine de la vision des mécanismes artificiels.

Nous proposons d'abord un bref état des recherches en systèmes de vision, ce qui nous permet d'identifier d'une part un objet d'étude : l'image, et d'autre part deux tâches essentielles : la segmentation et l'interprétation. Ces tâches mettent en œuvre des outils et des connaissances organisés en systèmes. Nous nous intéressons à l'image pour identifier des modes de représentation, dont nous montrons qu'ils caractérisent des classes de méthodes. Nous étudions ensuite chacune de ces méthodes.

Nous présentons la notion d'apprentissage, en classifiant les modèles.

Nous établissons tout d'abord un bref panorama historique de la notion d'apprentissage. Nous en déduisons cinq définitions de l'apprentissage qui s'apparentent à autant de points de vue. Nous les passons alors successivement en revue.

Après avoir exploré les mécanismes naturels et artificiels de la vision, puis les stratégies d'apprentissage, nous nous intéressons à la notion de modélisation des phénomènes. Un phénomène compliqué sera rendu compréhensible par sa simplification, sa décomposition en éléments simples. Un phénomène complexe sera rendu intelligible par une composition d'actions pour un projet concevable de connaissances.

Enfin nous résumons tout ce domaine, en un guide.

Partie 2 - Chapitre 1.

La machine PASTIS

où l'on s'adapte aux contraintes technologiques

Partie 2 - Chapitre 1:

Le but de ce chapitre est de présenter le projet dans lequel s'inscrit cette thèse : "la machine PASTIS".

Cette machine est une machine de vision de dessins au trait qui se veut efficace, en adoptant une conception modulaire (en ce sens elle reprend les propositions des cognitivistes sur une organisation modulaire des fonctions cognitives) en processeurs spécialisés.

L'hypothèse que nous formulons dans ce projet, est que l'efficacité résultera d'un compromis à chacun des niveaux, entre l'architecture et la complexité, le temps et l'espace, la redondance et l' à peu près.

Le comportement de PASTIS s'inscrit dans une durée, qui lui permet un apprentissage. Un oracle indique à PASTIS les noms des concepts que PASTIS a, tout seul, identifiés.

Ces exigences conduisent à des recherches à différents niveaux et dans des cadres multi disciplinaires, entre autres :

- la conception de modules architecturaux dédiés à des tâches précises.
(Jérôme DELEU a développé dans ce cadre un processeur de rotation rapide)
- la recherche de méthodes originales de prise en compte de la réalité.

01. La notion de machine PASTIS

PASTIS **P**rocesseurs **A**ssociatifs **S**ymboliques de **T**raitement d'**I**mage**S**

Le projet PASTIS se propose d'étudier une machine qui explore sur un jeu de données graphiques, un itinéraire de compréhension de dessin, en adaptant et en optimisant pour chaque niveau : l'architecture à la complexité, le temps à l'espace, la redondance à l' à peu près.

PASTIS conceptualise sa connaissance, reconnaît, classe, reproduit, ... des dessins au trait.

L'architecture de PASTIS s'élabore à partir de processeurs spécialisés, modules architecturaux.

Le parti pris de conception des modules architecturaux est la simplicité. Un manque de précision sera compensé par de la redondance, un gain de temps par une occupation mémoire.

02. Les étapes historiques de l'évolution du comportement PASTIS.

A. PASTIS nouveau né

La machine ne connaît rien (en fait elle disposera de quelques connaissances "innées"). Elle va partir à la découverte d'un monde de dessins qui lui seront présentés. Un précepteur lui fait donc des dessins, beaucoup de dessins. PASTIS va détecter des régularités, et afficher des fragments en posant des questions : Comment appeler ce morceau de dessin ?

B. PASTIS enfant

PASTIS pose de plus en plus de questions pour infirmer ou confirmer ses connaissances. Le monde de dessins se structure en une partie "comprise" et une partie "inconnue".

C. PASTIS adulte

PASTIS continue de se perfectionner en élaborant éventuellement de nouvelles "méthodes" d'investigation des images.

03. Les états du comportement quotidien de PASTIS.

A. État de veille.

En état de veille, PASTIS :

+ regarde un dessin qui vient de lui être présenté.

La vision de PASTIS est régulée par un "point de vue". Ce point de vue courant est un ensemble de codes qui définissent le "regard" porté par PASTIS sur le dessin, en termes de primitives d'analyse de dessin. PASTIS peut faire évoluer son point de vue ou le diversifier.

+ comprend des "morceaux" de dessin par une impression de déjà vu.

Pastis demande alors selon son niveau de compétence acquise une confirmation ou non.

La question est alors du type : "est-ce bien un ..." Réponse o/n

La communication de Pastis avec son environnement s'établit par l'intermédiaire d'un écran : échange de dessins et dialogues : phrases simples.

La découverte d'un micro monde se réalise à partir de quelques aptitudes innées de Pastis :

- associative : manipulation d'images,
- symbolique : structuration des connaissances,
- verbale : expression et compréhension de phrases grammaticalement simples et préétablies.

Une machine adulte ne posera plus de questions de ce type pour des concepts bien assimilés.

+ répond à des questions du type :

- que vois-tu dans ce dessin ?
- y a t il un cercle dans le dessin ?

+ réagit à des injonctions du type : "Change de point de vue !"

B. État de rêve

En état de rêve, PASTIS :

+ reprend les images, imagine des regards à porter, affine sa compréhension, restructure son organisation de concepts, prépare des questions de confirmation de ses hypothèses, pour sortir de ses dilemmes ...

C. État de réveil

En état de réveil, PASTIS :

+ pose toutes les questions qu'elle a pu élaborer durant son sommeil. PASTIS manifeste éventuellement son désaccord, ou son étonnement.

D. État d'hibernation

En état d'hibernation, PASTIS est tout simplement arrêté.

Une organisation logicielle modélisant ce comportement.

Lorsque PASTIS n'est pas en état d'hibernation, une boucle attend un événement qui peut être soit une présentation d'un dessin, soit des questions sur le dessin.

Au bout d'un certain temps sans événement PASTIS s'endort tout seul.

Un nouvel événement le placera en état de réveil. C'est alors qu'il posera éventuellement des questions, puis réagira à l'événement ayant causé son réveil.

On a donc l'algorithme suivant :

```
état ← veille
répéter
    saisir un événement
    traiter l'événement
jusqu'à ce que état = hibernation.
```

04. Les choix de base pour PASTIS.

4.1 Les choix des objets

Les primitives simples extraites de l'image sont des segments de différentes inclinaisons et de taille fixée.

Comme SERRA le propose nous allons intégrer les principales techniques (analyse structurelle, morphologie mathématique, analyse statistique,...), pour diversifier des points de vue sur une image.

La redondance ainsi obtenue palliera à l'imprécision ou à la spécialisation du point de vue.

Les études réalisées dans cette thèse nous ont permis d'identifier différents axes pour cette redondance :

Les traitements sur les primitives extraites se placeront dans différents cadres théoriques pour former un système hétérogène de reconnaissance de dessins:

- dessin perçu comme une structure informatique sur laquelle agissent des procédures
- dessin perçu comme un ensemble de pixels sur lesquels agissent des transformations morphologiques
- dessin perçu comme une liste de caractéristiques numériques du dessin sur lesquelles agissent des méthodes statistiques
- dessin perçu comme une équation sur les coordonnées de ses pixels sur laquelle agissent des transformations mathématiques
- dessin perçu comme une configuration d'états d'unités d'un système connexionniste qui adapte son comportement en fonction de contraintes qui lui sont imposées.

Les traitements symboliques se placeront dans différents cadres théoriques de l'apprentissage :

- modification stable du comportement imputable à l'expérience.
- modification stable de l'organisation imputable à la capacité à percevoir globalement les objets.
- modification stable de structures mentales internes imputable à une assimilation – accommodation.
- modification stable des poids des connexions entre les unités d'un système imputable à une adaptation, à une réponse imposée.
- modification stable d'un modèle mathématique sophistiqué imputable à un calcul.

Ainsi l'objectif général du projet "PASTIS" est :

Concevoir et fondre des modules architecturaux spécialisés pour l'apprentissage et la reconnaissance de "dessins au trait".

4.2 Les choix architecturaux

L'originalité principale de PASTIS est son architecture. Les ordinateurs classiques utilisent des unités arithmétiques et logiques, PASTIS, lui utilisera des modules spécialisés : RAFSIP (Rough And Fast Specialized Image Processors) ou PIRA (Processeurs d'Images, Rapides et Approximatifs).

Ces processeurs seront rapides parce qu'approximatifs, et comme ils sont très rapides, PASTIS pourra les exploiter très souvent pour analyser et classifier ses connaissances.

Les principaux RAFSIP ou PIRA identifiés actuellement sont :

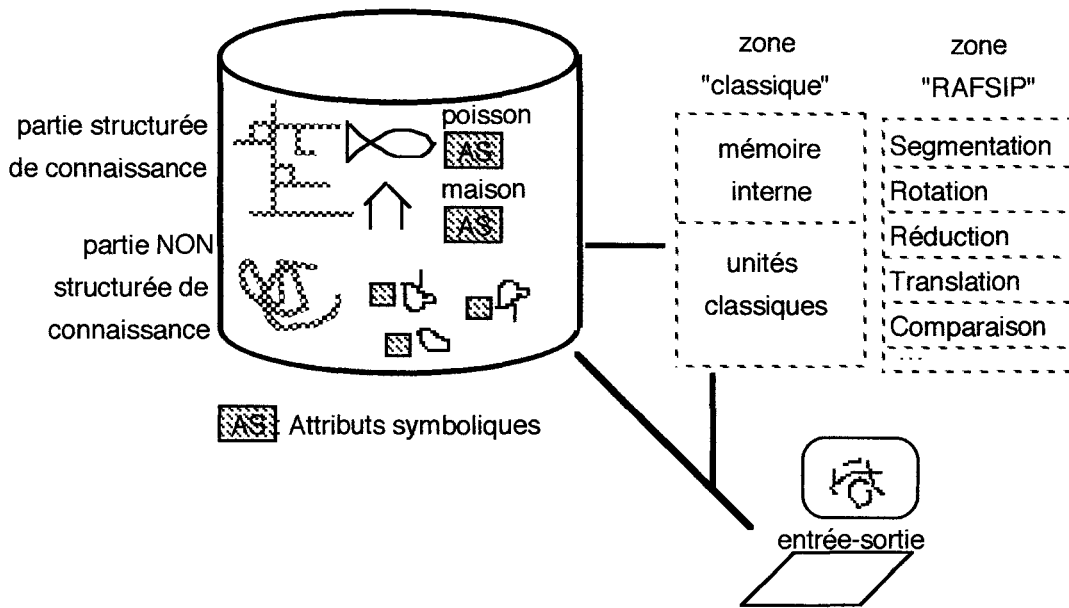
ROTATION : ce processeur réalise la rotation par rapport à un point, d'un angle donné, des pixels d'une rétine. Une première simulation de ce processeur a été réalisée par Jérôme DELEU dans le cadre de sa thèse. Les résultats obtenus pour une rétine de 256 x 256 pixels indiquent une durée du processus de 0,215 ms. A titre d'exemple, le très récent processeur graphique (Digital Signal Processing) T9506 de chez Toshiba mettrait environ une seconde pour tourner une image 512 x 512 pixels.

SEGMENTATION : ce processeur ne détecte que les segments à peu près horizontaux de quelques pixels de longueur. PASTIS utilise *ROTATION* pour détecter les segments ayant une autre inclinaison.

REDUCTION et *TRANSLATION* qui se combinent avec *ROTATION* pour *COMPARAISON*.

Ces processeurs seront utilisés aussi pour des données symboliques. La signature d'un dessin segmenté (c'est à dire la liste du nombre de segments selon chacune des directions d'analyse) est un exemple de donnée symbolique immédiate produite par la segmentation d'un dessin. Ainsi différentes données symboliques vont aider PASTIS dans la classification des ses connaissances. Une hypothèse produite par la voie symbolique sera confirmée ou infirmée par une comparaison directe des images.

05. La machine PASTIS illustrée.



Exemple de comportement de PASTIS en état de veille :

<p>•1•</p> <p>entrée</p>	<p>•2• opérations:</p> <p>segmentation rotation segmentation rotation,...</p>		<p>•3• sélection de l'information symbolique : "maisons tournée à droite" dans cette image puis rotation à gauche</p>		<p>•4• réduction à la taille normalisée</p>
	<p>•5• comparaison avec "maison"</p>		<p>•6• dialogue : est-ce bien une maison?</p> <p>maison ?</p>	<p>confirmation du concept de PASTIS enfant + partie incomprise:</p>	<p>•7• renforcement de "maison" et stockage de la partie incomprise</p> <p>↑ dans la partie non structurée de la connaissance</p>

Exemple de comportement de PASTIS en état de rêve

<p>ZZZ...</p>	<p>Détection de nouveaux concepts par exploration de la partie non structurée de la connaissance</p>		<p>sortie au réveil :</p> <p>qu'est ce que ceci ?</p>
---------------	---	--	--

Conclusion :

PASTIS est un projet de conception et réalisation d'une architecture spécialisée dans la symbolisation de dessins au trait.

Le parti pris est de sacrifier la précision à la rapidité, à tous les niveaux où des choix s'imposent.

PASTIS se propose d'intégrer des traitements simples dans une architecture basée sur des processeurs dédiés à des tâches spécialisées.

Chaque processeur tente de réduire la complexité de la tâche qui lui est assignée par un parti pris d'à peu près pour plus de rapidité.

PASTIS est une machine qui évolue dans le temps, dans un contexte de remises en cause de ses certitudes, en reconnaissance, en classification, en reproduction de dessins au trait, par un dialogue avec le monde extérieur qui lui ouvre les portes de l'apprentissage.

BIBLIOGRAPHIE

[DEUL1 91] DEULEU J.

" La machine PASTIS "

Thèse de doctorat Université de LILLE1. 1991.

[DEUL2 91] BINSE M., DAUCHET M., DELEU J., MERIAUX M.

" PASTIS 89 : A Specialised Computer which learns to draw "

IATED Eighth International Symposium Applied Informatics Innsbruck Autriche
20-23 Fev 90

[DEUL 90] DEULEU J.

" Présentation de la machine PASTIS "

APPLICA 1990.

Partie 2 - Chapitre 3.

Vision et dessins

où l'on présente et discute des mécanismes artificiels

Partie 2 - Chapitre 3 :

Le but de ce chapitre est de présenter le domaine de la vision des mécanismes artificiels.

Nous proposons d'abord un bref état des recherches en systèmes de vision, ce qui nous permet d'identifier d'une part un objet d'étude : l'image, et d'autre part deux tâches essentielles : la segmentation et l'interprétation.

Ces tâches mettent en œuvre des outils et des connaissances organisés en systèmes.

Nous allons nous intéresser alors à l'image pour identifier des modes de représentation, dont nous montrons qu'ils caractérisent des classes de méthodes.

Nous étudions ensuite chacune de ces méthodes.

01. Les systèmes de vision : état du domaine

1.1 Introduction

Les systèmes de vision s'articulent tous en deux étapes principales : la segmentation et l'interprétation de l'image.

Le but de la segmentation peut être :

- + soit découper une image en régions (ensembles de pixels vérifiant certaines propriétés) correspondant à des objets ou des parties d'objets ;
- + soit approcher les contours de l'objet par des lignes brisées de segments.

La segmentation met ainsi en relation spatiale des éléments d'image alors que la tâche d'interprétation va tenter de les mettre en relation "sémantique".

Ce problème "sémantique" n'est pas simple non plus : il s'agit de construire les modèles représentant les objets susceptibles d'apparaître dans l'image ; et de définir des stratégies d'identification des régions ou des contours isolés par la tâche de segmentation.

Ces deux tâches principales mettent en œuvre des outils et des connaissances spécifiques que nous décrivons tout d'abord. Nous insistons ensuite sur les modalités de leur usage et de leurs articulations.

1.2 Outils et connaissances

Dans le domaine de la segmentation [HARA 85] :

Deux classes d'outils se distinguent : les procédures (tâches élémentaires d'analyse d'image) et les algorithmes (enchaînements contrôlés de tâches élémentaires)

Parmi les procédures nous trouvons des opérations de filtrage, de calcul d'histogrammes, ...

Parmi les algorithmes nous trouvons ceux qui mettent en œuvre une même tâche dans un contexte itératif (ex : algorithme de "région growing"), ceux qui mettent en œuvre des suites de traitements (ex : algorithme de rehaussement puis détection puis lissage).

Critique : le succès de leur usage dépend de leur adéquation à la nature de l'image à analyser : adéquation non seulement du type de l'outil mais aussi de la séquence d'opérateurs qu'il met en œuvre et de leurs paramètres nécessitant ainsi à ce niveau une véritable expertise [DIXO 86] [GALL 88].

Dans le domaine de l'interprétation :

Deux classes d'approches apparaissent :

1 + Techniques de mise en correspondance s'appuyant sur des stratégies de prédiction vérification ou sur des techniques connexionnistes de relaxation. [LUX 85] [MOHR 88]

2 + Techniques basées sur des représentations par objets, de type déductives, fondées sur l'exploitation de règles de production, selon des stratégies guidées par l'ensemble des connaissances disponibles à un instant donné [GARB 87].

L'organisation logicielle est celle d'un système expert [NAZI 84], ou encore celle du "tableau noir" [HANSO 78] [HAYE 85].

Critique : dans tous les cas la recherche du meilleur compromis entre la représentativité et l'utilisation du modèle est essentielle.

1.3 Styles de conception et fonctions mises en œuvre

Les styles architecturaux et fonctionnels des systèmes de vision permettent d'identifier deux classes :

1 + les systèmes séparant les deux niveaux d'analyse : le module "de segmentation" et le module "d'interprétation".

2 + les systèmes hybrides, mettant en œuvre une coopération plus étroite entre les tâches.

Les systèmes séparés :

[MATS 88] présente une étude des systèmes experts réalisés dans le domaine de la segmentation d'image.

Module "de segmentation" :

La stratégie est la suivante : générer un plan d'analyse de l'image puis sélectionner les opérateurs et les paramètres optimaux de ceux-ci par essai - erreur. Le plan est généré grâce à des connaissances sur les modes de séquençement des opérateurs, et éventuellement sur l'image elle-même. La sélection des opérateurs est guidée par des connaissances sur leurs conditions d'application.

Module "d'interprétation" :

Ce module dispose de connaissances dites de "haut niveau" pour interpréter la segmentation fournie par le module "de segmentation". Il peut parfois fournir en retour de nouveaux objectifs au module "de segmentation".

MATSUYAMA distingue quatre types de tels systèmes ayant deux modules :

1 + les systèmes de consultation sollicitant la participation de l'utilisateur dans l'évaluation des résultats.

2 + les systèmes de création de programmes demandant, pour pouvoir fonctionner, la donnée par l'utilisateur d'un "pseudo programme" ou d'un exemple.

3 + les systèmes à base de règles travaillant par raffinement d'une segmentation initiale, en appliquant des règles de traitement et de contrôle guidées par la notion de zone d'intérêt.

4 + les systèmes de segmentation dirigés par l'image développant une stratégie de planification des opérateurs guidée par le but, s'appuyant sur un réseau décrivant la composition de l'image.

Critique : Les notions essentielles sont celles de focalisation, d'adaptation et de contrôle. Leur mise en œuvre suppose une modélisation approfondie de l'expertise en traitement d'image, imposée de manière innée au système.

Notre perspective sera plutôt inverse : le système construisant lui même sa propre expertise. D'autre part les résultats de l'interprétation sont souvent peu utilisés pour améliorer ou infléchir le processus de segmentation, le module d'interprétation étant même parfois supposé interpréter convenablement des images segmentées de manière déficiente.

Les systèmes hybrides :

[PORQ 86] [LUX 85] présentent des études détaillées, dont nous pouvons retenir les points suivants:

- + les systèmes faisant appel au principe de prédiction vérification, les hypothèses d'interprétation impliquant des contraintes sur la segmentation.
- + les systèmes introduisant la notion d'îlot de confiance.
- + les systèmes [KOHL 87] ayant la possibilité de revenir sur une segmentation lorsque son interprétation en est ambiguë, mais en disposant alors des conclusions de l'interprétation.
- + les systèmes de reconnaissance de cibles, exploitant des séquences d'images, réalisant le retour sur une image, d'informations extraites et interprétées de l'image précédente.

Critique : la tâche d'interprétation joue un rôle primordial : elle permet de guider et de contrôler la tâche d'analyse de l'image. Ceci implique qu'elle puisse disposer à cette fin d'une représentation détaillée et structurée du problème à résoudre, associée à une solide capacité de raisonnement.

1.4 Conclusion

Il apparaît ainsi que l'objet d'étude des systèmes de vision est l'image, que deux classes d'expertises sont nécessaires : celles relatives à l'utilisation des processus d'analyse de l'image et celles relatives aux caractéristiques du domaine d'application considéré. Ces formes d'expertise touchent cependant des champs de connaissances extrêmement divers, dont l'hétérogénéité suggère l'emploi de modes de représentation et d'exploitation adaptés.

Nous présentons maintenant les modes de représentation de l'image ou de ses caractéristiques et les méthodes qui en découlent.

02. L'image objet d'étude : ses représentations.

Les représentations d'une image se répartissent en cinq catégories caractérisées par des classes de méthodes :

Méthodes morphologiques : on code l'image comme un ensemble de pixels d'un plan (euclidien \mathbb{R}^2 ou discret \mathbb{Z}^2) sur lequel on va faire agir des transformations.

Le présumé est : un acte perceptif visuel peut se modéliser par des transformations ensemblistes.

Méthodes statistiques : on code l'image comme un vecteur de \mathbb{R}^p dont les composantes mesurent p caractéristiques définissant l'image.

Le présumé est : il y a beaucoup de chances pour que ...

Méthodes structurelles : on code l'image comme une structure informatique.

Le présumé est : il y a une forme, une structure dans l'image.

Méthodes analytiques : on code l'image par une équation sur les coordonnées de ses points dans un espace.

Le présumé est : il y a vérification d'une relation numérique.

Méthodes connexionnistes : on code l'image comme une configuration d'état d'un ensemble d'unités interconnectées.

Le présumé est : il y a des liens.

Nous observons ici que le mode de représentation de l'image enferme dans une classe de méthodes.

Nous étudions maintenant chacune de ces classes.

03. Les méthodes

3.1 . Méthodes de la morphologie

Contrairement aux sons qui se combinent algébriquement, les signaux visuels interagissent de manière ensembliste. En effet : les objets vus sont en général opaques et perçus en perspective, et puisque tout objet non transparent cache tout ce qui est placé derrière lui, on peut affirmer que "si le contour de l'objet le plus proche contient celui du plus lointain, alors il le cache complètement, sinon s'il lui est inclus alors ...".

Une morphologie, (c'est à dire une description de forme) si elle se veut mathématique doit donc s'exprimer dans le cadre adapté à l'inclusion, à savoir dans le cadre de la structure mathématique de treillis.

Une image sera tout simplement représentée par un ensemble E dont les éléments sont appelés pixels.

Une forme sera une partie de E, élément du treillis P(E) muni de la relation d'inclusion et des opérations d'intersection, union et complémentaire (le complémentaire de A est noté A^c).

Dans ce cadre, un acte perceptif visuel peut s'interpréter comme une transformation de formes, c'est à dire une application f de P(E) dans P(E) .

Situer une perspective d'étude par rapport à l'inclusion focalise l'intérêt sur deux propriétés de ces transformations : f est extensive et f est croissante.

Nous rappelons maintenant quelques définitions relatives à ces transformations modèles d'actes perceptifs, avant d'étudier quelques unes de leurs propriétés et définir les dilatations et érosions.

Ces transformations, actes perceptifs de formes, modélisent des façons d'agir sur des ensembles de pixels selon un point de vue identifié par un élément structurant.

Les propriétés de ces transformations rendent compte de la "perte d'information" résultant de leur application sur l'ensemble de pixels et permettent donc de la gérer au mieux.

Le paragraphe suivant définit quelques notions qui vont être utilisées par la suite pour décrire des situations.

Quelques définitions générales préalables:

D1 : *f est duale de f	<i>si et seulement si</i>	Pour tout X de P(E) , on a : $*f(x) = (f(x^c))^c$
D2 : g est un graphe	<i>si et seulement si</i>	g est une application de E dans P(E)
D3 : g^{-1} est graphe symétrique de g	<i>si et seulement si</i>	g^{-1} est l'application de E dans P(E) définie par pour tout x de E : $g(x) = \{y / x \in g(y)\}$
D4 : g est un graphe symétrique	<i>si et seulement si</i>	$g = g^{-1}$

D5 : Une application f de $P(E)$ dans $P(E)$ est croissante	<i>si et seulement si</i>	Pour tout X de $P(E)$, Y de $P(E)$, on a : si $X \subset Y$ alors $f(X) \subset f(Y)$.
D6 : Une application f de $P(E)$ dans $P(E)$ est extensive	<i>si et seulement si</i>	Pour tout X de $P(E)$ on a : $X \subset f(X)$
D7 : Une application f de $P(E)$ dans $P(E)$ est réunitive	<i>si et seulement si</i>	Pour toute famille $\{X_i\}_{i \in I}$ telle que $X_i \in P(E)$, on a $\bigcup_{i \in I} f(X_i) = f(\bigcup_{i \in I} X_i)$

Propriétés des applications réunitives :

P1 :	<i>si</i>	Une application f de $P(E)$ dans $P(E)$ est réunitive	<i>alors</i>	f est croissante
P2 :	<i>si</i>	Une application f de $P(E)$ dans $P(E)$ est réunitive	<i>alors</i>	L'application g de E dans $P(E)$ définie pour tout e de E par $g(e) = f(\{e\})$ est un graphe tel que : Pour tout X de $P(E)$ on a : $f(X) = \bigcup_{e \in X} g(e)$ g est le graphe associé à l'application réunitive f
P3 : inversement :	<i>si</i>	g est un graphe de E dans $P(E)$	<i>alors</i>	L'application f de $P(E)$ dans $P(E)$ définie pour tout X de $P(E)$ par $f(X) = \bigcup_{e \in X} g(e)$ est réunitive : f est l'application réunitive associée au graphe g

Notion de dilatation :

D8 : f est appelée dilatation .	<i>si et seulement si</i>	f est réunitive de $P(E)$ dans $P(E)$. Le graphe g associé à f est dit élément structurant de la dilatation
---	---------------------------	---

P4 : Pour tout X de $P(E)$ on a **dilatation $_g$** $(X) = \bigcup_{e \in X} g^{-1}(e) = X \oplus g^{-1}$

P5 : Pour tout X de $P(E)$ on a **dilatation $_g$** $(X) = \{e \in E / g(e) \cap X \neq \emptyset\}$

Notion d'érosion :

D9 : f est appelée érosion .	<i>si et seulement si</i>	f est dual d'une dilatation. Le graphe g associé à f est dit élément structurant de l'érosion
--	---------------------------	---

P6 : Pour tout X de $P(E)$: **érosion $_g$** $(X) = * \text{dilatation}_g(X)$
Pour tout X de $P(E)$: **érosion $_g$** $(X) = X \ominus g^{-1} = \bigcap_{e \in X^c} [g^{-1}(e)]^c = [X^c \oplus g^{-1}]^c$

P7 : Pour tout X de P(E) on a **érosion_g** (X) = { e ∈ E / g (e) ⊂ X }

Autres notions : ouverture, fermeture, HMT, amincissement, épaissement

D10 : Pour tout X de P(E) on a **ouverture_g** (X) = [X ⊕ g⁻¹] ⊖ g
 Pour tout X de P(E) on a **ouverture_g** (X) = **érosion_g** (**dilatation_g**⁻¹ (X))

D11 : Pour tout X de P(E) on a **fermeture_g** (X) = [X ⊖ g⁻¹] ⊕ g
 Pour tout X de P(E) on a **fermeture_g** (X) = **dilatation_g** (**érosion_g**⁻¹ (X))

D12 : Pour tout X de P(E) on a
HMT(g₁,g₂) (X) = **érosion_{g₁}** (X) / **dilatation_{g₂}** (X)

/ est l'opération ensembliste "différence" : X / Y = X ∩ Y^c

HMT(X) modélise ce que l'on gagne ou perd dans l'érosion X par g₁ ou la dilatation de X par g₂.

Son nom Hit or Miss Transformation l'indique bien. Les érosions et dilatations sont des cas particuliers de HMT. (en particulier l'érosion est obtenue avec le graphe g₂ tel que pour tout x de E on a g₂(x) = ∅).

P8 : Pour tout X de P(E) on a **HMT(g₁,g₂)** (X) = X ⊗ (g₁,g₂)

D13 : Pour tout X de P(E) on a
amincissement(g₁,g₂) (X) = X ⊖ (g₁,g₂) = X / (X ⊗ (g₁,g₂))

D14 : Pour tout X de P(E) on a
épaississement(g₁,g₂) (X) = X ⊕ (g₁,g₂) = X^c ⊖ (g₁,g₂)

Pour tout X de P(E) on a
épaississement(g₁,g₂) (X) = ***amincissement(g₁,g₂)** (X)

+ Exemples :

Le plan euclidien peut être modélisé par le corps des complexes \mathbb{R}^2

Nous allons définir un élément structurant non symétrique, et observer l'effet des principales transformations définies ci-dessus sur un ensemble donné de pixels.

Définition d'un élément structurant g :

pour tout point x de \mathbb{R}^2 : g(x) = le triangle équilatéral de côté 1, centré en x (pointe en "l'air").

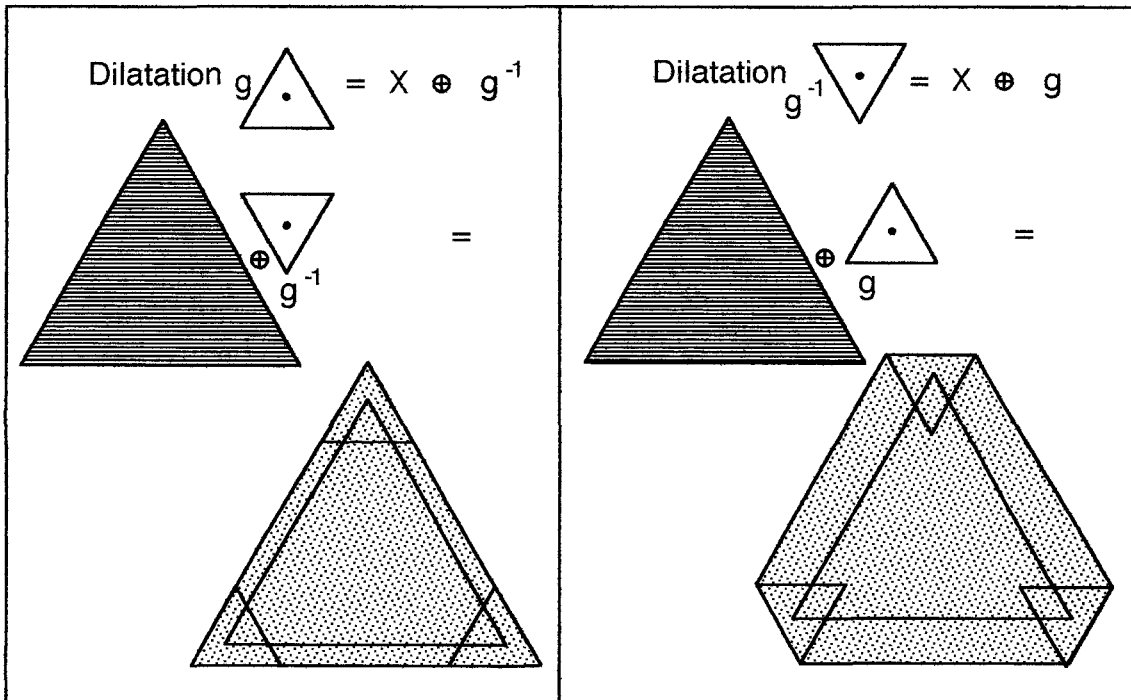
Ce graphe g n'est pas symétrique.

Son symétrique g⁻¹ est défini par :

pour tout point x de \mathbb{R}^2 : g⁻¹(x) = le triangle équilatéral de côté 1, centré en x (pointe en "bas").

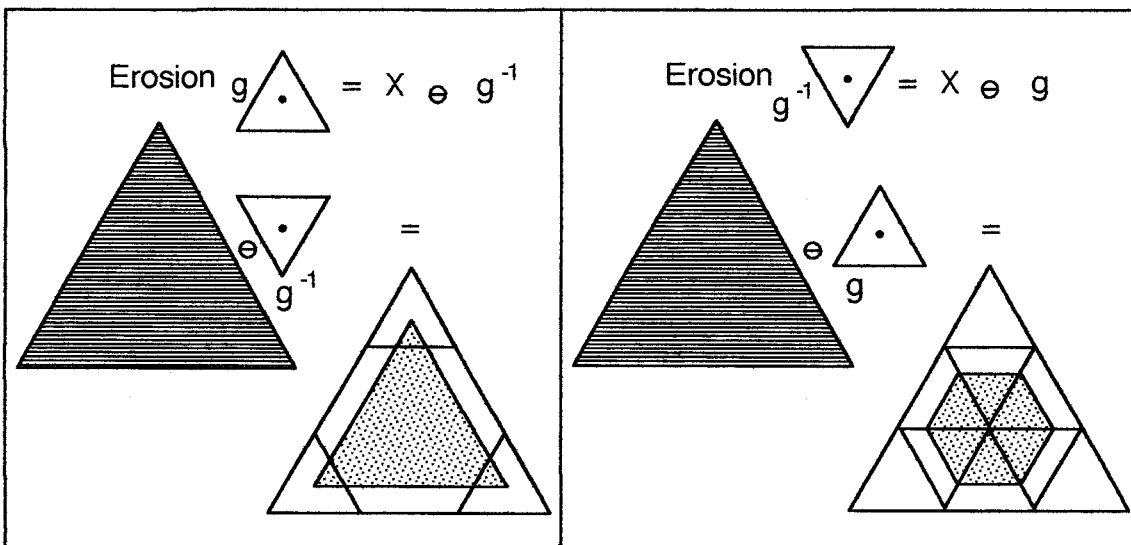
La forme, ensemble de pixels sur lequel on fera agir les transformations sera un triangle équilatéral de côté 3 (pointe en "l'air").

dilatations :



On remarque que l'on n'obtient pas la même dilatation avec un élément structurant et avec son symétrique. Dans la grande majorité des cas, on prendra un élément structurant symétrique, et on aura donc les mêmes dilatations.

érosions :



Ici aussi nous n'obtenons pas le même érodé avec un élément structurant et son symétrique.

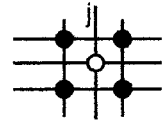
Intéressons nous maintenant à illustrer l'effet de ces transformations dans un plan discret avec un élément structurant symétrique cette fois.

Le plan discret peut être modélisé par \mathbf{Z}^2 selon deux points de vue : la maille carrée ou la maille hexagonale :

En maille carrée :

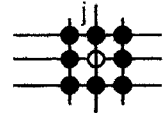
Définition d'un élément structurant g_4 :

pour tout (i, j) de \mathbf{Z}^2 : on a $g_4((i, j)) = \{ (i+1, j+1) ; (i-1, j+1) ; (i+1, j-1) ; (i-1, j-1) \}$



Définition d'un élément structurant g_8 :

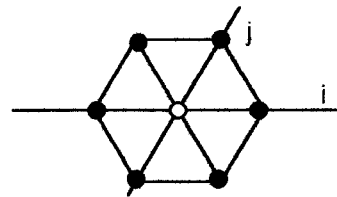
pour tout (i, j) de \mathbf{Z}^2 : on a $g_8((i, j)) = \{ (i+1, j+1) ; (i, j+1) ; (i-1, j+1) ; (i+1, j-1) ; (i, j-1) ; (i-1, j-1) ; (i+1, j) ; (i-1, j) \}$



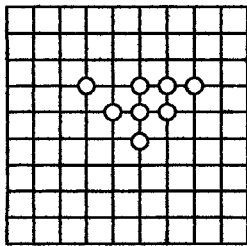
En maille hexagonale :

Définition d'un élément structurant g_6 :

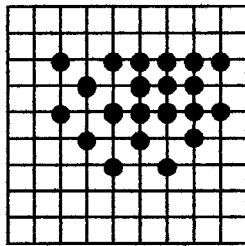
pour tout (i, j) de \mathbf{Z}^2 : on a $g_6((i, j)) = \{ (i+1, j) ; (i-1, j) ; (i+1, j-1) ; (i, j-1) ; (i-1, j+1) ; (i, j+1) \}$



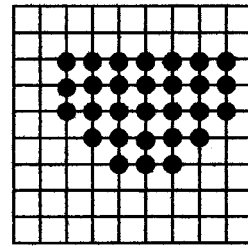
Exemple 1 : en maille carrée : effet dilatation des éléments structurants g_4 et g_8



pixels sur lesquels on agit

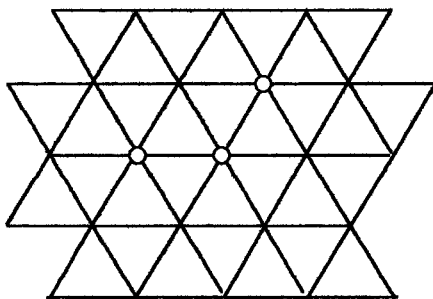


Effet de g_4

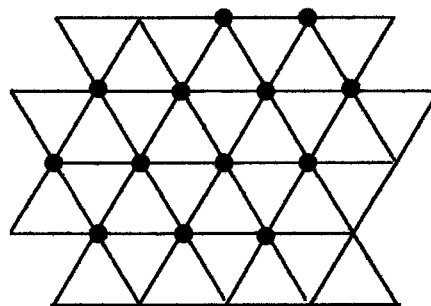


Effet de g_8

Exemple 2 : en maille hexagonale : effet dilatation de l'élément structurant g_6



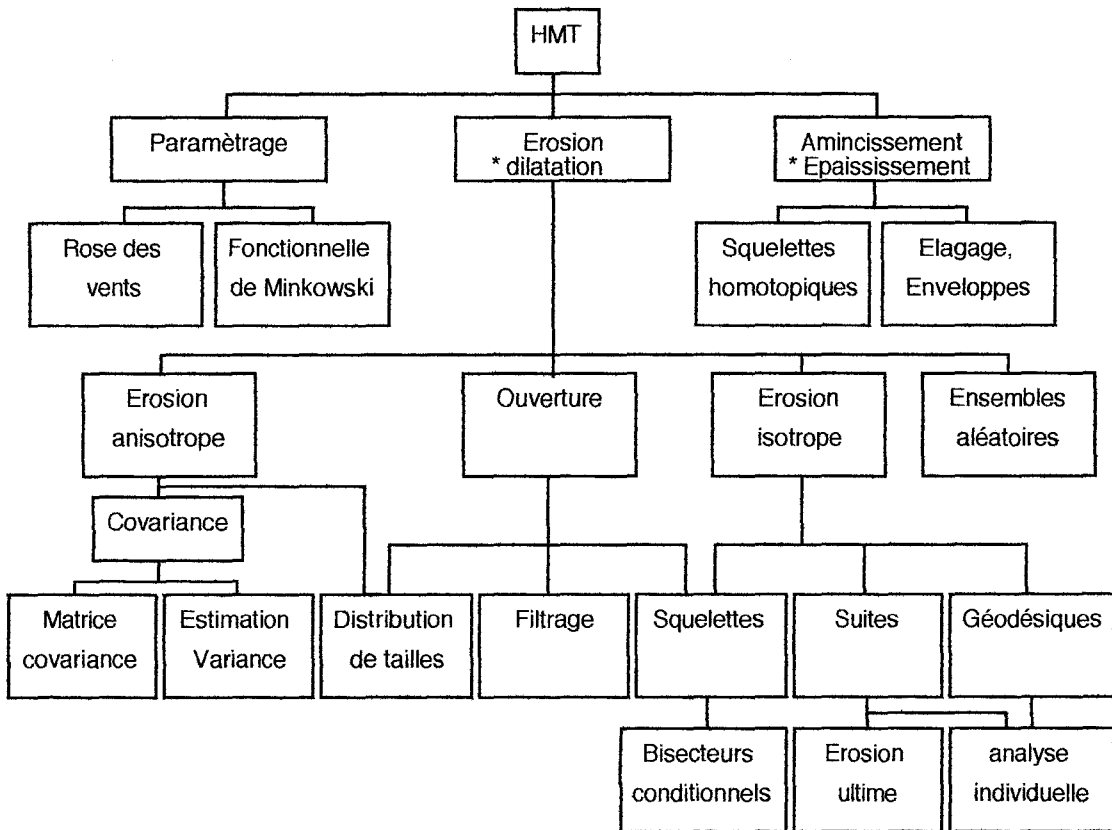
pixels sur lesquels on agit



Effet de g_6

+ Hiérarchie des transformations de la morphologie mathématique :

Nous disposons ainsi d'après SERRA de toute une hiérarchie de transformations.



+ Conclusion :

Nous avons vu que l'on pouvait modéliser des actes perceptifs visuels par des transformations, des façons d'agir sur des pixels selon un point de vue identifié par un élément structurant. Ces transformations peuvent permettre de mettre en évidence dans une image des ensembles de pixels représentatifs par exemple d'une certaine forme recherchée.

3.2 . Méthodes statistiques :

Une image peut être perçue comme une grande masse de données d'où l'on cherche à extraire des informations utiles dans le but de réaliser une classification.

Nous cherchons à appliquer des méthodes statistiques de classification automatique pour reconnaître des formes dans une image.

Nous allons nous intéresser tout d'abord aux différents types de classifications réalisables : les partitions, les recouvrements, les hiérarchies et les pyramides, qui se formalisent dans la notion d'espace de classification. Les classes ainsi identifiées seront représentées. Nous étudions ensuite les espaces de représentation et enfin un algorithme : "les nuées dynamiques".

Définissons quelques types de classifications :

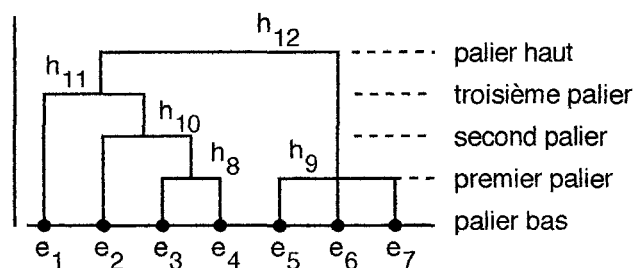
1 • une **partition** d'un ensemble E fini est un ensemble $P = (P_1, \dots, P_k)$ de parties non vides de E, d'intersections vides deux à deux et dont la réunion forme E.

2 • un **recouvrement** d'un ensemble E fini est un ensemble $R = (R_1, \dots, R_k)$ de parties non vides de E, dont la réunion forme E. (une partition est un cas particulier de recouvrement).

3 • une **hiérarchie** d'un ensemble E fini est un ensemble $H = (H_1, \dots, H_k)$ de parties non vides de E appelées paliers tels que

- (1) $E \in H$ (le palier le plus haut contient tous les éléments)
- (2) pour tout e de E, on a $\{e\} \in H$ (tous les plus bas paliers sont les singletons)
- (3) pour tout h et h' de H on a $h \cap h' \neq \emptyset \Rightarrow h \subset h'$ ou $h' \subset h$ (emboîtement hiérarchique)

Une hiérarchie H d'un ensemble E peut se représenter par un dendogramme :



Exemple de hiérarchie d'un ensemble $E = \{ e_1, \dots, e_7 \}$ représentée par un dendogramme.

On a $H = \{ \{e_1\}, \dots, \{e_7\}, h_8, h_9, h_{10}, h_{11}, h_{12}, \}$ remarque : $h_{12} = E$

Si l'on donne une "valeur" aux paliers d'une hiérarchie, on dit définir une **hiérarchie indicée**.

La hiérarchie apparaît alors comme un emboîtement de partitions de moins en moins fines du bas vers le haut:

Tout en bas la partition la plus fine formée de tous les singletons, tout en haut la partition la moins fine formée d'un seul ensemble $h_{12} = E$.

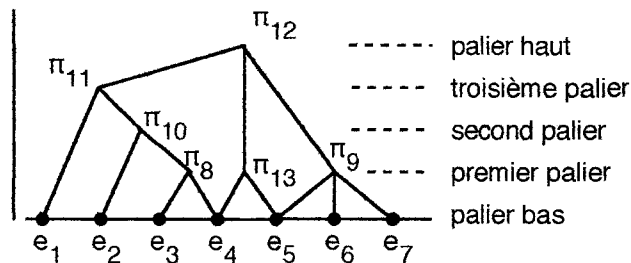
Par exemple entre le second et le troisième palier, une droite horizontale coupe la hiérarchie en trois points, ce qui donne une partition en trois ensembles :

$$P = \{ \{e_1\}, h_9, h_{10} \}$$

4 • une **pyramide** d'un ensemble E fini est un ensemble $\Pi = (\Pi_1, \dots, \Pi_k)$ de parties non vides de E appelées paliers tels que

- (1) $E \in \Pi$ (le palier le plus haut contient tous les éléments)
- (2) pour tout e de E, on a $\{e\} \in \Pi$ (tous les plus bas paliers sont les singletons)
- (3) pour tout π et π' de Π on a $\pi \cap \pi' \neq \emptyset \Rightarrow \pi \cap \pi' \in \Pi$
- (4) il existe un ordre θ tel que tout élément de la pyramide Π soit un intervalle de θ

Une pyramide Π d'un ensemble E peut se représenter par un dendogramme :



Exemple de pyramide d'un ensemble $E = \{ e_1, \dots, e_7 \}$ représentée par un dendogramme.

On a $\Pi = \{ \{e_1\}, \dots, \{e_7\}, \pi_8, \pi_9, \pi_{10}, \pi_{11}, \pi_{12}, \pi_{13} \}$

Formalisons maintenant la notion d'espace de classification, en mettant en évidence les propriétés communes aux partitions, recouvrements, hiérarchies et pyramides.

Définition :

S constitue un espace de classification sur l'ensemble E fini, si S satisfait aux propriétés suivantes :

- 1• tout élément s de S est un ensemble de parties de E.
- 2• tout élément s de S recouvre E.
- 3• il existe au moins un élément s de S qui contient E. (une des classifications utilise E).
- 4• il existe au moins un élément s de S qui contient tous les singletons de E.

Exemple : soit $E = \{ e_1, e_2, e_3, e_4, e_5 \}$ un espace S de classification sur E est par exemple :

$$S = \{ \{E\}, \{ \{e_1, e_2, e_3\}, \{e_4, e_5\} \}, \{ \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_5\} \} \}$$

Autre exemple : l'ensemble de toutes les partitions d'un ensemble E est un espace de classification.

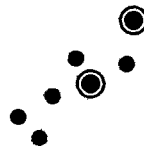
Il en est de même pour les ensembles de tous les recouvrements, de toutes les hiérarchies et de toutes les pyramides d'un ensemble E fini.

Formalisons maintenant la notion d'espace de représentation.

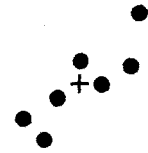
Un groupe d'éléments, une classe, peut être représenté par :



une droite



un ou des points de la classe



un centre de gravité

Le nom générique donné à l'une quelconque de ces représentations est "noyau".

Définissons la notion d'espace de représentation :

Définition :

Tout ensemble peut être un espace de représentation.

On distinguera l'ensemble L appelé espace de représentation d'une classe et l'ensemble L_k appelé espace de représentation d'une partition de P_k .

Illustrons par quelques exemples :

Soit $E = [1, n]^2$, un carré du plan discret (ensemble de n^2 pixels) appelé rétine.

On peut choisir E comme espace de représentation d'une classe : $L = E$.

On peut choisir $E \times \dots \times E = E^k$ comme espace de représentation d'une partition en k classes de P_k

Une classe de la partition est donc représentée par un pixel "représentatif" de la rétine, le k uplet des classes de la partition en k classes par un k uplet de pixels de la rétine

Définition :

Un ensemble E est muni d'une structure de représentation d'une partition P_k si on lui associe :

- 1• un ensemble L appelé espace de représentation d'une classe
- 2• une application D de $P(E) \times L$ dans \mathbf{R}^+ appelée mesure de dissemblance entre une classe de $P(E)$ et une représentation de classe de L
- 3• un ensemble L_k appelé espace de représentation d'une partition P_k
- 4• une application W de $P_k \times L_k$ dans \mathbf{R}^+ appelée critère de mesure de l'adéquation entre une partition de P_k et une représentation de partition de L_k

Une fonction g permettant d'associer à une partition de P_k , un élément de L_k est appelée fonction de représentation.

Une fonction f permettant d'associer à un élément de L_k , une partition de P_k est appelée fonction d'affectation.

Exemples de structures de représentation d'une partition P_k

Si l'on reprend l'exemple ci-dessus de la rétine, il reste à préciser :

la mesure de dissemblance entre une classe et une représentation de classe :

$D(A, x) =$ inertie de A par rapport à x

la mesure d'adéquation entre une partition et une représentation de partition :

$$W(P, L) = \sum_{n=1}^k D(P_n, L_n)$$

La fonction de représentation g est par exemple $g(P) = L$ avec $L = (L_1, \dots, L_n, \dots, L_k)$ soit les k centres de gravité discrets des k classes de la partition P , ce sont les k éléments de la rétine qui minimisent $D(P_n, \dots)$

La fonction d'affectation f est définie par $f(L) = P$ avec $P_n = \{x \in E \text{ tels que pour tout } m \text{ de } \{1, \dots, k\} \text{ on ait } \text{distance}(x, L_n) \leq \text{distance}(x, L_m) \text{ et } n < m \text{ en cas d'égalité}\}$

Nous allons montrer qu'une grande famille de problèmes de la classification automatique peuvent s'énoncer en termes d'optimisation d'un critère mathématiquement bien défini.

Le cadre général permettant d'énoncer ces problèmes et d'obtenir des algorithmes pour leur donner des solutions approchées est celui des "nuées dynamiques".

La méthode des nuées dynamiques a introduit une problématique nouvelle, en cherchant à optimiser un critère qui exprime l'adéquation entre une classification d'objets et un mode de représentation des classes de cette classification.

Le problème d'optimisation se pose en termes de recherche simultanée de la classification et de la représentation des classes de cette classification parmi un ensemble de classifications et de représentations possibles, qui optimisent ce critère.

L'ensemble des partitions de E constitue un exemple privilégié d'espace de classification.

L'ensemble des centres de gravité des classes de la partition constitue un exemple classique d'ensemble de représentation.

Malheureusement, un algorithme de type "nuées dynamiques", comme la plupart des algorithmes de partitionnement conduit à des optimum locaux du critère et la solution obtenue dépend de la partition ou de la représentation choisie à l'initialisation de l'algorithme, de plus elle nécessite le choix à priori d'un nombre k de classes au départ. Ces deux problèmes ne sont pas encore résolus.

Algorithme des "nuées dynamiques"

Énoncé :

Trouver un couple (P^*, L^*) de $P_k \times L_k$ où P_k est l'ensemble des partitions en k classes d'un ensemble E et L_k un espace de représentation des classes de P_k qui minimise un critère $W : P_k \times L_k \rightarrow \mathbb{R}^+$, autrement dit tel que $W(P^*, L^*) = \text{Min} \{ W(P, L) \text{ tels que } P \in P_k \text{ et } L \in L_k \}$

On note g la fonction de représentation : $g : P_k \rightarrow L_k$ et f la fonction d'affectation : $f : L_k \rightarrow P_k$

algorithme :

à partir d'une solution initiale $v_0 = (P^0, L^0)$ estimée ou tirée au hasard, itérer jusqu'à stabilisation de la suite v par : $v_{n+1} = (P^{n+1}, L^{n+1})$ où $P^{n+1} = f(L^n)$ et $L^{n+1} = g(P^{n+1})$ et $u_n = W(v_n)$

On prouve que la suite u converge en décroissant et que la suite v est stationnaire à partir d'un certain rang fini.

Tableau de différents algorithmes de type “nuées dynamiques” selon les modes de représentation (la classification est toujours une partition en k classes) :

Critère, fonction d'affectation et de représentation :

- $W(P,L) = \sum_{n=1}^k D(P_n, L_n)$ où $D(A,x)$ est l'inertie de A par rapport à x.
- $g(P)$ = les centres de gravité des classes de P
- $f(L) = P$ où les $P_n = \{ x \in E \text{ tels que pour tout } m \text{ de } \{1, \dots, k\} \text{ on ait } d(x, L_n) \leq d(x, L_m) \text{ et } n < m \text{ en cas d'égalité} \}$

Représentation	Nom de l'algorithme	Référence
par centre de gravité	algorithme des centres mobiles	[BENZECRI 1973]
par loi de probabilité	algorithme de décomposition de mélanges	[SCHROEDER 1976]
par distance	algorithme des distances adaptatives	[FRIEDMAN 1967]
par régression	algorithme de régression typologique	[Charles 1977]
par axe factoriel	algorithme d'analyse factorielle typologique	[Ok 1975]
par axe discriminant	algorithme d'analyse discriminante typologique	[LEMOINE 1979]
par variables	algorithme de segmentation typologique	[Meunier 1986]
par individus	algorithme des tableaux de distances	
par courbes	algorithme de lissage typologique	[Ok 1975]

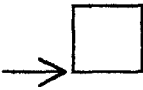
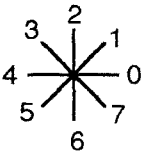
3.3 . Méthodes structurales .:

Tout naturellement, les structures informatiques disponibles, sont utilisées pour représenter l'image. Nous présentons ci-dessous les différentes structures informatiques codantes et les méthodes et algorithmes correspondants.

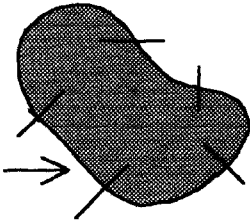
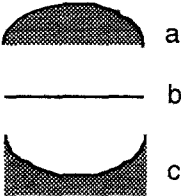
3.3.1. On code la forme comme une chaîne.

C'est à dire comme une liste de composants élémentaires pris dans un alphabet.

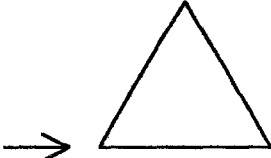
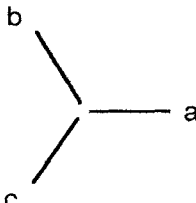
Exemple 1 : le code de FREEMAN : on code huit directions de segments constitutifs du contour.

forme :	code :	alphabet :	codage
		{0, 1, 2, 3, 4, 5, 6, 7}	carré = 0246

Exemple 2 : le code de convexité : on code trois convexités du contour : droit, concave, convexe.

forme :	code :	alphabet :	codage
		{a, b, c}	patate = baaca

Exemple 3 : le code à trois directions

forme :	code :	alphabet :	codage
		{a, b, c}	triangle équilatéral = aabbcc

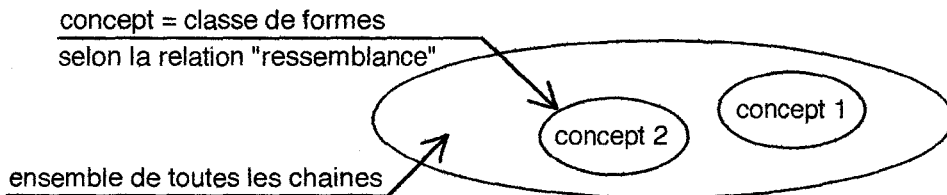
structure de la forme : une chaîne de caractères d'un alphabet

ensemble d'apprentissage : un ensemble de chaînes de caractères

techniques de reconnaissance et de classification des formes :

idée 1 :

on établit des liens de "ressemblance" entre les éléments de la forme. Cette relation de "ressemblance" étant dotée des qualités d'une relation d'équivalence, on génère ainsi des classes de formes correspondant aux concepts à reconnaître.



méthodes et algorithmes :

CC comparaison de chaînes : [MORRIS PRATT KNUTH]

problème : Rechercher un motif dans une forme chaîne

principe : On manipule un indicateur P qui indique la longueur du plus long préfixe de M reconnu comme sous texte de F au cours de l'exploration. Une fonction Lien qui ne dépend que du motif M permet de faire évoluer P.

lexique : M est le motif recherché dans la forme F c'est un texte de m caractères
F est la forme explorée : c'est un texte de f caractères. on a $m \leq f$.

algorithme :

```
procédure Créer-Lien
DEBUT { de création }
Lien(1) := 0
POUR j := 2 à m
FAIRE u := Lien(j - 1)
TANT QUE M(j) ≠ M(u+1) et u ≠ 0 FAIRE u := Lien(u)
SI M(j) ≠ M(u+1) et u = 0
ALORS Lien(j) := 0
SINON Lien(j) := u+1
FIN { de création }
```

```
procédure ComparaisonChaîne ( M, F : texte )
DEBUT { comparaison chaîne }
i := 1 ; j := 1
TANT QUE i ≤ f
FAIRE TANT QUE j ≠ 0 et M(j) ≠ F(i) FAIRE j := Lien(j)
SI j = m ALORS "succès" SINON i := i + 1 ; j := j + 1
FIN { comparaison chaîne }
```

DE : distance d'édition [WAGNER FISCHER]

problème : Calculer une distance entre deux motifs à une certaine "confusion" près

principe : On définit trois opérations élémentaires qui modélisent les confusions autorisées :

- la substitution d'un élément a du motif par un autre b : de coût $C(a, b)$
- l'insertion d'un élément novateur a dans le motif : de coût $C(f, a)$
- la destruction d'un élément a du motif : de coût $C(a, f)$

Par exemple le motif ab peut être transformé en le motif bac par la suite des transformations : insertion(c) suppression(b) substitution(c, b) insertion(c) donnant la suite des motifs : ab, cab, ca, ba, bac de coût total $C(f, c) + C(b, f) + C(c, b) + C(f, c)$

On définit la distance d'édition $D(M1, M2)$ entre deux motifs $M1$ et $M2$ comme le coût de la suite de transformations élémentaires la moins coûteuse pour transformer $M1$ en $M2$ (distance de LEVENSTEIN)

Soit $D(i, j)$ la distance entre le préfixe de longueur i de $M1$ et le préfixe de longueur j de $M2$: on a la propriété de récurrence :

$$D(i, j) = \text{Min} \{ D(i-1, j-1) + C(M1(i), M2(j)) ; D(i-1, j) + C(M1(i), f) ; D(i, j-1) + C(f, M2(j)) \}$$

lexique : $M1$ et $M2$ sont les deux motifs dont on cherche à évaluer la distance $D(i, j)$ est un tableau des distances entre préfixes de longueur i de $M1$ et le préfixe de longueur j de $M2$

$n1$ est la longueur du motif $M1$ et $n2$ la longueur du motif $M2$

algorithme :

```
fonction DE ( M1, M2 : Texte ) : réel;
{ distance entre les motifs M1 et M2 }
DEBUT { de DE }
  D(0,0) := 0;
  POUR i de 1 à n1 FAIRE D(i,0) := D(i-1,0) + C(M1(i),f) FAIT
  POUR j de 1 à n2 FAIRE D(0,j) := D(0,j-1) + C(f,M2(j)) FAIT
  POUR i de 1 à n1
  FAIRE  POUR j de 1 à n2
  FAIRE
    D(i, j) :=
    Min(D(i-1,j-1)+C(M1(i),M2(j));D(i-1,j)+C(M1(i),f);D(i,j-1)+C(f,M2(j)) )
  FAIT
  FAIT
  DE := D(n1,n2) { La distance d'édition entre M1 et M2 est D(n1,n2) }
FIN { de DE }
```

La complexité de cet algorithme est en $O(n1 \cdot n2)$

CD : Comparaison dynamique ou programmation dynamique : []

problème : Soit un processus P décomposable en n phases.
 A chaque phase i, une décision X_i est prise, permettant de passer de l'état E_{i-1} à l'état E_i .
 Soient X l'ensemble des décisions et E l'ensemble des états possibles.
 L'état E_i est supposé ne dépendre que de E_{i-1} et de X_i
 Le passage de E_{i-1} à E_i est valué par une fonction de coût r_i qui ne dépend que de E_{i-1} et X_i .

Le problème posé est de trouver le coût optimal et le chemin correspondant.

principe : Soit R le coût d'un chemin $E_0, \dots, E_i, \dots, E_n$ et r_i le coût de la transition de E_{i-1} à E_i par la décision X_i

Si la fonction R est décomposable
 [c'est à dire $R(u, v, w) = g(u, h(v, w))$ et g est monotone non décroissante]

alors MITTEN [1964] a prouvé que :

$$\underset{u,v,w}{\text{opt}} R(u, v, w) = \underset{u}{\text{opt}} g(u, \underset{v,w}{\text{opt}} h(v, w))$$

Si la fonction R est décomposable et si les coûts r_i sont positifs

alors on pourra écrire :

$$R = R_1 [r_1 (E_0, X_1), R'_1 [r_2 (E_1, X_2), \dots, r_n (E_{n-1}, X_n)]]$$

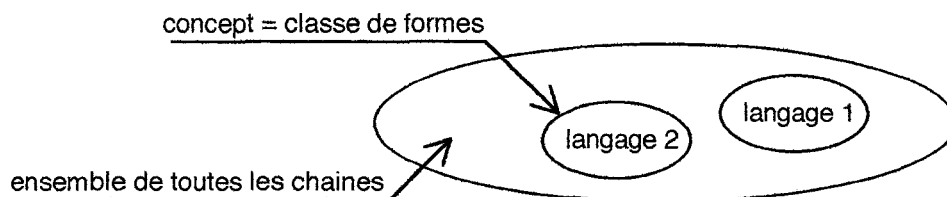
et donc : $R_{\text{opt}} = \underset{X_1}{\text{opt}} R_1 (r_1, \underset{X_2..X_n}{\text{opt}} R'_1 (r_2, \dots, r_n))$

soit : $R_{\text{opt}} = \underset{X_1}{\text{opt}} R_1 (r_1, \underset{X_2}{\text{opt}} R_2 (r_2, \dots, \underset{X_n}{\text{opt}} (r_n) \dots))$

ce qui établit une formule récursive sur laquelle est basé l'algorithme.

idée 2 :

Les classes de formes correspondant aux concepts sont des langages réguliers ou algébriques, sous ensembles particuliers de toutes les chaînes possibles codant toutes les formes.



méthodes et algorithmes :

AS analyse syntaxique: (reconnaissance) []

problème : Étant donnée un automate d'états finis A, **Décider** si une phrase x est élément du langage accepté par cet automate d'états finis.

principe : Notons $x = a_1 a_2 \dots a_n$; on construit la suite d'états $q_1 = \delta(q_0 a_1)$; $q_2 = \delta(q_1 a_2)$; ... ; $q_n = \delta(q_{n-1} a_n)$; si q_n est un état final alors la phrase x est reconnue sinon elle ne l'est pas.

IR : inférence régulière : (apprentissage)[$uv^k w$] [k finales] [agrégation de finales]

problème : Étant donné un ensemble fini F de phrases (échantillon d'apprentissage) **Trouver** une grammaire $G = (A, V, \sigma, P)$ telle que F soit complet par rapport à G.

F est dit complet par rapport à une grammaire G si :

- 1 • $L(G) \supset F$
- 2 • L'alphabet sur lequel est écrit F est égal à A
- 3 • Toutes les règles de P sont utilisées au moins une fois dans une génération d'un phrase de F.

Le problème posé de cette manière admet un grand nombre de solutions.

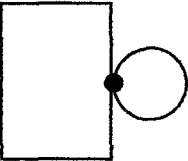
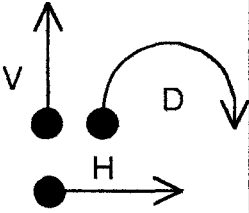
principe : de l'algorithme $uv^k w$

On recherche dans l'échantillon F les traces des règles récursives caractéristiques de la grammaire qui l'a engendré.

3.3.2. On code la forme comme un terme.

C'est à dire comme une formule, cas particulier d'arbre

Exemple 1 : Code de SHAW langage de description de forme à l'aide d'opérateurs sur des descripteurs

forme :	code :	alphabet :	codage
	superposition	!	forme =
	juxtaposition (bout à bout)	+	H : vecteur horizontal
	opposé	-	D : vecteur demi tour
			chope à bière = $(V + -H) !$ $(D ! -D) +$ $-V + -V +$ $H + V$

méthodes et algorithmes :

Définition :

Un terme est défini à partir :

- d'un ensemble de symboles fonctionnels $S = \{ f, g, a, \dots \}$, chaque symbole étant interprété comme un nom de fonction à n ($n \geq 0$) arguments
- et d'un ensemble $V = \{ x, y, \dots \}$ de variables :
Exemple : $f(g(x), a, h(g(y), z))$.

Un terme sera représenté par un arbre non limité étiqueté par $V' = V \cup S$

Opérations : Terme désigne le type des termes et ListeTermes le type liste de termes

racine : Terme $\rightarrow V'$

fil : Terme \rightarrow ListeTermes

composer : .. S x Terme \rightarrow Terme

feuille : V \rightarrow Terme

tête : ListeTermes \rightarrow Terme

reste : ListeTermes \rightarrow ListeTermes

adj : Terme x ListeTermes \rightarrow Terme

S substituer : []

Substitution :

on appelle Substitution toute fonction $V \rightarrow \text{Terme}$, qui à chaque variable associe le terme qui doit lui être substitué. Notons sigma une telle substitution.

problème : Substituer (t , sigma) : substituer dans le terme t selon les indications de la substitution sigma, c'est **générer** un terme dans lequel toute occurrence des variables est remplacée par son terme remplaçant.

principe : On va utiliser une fonction sigma représentée par une table qui associe à chaque variable le terme particulier à substituer ou elle même s'il n'y a pas de substitution à opérer pour cette variable pour agir sur la racine du terme. Pour la liste fille, on composera la substitution de la tête à celle du reste de la liste.

algorithme : détaillons l'algorithme substituer

substituer : Terme x Substitut \rightarrow Terme : c'est l'opération qui va substituer dans un terme donné toutes les variables par le terme à lui substituer défini par la fonction Substitution.

Soit t un terme et sigma une substitution, on a :

fonction **substituer** (t : Terme , sigma : Substitution) : terme

```
DEBUT { de substituer }
  SI racine ( t )  $\in$  V (c'est une variable)
  ALORS sigma ( racine ( t ) )
  SINON composer ( racine ( t ) , substituerListe ( fils ( t ) ) )
FIN { de substituer }
```

fonction **substituerListe** (liste) : liste

```
DEBUT { de substituerListe }
  SI vide ( liste )
  ALORS ListeVide
  SINON adj ( substituer ( tête ( liste ) , substituerListe ( reste ( liste ) ) )
FIN { de substituerListe }
```

AU antiunifier: (ou généraliser) [PLOTKIN 1970]

C'est l'algorithme de base de l'apprentissage symbolique

problème : Antiunifier (t_1, t_2) c'est **générer** un terme t qui "généralise" au mieux deux termes donnés t_1 et t_2 .

principe :

si les deux termes t_1 et t_2 sont des variables ou si $\text{racine}(t_1) \neq \text{racine}(t_2)$

alors on généralise t_1 et t_2 par une variable t de nom arbitraire

sinon on a $\text{racine}(t_1) = \text{racine}(t_2)$ et le généralisé t sera de racine $\text{racine}(t_1)$ c'est à dire $t = \text{composer}(\text{racine}(t_1), l_t)$ où l_t est l'antiunifié des termes qui se correspondent dans les listes fils (t_1) et fils (t_2) mais en prenant garde à ne reprendre une variable déjà utilisée que si on lui associe dans sigma1 et sigma2 les mêmes termes que précédemment.

algorithme : détaillons l'algorithme "antiunifier"

AntiUnifier : Terme \times Terme \rightarrow Terme est une fonction qui fournit un nouveau terme à partir de deux termes en les unifiant, c'est à dire en fournissant leur "meilleure" généralisation.

fonction **AntiUnifier** (t_1, t_2 : Terme , var $\text{sigma1}, \text{sigma2}$: Substitution);

{ sigma1 et sigma2 sont les substitutions qui génèrent l'AntiUnifié t à partir de t_1 pour sigma1 et à partir de t_2 pour sigma2 }

procédure **Anti** (t_1, t_2 : Terme , var t : Terme)

{ effectue l'antiUnification de t_1 et t_2 en tenant compte des variables déjà affectées par sigma1 et sigma2 }

var liste : ListeTermes;

DEBUT { de **Anti** }

SI $\text{racine}(t_1) \in S$ et $\text{racine}(t_1) = \text{racine}(t_2)$

ALORS { antiUnification des sous termes }

AntiListe (fils (t_1) , fils (t_2) , liste)

$t := \text{composer}(\text{racine}(t_1), \text{liste})$

SINON $x := \text{recherche}(t_1, \text{sigma1}, t_2, \text{sigma2})$

SI $x =$

ALORS $\text{sigma1} := \text{ajout}(y, t_1, \text{sigma1})$

$\text{sigma2} := \text{ajout}(y, t_2, \text{sigma2})$

$t := \text{feuille}(y)$ $y := \text{succ}(y)$

SINON $t := \text{feuille}(x)$

FIN { de **Anti** }

procédure **AntiListe** (liste1, liste2 : ListeTermes , var listeAntiUnifiée : ListeTermes)

{ effectue l'antiUnification de t_1 et t_2 en tenant compte des variables déjà affectées par sigma1 et sigma2 }

var te : terme ; li : ListeTermes

DEBUT { de **AntiListe** }

SI vide (liste1)

ALORS listeAntiUnifiée := listeVide

SINON Anti (tête (liste1) , tête (liste2) , te)

AntiListe (reste (liste1) , reste (liste2) , li)

listeAntiUnifiée := adj (te , li)

FIN { de **AntiListe** }

DEBUT { de **AntiUnifier** }

$\text{sigma1} := \text{identité}$; $\text{sigma2} := \text{identité}$;

$y := \text{minvariable}$;

anti (t_1, t_2, t) ; AntiUnifier := t

FIN { de **AntiUnifier** }

F filtrer: []

Algorithme de reconnaissance de motif

problème : Filtrer (t, u) : Filtrer un terme t par un terme u, c'est **générer** une substitution sigma telle que substituer (t , sigma) = u. C'est à dire que Filtrer (t, u) est l'opération inverse de substituer. Si le filtrage n'est pas possible Filtrer (t, u) fournit la substitution non définie nil.

principe : On génère le substitut par ajouts successifs de couples (variable x , terme v). Si on a successivement les couples (x, v) et (x, v'), il faut vérifier v = v' sinon on a un échec.

algorithme : détaillons l'algorithme de filtrer.

```
fonction Filtrer ( t1 , t2 : terme ) : substitut;  
var sigma : substitut;  
var r : V';
```

```
fonction AjoutSiEgal ( x, v, sigma )  
DEBUT  
si accès ( x, sigma ) = x { x n'est pas affectée par sigma }  
alors AjoutSiEgal := ajout ( x, v, sigma )  
sinon si accès ( x, sigma ) = v  
alors AjoutSiEgal := sigma  
sinon AjoutSiEgal := nil;
```

FIN

```
procédure Filtre ( t, u : terme )  
DEBUT { de Filtre }  
r := racine ( t );  
SI r ∈ V  
ALORS sigma ( x ) := AjoutSiEgal ( r, u, sigma );  
SINON SI r = racine ( u )  
ALORS FiltreListe( fils ( t ), fils ( u ) )  
SINON sigma := nil
```

FIN{ de **Filtre** }

```
procédure FiltreListe ( l1, l2 : ListeTerme )  
DEBUT{ de FiltreListe }  
SI non vide ( l1 )  
ALORS Filtre ( tête ( l1 ), tête ( l2 ) ); FiltreListe ( reste ( l1 ), reste ( l2 ) );  
FIN { de FiltreListe }
```

```
DEBUT { de Filtrer }  
sigma := identité ;  
Filtre ( t1 , t2 ) ;  
Filtrer := sigma;  
FIN { de Filtrer }
```

U unifier: [ROBINSON 1965]

Généralisation du filtrage le rendant symétrique

problème : Unifier (t1, t2) : unifier les termes t1 et t2, c'est **générer** une substitution sigma telle que $\sigma (t1) = \sigma (t2) = t$. Ce terme t est dit unifié de t1 et t2.

Si l'unification n'est pas possible, Unifier (t1, t2) fournit la substitution indéfinie nil.

principe : si $t1 \neq t2$
 alors on détermine une disparité (x, v) entre t1 et t2
 si x est une variable
 alors on cherche à unifier subst (t1, x, v) et subst (t2, x, v)
 sinon l'unification est impossible.

algorithme : détaillons l'algorithme d'unification de deux termes.

Unifier:Terme x Terme \rightarrow Substitut est une fonction qui fournit une substitution σ à partir de deux termes t1 et t2 qui les unifie en un même terme t, c'est à dire tel que $\sigma (t1) = \sigma (t2) = t$

fonction **Unifier** (t1 , t2 : terme) : substitut;

var sigma : substitut;

procédure **Sub** (x : V ; v : terme) { substitue v à x dans t1, t2, sigma }

var p : terme;

DEBUT { de **Sub** }

SI non occur (x, v)

ALORS p := sigma (x) ; p↑ := v↑ ;

SINON sigma := nil

FIN { de **Sub** }

procédure **Dispar** (t1, t2 : terme)

var r1 , r2 : V'

DEBUT { de **Dispar** }

r1 := racine (r1) ; r2 := racine (r2) ;

SI r1 ∈ V

ALORS Sub (r1 , t1)

SINON SI r2 ∈ V

ALORS Sub (r2 , t2)

SINON SI r1 ≠ r2

ALORS sigma := nil

SINON DisparListe (fils (t1) , fils (t2))

FIN { de **Dispar** }

procédure **DisparListe** (l1, l2 : ListeTermes)

DEBUT { de **DisparListe** }

SI nonVide(l1) ALORS dispar (tête (l1) , tête (l2)) ; DisparListe (reste (l1) ,
 reste (l2)) ;

FIN { de **DisparListe** }

DEBUT { de **Unifier** }

sigma := identité ;

dispar (Substitution (t1, adaptation) , Substitution (t2, adaptation)) ;

unification := sigma;

FIN { de **Unifier** }

3.3.3. On code la forme comme un arbre

techniques de reconnaissance et de classification de formes :

idée 1 :

on établit des liens de "ressemblance" entre les éléments de la forme. Cette relation de "ressemblance" étant dotée des qualités d'une relation d'équivalence, on génère ainsi des classes de formes correspondant aux concepts à reconnaître.

méthodes et algorithmes :

DA : distance entre deux arbres [LU]

problème : Calculer la distance entre deux arbres

principe de l'algorithme :

On définit trois opérations élémentaires qui modélisent les confusions autorisées :

- 1 • suppression d'un nœud étiqueté "a" n'importe où dans l'arbre de coût $C(a, f)$
- 2 • insertion d'un nœud étiqueté "a" n'importe où dans l'arbre de coût $C(f, a)$
- 3 • substitution d'un nœud étiqueté "a" par "b" de coût $C(a, b)$

La distance entre deux arbres est alors définie comme le coût de la suite d'opérations T la moins coûteuse transformant un arbre en l'autre et vérifiant les contraintes suivantes :

- 1 • $N^a = N^b \Leftrightarrow T(a) = T(b)$
- 2 • la transformation T préserve l'ordre filial
- 3 • la transformation T préserve l'ordre post fixé

Notations : N^a et N^b sont les nœuds d'étiquettes "a" et "b"

L'algorithme utilise la programmation dynamique pour construire un tableau $D(i, j)$ où i et j sont les rangs dans l'ordre post fixé des nœuds des arbres A_1 et A_2 .

Ainsi $D(i, j)$ est la distance entre le sous arbre de A_1 dont la racine a pour rang i et les sous arbres dont la racine a pour rang j

Si n_1 est le nombre de nœuds de A_1 et n_2 est le nombre de nœuds de A_2 alors la complexité est en $O(n_1 \cdot n_2)$

idée 2 :

les classes d'images correspondant aux concepts sont des langages réguliers ou algébriques, sous ensembles particuliers de tous les arbres possibles codant toutes les images.

méthodes et algorithmes :

AS analyse syntaxique: (reconnaissance) []

problème : Reconnaître par automate d'arbre, l'appartenance d'un arbre à un langage

principe de l'algorithme :

L'automate d'arbre partira de l'ensemble des feuilles et cherchera à remonter à la racine en appliquant à l'envers les règles de production de la grammaire associée. Un automate d'arbre est un quadruplet $(X, Q, F, \{f_a / f_a \in X\})$ où X est l'alphabet terminal ; D un ensemble d'états ; F un ensemble d'états finals inclus dans Q ; f_a une application de Q^n dans Q avec $n \in \mathbb{N}$.

Une telle machine a un fonctionnement parallèle sur l'arbre à analyser, que l'on peut décrire brièvement de la façon suivante : L'automate affecte un état initial (état particulier de Q) à chaque feuille de l'arbre. Tous les chemins vers la racine sont simultanément explorés, par l'examen d'un nœud et de ses successeurs. Si l'ensemble des états déjà affectés aux successeurs d'un nœud R d'étiquette a est un élément de Q^n dans l'application f_a , on affectera l'état correspondant à cette application à l'état R . Le procédé s'arrête soit par impossibilité d'étiquette de nœud (échec) ; soit quand la racine est étiquetée avec un état de $Q - F$ (échec) ; soit avec un état de F (réussite : l'arbre peut être engendré par la grammaire associée à l'automate).

IG : inférence de grammaire d'arbre: (apprentissage)[LEVINE]

problème : Chercher à détecter des régularités dans un échantillon d'apprentissage, dans une perspective d'apprentissage.

principe de l'algorithme:

Les arbres sont considérés ici comme un *domaine d'arbre* D , c'est à dire une liste de mots sur N^* qui décrit la structure seule associée à un étiquetage ; un arbre est donc une application t de D dans l'ensemble A des étiquettes. On notera $D = \text{dom}(t)$.

L'arbre t peut être représenté par $t = \{(x, a) / x \in \text{dom}(t) \text{ et } a = t(x)\}$. On notera $b/a = b' \Leftrightarrow b = a \cdot b'$ pour $a, b, b' \in N^*$. Le sous arbre de t en y où $y \in \text{dom}(t)$ est défini par $t/y = \{(x, b) / (y \cdot x, b) \in t\}$

Le remplacement du sous arbre de t en y par un arbre u fournit l'arbre t' défini par : $t' = t(y \leftarrow u) = \{(x, b) \in t / t(x) = b \text{ et } y \leq x\} \cup \{(y \cdot z, u(z)) / z \in \text{dom}(u)\}$ avec la convention $y \cdot 0 = y$.

On appelle dérivée d'ordre m d'un ensemble S d'arbres, relativement à l'arbre t , l'ensemble noté $D_t^m(S)$ défini récursivement par :

$$D_t^1(S) = \{u(b \leftarrow \$) / u \in S \text{ et } u/b = t\}$$

$$\text{et pour tout } i > 1 \text{ on ait : } D_t^{i+1}(S) = D_t^1(D_t^i(S))$$

"\$" est un caractère particulier de l'ensemble A des étiquettes.

On a la propriété : $D_t^1(S) = D_u^1(S) \Rightarrow$ pour tout j , on a $D_t^j(S) = D_u^j(S)$

Théorème 1 : Soit M un automate d'arbre et $T(M)$ l'ensemble des arbres qu'il accepte.

les trois propositions suivantes sont équivalentes :

1. S est un ensemble d'arbres acceptés par M
2. S est l'union des classes d'une relation d'équivalence invariante d'index fini.
3. La relation d'équivalence R^M définie par :
 $[t R^M u] \Leftrightarrow [\text{pour tout arbre } v \text{ et tout } x \text{ de } \text{dom}(v) \text{ on } a(v(x \leftarrow t) \in S) \Leftrightarrow (v(x \leftarrow u) \in S)]$

est une relation d'équivalence d'index fini.

Théorème 2 : Soit M un automate d'arbre ; $T(M)$ l'ensemble des arbres qu'il accepte et R^M la relation d'équivalence définie dans le théorème 1.

Pour tout $j \geq 1$ on a : $[t R^M u] \Leftrightarrow D_t^j(T(M)) = D_u^j(T(M))$

On définit la dérivée par rapport à t , d'ordre m bornée à la profondeur K d'un ensemble S d'arbres, par : $D_t^{K,m}(S) = \{u / u \in D_t^m(S) \text{ avec niveau}(u) \leq K\}$

On a la propriété : pour tout $K \geq 0$ et tout $i \geq 1$ on a $[t R^M u] \Leftrightarrow D_t^{K,i}(T(M)) = D_u^{K,i}(T(M))$

L'algorithme ci-dessous utilise cette propriété : on construit l'automate canonique, et on confond éventuellement certains de ses états, de façon à réaliser une généralisation du langage qu'il accepte. La confusion des états se fait à l'examen des dérivées bornées des ensembles d'arbres correspondant aux états.

algorithme de LEVINE

1. Soit S l'échantillon d'apprentissage.
2. Soit R^M la relation d'équivalence associée à l'automate d'arbres canonique, c'est à dire qui possède un état pour chaque sous arbre de S .
3. Pour toutes les paires de sous arbres de S (t_e, t_j), n'appartenant pas à la même classe faire :

si $D_{t_e}^{K,i}(S) = D_{t_j}^{K,i}(S)$ et $D_{t_e}^{K,i}(S) \neq \emptyset$ et $D_{t_j}^{K,i}(S) \neq \{\$ \}$

alors confondre les états de l'automate correspondant aux classes contenant t_e et t_j .

4. Résultat : une relation d'équivalence $R^{M'}$ moins fine que R^M correspondant à un automate d'arbre M' reconnaissant un langage d'arbre contenant S , donc solution du problème d'apprentissage posé.

3.3.4. On code la forme comme un graphe.

Remarque : plus un outil est général et moins sa manipulation dans un domaine particulier est rapide.

De plus les graphes sont beaucoup plus utilisés comme des outils de description que comme des modèles permettant réellement un apprentissage ou une décision.

DA : distance entre deux graphes

problème : Calculer une distance entre deux graphes.

On peut imaginer définir une distance entre graphes, fondée sur la manière la moins "chère" de passer d'un graphe à un autre, comme nous l'avons vu pour les chaînes ou les arbres. Il est vraisemblable que la complexité d'un tel algorithme est exponentielle en fonction de la taille des graphes. SANTELIU et Fu [SANF 83] traitent de ce sujet pour un type particulier de graphe.

MG: "matching" entre deux graphes

problème : Mettre en correspondance des graphes (matching).

Les solutions d'un tel problème n'apparaissent dans la littérature que dans des cas particuliers.

Les définitions du "graph matching" y sont nombreuses et correspondent chacune plus ou moins au type d'application les mettant en œuvre.

Pour illustrer ces différentes approches, nous allons en faire une revue rapide :

D.H. BALLARD et C.H. BROWN [BALL 82] présentent cinq définitions

$G_1 = (S_1, A_1)$ et $G_2 = (S_2, A_2)$ sont deux graphes dont les sommets sont S et les arêtes A .

Définition 1 : Isomorphisme de graphe

Trouver une bijection f entre S_1 et S_2 telle que :

pour tout s_1 de S_1 et tout s_2 de S_2 tels que : $f(s_1) = s_2$ on ait pour tout s'_1 de S_1 :
si (s_1, s'_1) est dans A_1 alors $(s_2, f(s'_1))$ est dans A_2

Définition 2 : isomorphisme de sous graphe

Trouver les bijections f reliant G_1 aux sous graphes de G_2

Définition 3 : isomorphisme de sous graphes "doubles"

Trouver les bijections f reliant les sous graphes de G_1 aux sous graphes de G_2

Définition 4 : mise en correspondance de sous graphes avec omission

Dans cette mise en correspondance certains sommets ou arêtes jugés non importants peuvent être omis.

Définition 5 : mise en correspondance de sous graphes en cherchant une similarité.

Dans cette mise en correspondance un graphe peut être plus "riche" qu'un autre.

Z. GALIL [GALI 86] présente quatre problèmes :

$G = (S, A)$ est un graphe non orienté. Les sommets S sont des personnes, les arêtes A des mariages. Le but est de sélectionner certaines de ces arêtes. Les quatre problèmes excluent la polygamie.

Problème 1 : Cardinalité maximum de la mise en correspondance dans un graphe bipartite
Les sommets S représentent des hommes ou des femmes, il ne peut exister d'arête (mariage) qu'entre un homme et une femme. On cherche à faire le plus de mariages possibles.

Problème 2: Cardinalité maximum de la mise en correspondance dans un graphe quelconque
Les sommets S représentent des personnes "asexuées", il n'y a toujours pas de polygamie. On cherche à faire le plus de mariages possibles.

Problème 3 : Poids maximum de la mise en correspondance dans un graphe bipartite
Les sommets S représentent des hommes ou des femmes, chaque arête (mariage) entre un homme et une femme rapporte une valeur (poids de l'arête). On cherche à maximaliser le bénéfice et pas nécessairement le nombre de mariages.

Problème 4 : Poids maximum de la mise en correspondance dans un graphe quelconque
Les sommets S sont asexués, chaque arête rapporte un bénéfice, on cherche à maximaliser le gain obtenu par les poids des arêtes sélectionnées.

Intéressons nous maintenant aux méthodes de la mise en correspondance :

Quatre classes de méthodes apparaissent

Méthode 1 : Méthode métrique de mise en correspondance

Idée : Elle repose sur une analogie avec le modèle physique des "gabarits et ressorts" [BALL 82] Chaque gabarit est un sommet du graphe, chaque ressort est un arc.
+ La tension du ressort est proportionnelle aux relations entre les sommets (une déformation du graphe plus ou moins facile se traduit par une tension plus ou moins grande)
+ Le type du gabarit caractérise la nature du sommet. Deux gabarits sommets seront ainsi plus ou moins similaires.
On met en place une fonction de coût de la mise en correspondance, du coût en "énergie" nécessaire pour effectuer la déformation d'une graphe vers l'autre.

avantages : simple à formuler.

inconvénients : mise en œuvre problématique, comme beaucoup de méthodes fondées sur des modèles physiques de la déformation.

Méthode 2 : Méthode de recherche par retour arrière

Idée : Elle repose sur une recherche exhaustive de solutions de manière étagée.[BALL82]

avantages : facile à mettre en œuvre.

inconvénients : ne résout que les problèmes d'isomorphisme strict.

Méthode 3 : Méthode utilisant un graphe de mise en correspondance ("association graph")

Idée : Elle repose sur la construction d'un graphe de mise en correspondance

Un sommet du graphe "association graph" correspond à un appariement entre un sommet du premier graphe et un sommet du second.

Une arête correspond à la compatibilité de l'appariement.

Il suffit alors de rechercher une clique dans ce graphe "association graph".

avantages : méthode permettant la polygamie si on le souhaite

inconvénients : le problème est NP complet..

Méthode 4 : Méthode spécifique aux graphes planaires

avantages : méthode polynomiale [BALL 82]

3.4 . Méthodes analytiques :

Une image peut être définie par des courbes ou des a plats définis par des équations sur les coordonnées des pixels de l'image. Nous allons nous placer alors dans le cadre de l'outil analytique.

Les techniques analytiques apparaissent :

- soit en synthèse d'image, pour "approcher" des courbes à l'aide de polygones ou de courbes simples,
- soit en analyse d'image, pour représenter un ensemble de pixels par une équation facilitant ainsi son identification par comparaison par exemple dans une base de données.

Nous allons nous intéresser tout d'abord aux techniques d'approximation par une ligne polygonale ou une courbe simple d'un ensemble de pixels dans une perspective de reconnaissance, c'est à dire l'extraction d'une description sous forme d'une suite de segments de taille variable. C'est une phase dite de vectorisation.

Nous allons nous intéresser ensuite aux techniques d'approximation par B spline.

3.4.1. l'approximation par une ligne polygonale ou une courbe.

Problème :

Étant donné un ensemble F de n pixels du plan représentant une figure, trouver une ligne polygonale P approchant au mieux F , tout en ayant un nombre minimum de sommets. autrement dit trouver un recouvrement minimum approximant au mieux F , par des segments.

Le problème de trouver une courbe C approximant au mieux F , est tout à fait semblable.

Modélisation :

Ici encore, nous allons mettre en place une fonction de coût, qui évalue l'erreur entre l'objectif F et l'approximation P proposée.

Notons err_i l'erreur calculée au i ème point de E :

fonction de coût maximal : $Coût (F, P) = \text{Max}_i | err_i |$;

fonction de coût aux moindres carrés : $Coût (F, P) = \sum_i (err_i)^2$

Notons C une courbe définie par une équation $y = g(x)$ où $g (x) = \sum_{j=0}^m a_j \cdot b_j (x)$ où les fonctions b_j forment une base de l'espace vectoriel de dimension finie m des applications g caractéristiques des courbes observées.

Notons $F = \bigcup_{i=0}^m \{ (x_i, y_i) \}$

Approximation au sens des moindres carrés :

La fonction de coût s'écrit :
$$\text{Coût} (F, P) = \sum_{i=0}^n \left[y_i - \sum_{j=0}^m [a_j \cdot b_j (x_i)] \right]^2$$

Notons B_X la matrice définie de la façon suivante :

$$B_X = \begin{bmatrix} b_0(x_0) & b_1(x_0) & \dots & b_m(x_0) \\ b_0(x_1) & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ b_0(x_n) & \dots & \dots & b_m(x_n) \end{bmatrix}$$

Ce que l'on cherche :

il s'agit de chercher le vecteur A, combinaison linéaire $\sum a_j \cdot b_j (x_k)$, la plus proche de y_k pour chacun des entiers k de l'intervalle [0, n] tel que : $\inf_{A \in \mathbb{R}^m} \| Y - B_X \cdot A \| = \| Y - B_X \cdot A \| = E_{mc}$

Existence et unicité de la solution A cherchée:

Comme $\text{im} (B_X) = \{ U \in \mathbb{R}^m \text{ tels que pour tout } A \in \mathbb{R}^m \text{ on ait } U = B_X \cdot A \}$ est un sous espace vectoriel complet de \mathbb{R}^m , et comme les b_j forment une famille libre, on peut affirmer d'après le théorème d'unicité de la projection que quel que soit le vecteur Y de \mathbb{R}^m il existe un unique vecteur $U = B_X \cdot A$ de $\text{im} (B_X)$ tel que ce vecteur U soit la projection de Y sur $\text{im} (B_X)$

Cette situation idéale n'est pas toujours réalisé, aussi sommes-nous amenés à mettre en œuvre des solutions approchées.

Méthode approchée :

La recherche de la meilleure valeur au sens des moindres carrés est généralement un problème gourmand en calculs. Nous mettons alors en œuvre des heuristiques approchées qui souvent sont beaucoup plus rapides que les algorithmes qui fournissent la meilleure solution.

En particulier dans le domaine de la robotique, les applications utilisent des algorithmes linéaires ou *quasi linéaires*.

Nous allons présenter une heuristique d'approximation polygonale souvent employée en robotique, appelée "*split-and-merge* ", c'est à dire "découper et fusionner".

Cette heuristique vérifie la colinéarité d'une suite de points :

S'ils ne sont pas colinéaires alors la suite est séparée, découpée jusqu'à obtention de la condition de colinéarité.

Dans le cas contraire, certaines suites sont fusionnées si leur concaténation vérifie le critère de colinéarité.

Le critère de colinéarité approximative est simple : il correspond au tracé d'une corde entre deux points et à la vérification que tous les points entre ces deux extrêmes ne s'écartent pas plus de la corde d'une distance D fixée.

[PAVL82] indique l'intérêt de cette approximation, et présente la propriété suivante que possèdent les suites de segments obtenus :

Proposition :

Le nombre de segments obtenus par un algorithme du type "découper et fusionner" est toujours inférieur à deux fois le nombre minimum de segments. [PAVL 82].

Preuve :

Soient I_1, I_2, \dots, I_n les intervalles de la partition minimale et J_1, J_2, \dots, J_p ceux trouvés par l'algorithme "découper et fusionner". Aucun intervalle I_i ne peut contenir plus d'un intervalle J_k car dans le cas contraire, J_k et J_{k+1} auraient forcément fusionnés. Par contre, chaque I_j , peut contenir à l'intérieur de ces bornes l'intervalle J_k et une partie des intervalles J_{k-1} et J_{k+1} , recouvrant aussi I_{j-1} et I_{j+1} respectivement.

On a donc au plus n intervalles J_k à l'intérieur de chacun des intervalles I_j de la partition minimale, ainsi qu'autant d'intervalles J_k qu'il n'y a de points de séparation dans cette partition minimale. C'est à dire : $n + n - 1$. D'où $p \leq 2n - 1$.

Les algorithmes de l'approximation polygonale :

Ces algorithmes possèdent plusieurs critères d'optimisation difficiles à satisfaire complètement :

- 1 • le maximum d'erreur obtenue sur chacun des segments
- 2 • le nombre maximal de segments que possède la description
- 3 • le périmètre du polygone minimal
- 4 • un temps d'exécution raisonnable

Nous présentons ci-dessous un algorithme de **découpe** [RAME72] considéré par [BANG86] comme l'un des plus utilisés en robotique :

- 1• Choisir n points ($n \geq 2$) qui forment un polygone de départ
- 2• **Pour** chaque segment du polygone courant
faire Chercher le point image P de distance maximale d_{max}
Si $d_{max} > \text{seuil}$
alors Diviser le côté $[P_n, P_{n+1}]$ considéré en deux au niveau du point $P : [P_n, P]$
et $[P, P_{n+1}]$
is
fait

Cet algorithme se retrouve dans diverses applications robotiques [DELO88]. Les modifications apportées concernant surtout la poursuite de l'approximation d'une corde tant que les points restent alignés sur cette corde. Cependant, les points choisis au départ ne sont pas forcément les points de cassure obtenus à la fin de l'algorithme.

Nous présentons ci-dessous un algorithme [LUX84] qui permet d'obtenir une liste finale simple de directions de droites pour le système de vision PVV.

- 1• Rechercher n points alignés à un seuil d'erreur "seuil1" près
 - Prendre n points image successifs de la liste ordonnée des éléments du contour P_i, P_i
 $+n$
 - Tracer la corde $[P_i, P_{i+n}]$
 - Chercher P_{max} tel que $d_{max} = d(P_{max}, \text{corde})$
 - **Si** $d_{max} > \text{seuil1}$ **alors** $P_i = P_{max}$; **aller en 1• is**

- 2• Rechercher des points suivants alignés à un seuil d'erreur "seuil2" près
 - Construire une bande d'épaisseur "seuil2" parallèle au segment issu des n points précédents
 - Les points suivants P_{i+n+1} , P_k placés à l'intérieur de cette bande appartiennent au même segment de droite
 - $P_i = P_k$
 - **aller en 1•**

3.4.2. l'approximation par B spline.

Problème :

Étant donné un ensemble F de n pixels du plan représentant une figure, trouver une ligne courbe C approchant au mieux F. Cette courbe étant définie par un polygone P, sorte d'enveloppe dans laquelle la courbe est contrainte à passer. C'est à dire que l'on cherche à approcher un ensemble F par une courbe C définie à partir d'un ensemble de points dits points de contrôle, formant un polygone, dit polygone de contrôle. Les sommets de ce polygone, ne sont d'ailleurs pas obligatoirement des points de la courbe à approximer.

Modélisation :

Définition 1 :

Soient $k+1$ réels $x_0, x_1, \dots, x_j, \dots, x_{k-1}, x_k$ de l'intervalle $[a, b]$ le subdivisant en k sous intervalles appelés points de cassure tels que $x_0 = a$ et $x_k = b$

La donnée d'une fonction polynomiale par morceaux est la donnée de

(1) pour tout i de 0 à $k-1$ on a $p(x) = p_i(x)$

(2) pour tout j de 0 à $r-1$ et pour tout i de 1 à $k-1$ on a $p_i^{(j)}(x_i) = p_{i+1}^{(j)}(x_i)$

p_i est un polynôme de degré m ou moins.




Définition 2 :

Une fonction spline simple est une fonction polynomiale par morceaux décrite par les équations (1) et (2) avec $r = m$.

Pour $n=1$, on parle de spline linéaire. Pour $n=2$, on parle de spline quadratique. Pour $n=3$, on parle de spline cubique.

La courbe de degré n est entièrement contenue dans le polygone convexe formé des suites de $n+1$ points consécutifs.

Voici quelques illustrations pour les premières valeurs de n :

		
n=1 : linéaire	n=2 quadratique	n=3 cubique

La courbe ou la surface sera approchée par une courbe ou une surface polynomiale contrôlée par des points ou des vecteurs via les polynômes de BERNSTEIN.

Définition 3 :

Un polynôme de BERNSTEIN est un polynôme décrit par deux paramètres i et n dans l'expression :

$$B_i^n(t) = C_i^n (1-t)^{n-i} t^i$$

Les polynômes de BERNSTEIN possèdent un certain nombre de propriétés assez remarquables.

- pour tout t de $[0 ; 1]$ on a $B_i^n(t) \geq 0$
- pour tout t de $[0 ; 1]$ on a $B_i^n(1-t) = B_{n-i}^n(t)$
- pour tout t de $[0 ; 1]$ on a $\sum_{i=0}^n B_i^n(t) = 1$
- pour tout t de $[0 ; 1]$ on a $B_i^n(t) = (1-t) \cdot B_i^{n-1}(t) + t \cdot B_{i-1}^{n-1}(t)$: formule de récurrence.
- pour tout t de $[0 ; 1]$ on a $\frac{d B_i^n(t)}{d t} = n \cdot (B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$

Notation : On note $BP [P_0, \dots, P_n] (t) = \sum_{i=0}^n P_i \cdot B_i^n(t)$

Algorithme de DE CASTELJAU de calcul de $BP [P_0, \dots, P_n] (t)$:

{ initialisation de la première colonne du triangle des coefficients P_j^i }

pour j de 0 à n faire : $P_j^0 \leftarrow P_j$

{ calcul du reste du triangle }

pour i de 1 à n faire

pour j de 0 à $n - 1$ faire $P_j^i \leftarrow (1-t) \cdot P_j^{i-1} + t \cdot P_{j+1}^{i-1}$

résultat : $BP [P_0, \dots, P_n] (t) = P_0^n$

Cet algorithme est assez remarquable car il est facilement parallélisable, il ne fait intervenir que des calculs "simples".

FIOROT et JEANNIN, ont proposé un contrôle par système de points pondérés et vecteurs qui permet de caractériser simplement, non plus des courbes polynomiales, mais des courbes rationnelles.

3.5 . Méthodes connexionnistes :

3.5.1 Description d'un système connexionniste

On regroupe sous le nom de connexionnisme, les techniques d'utilisation de réseaux d'automates simples, interconnectés, dans lesquels la connaissance est distribuée dans les connexions, capables d'adapter leurs comportements en fonction de leur expérience acquise, capables de résister aux perturbations.

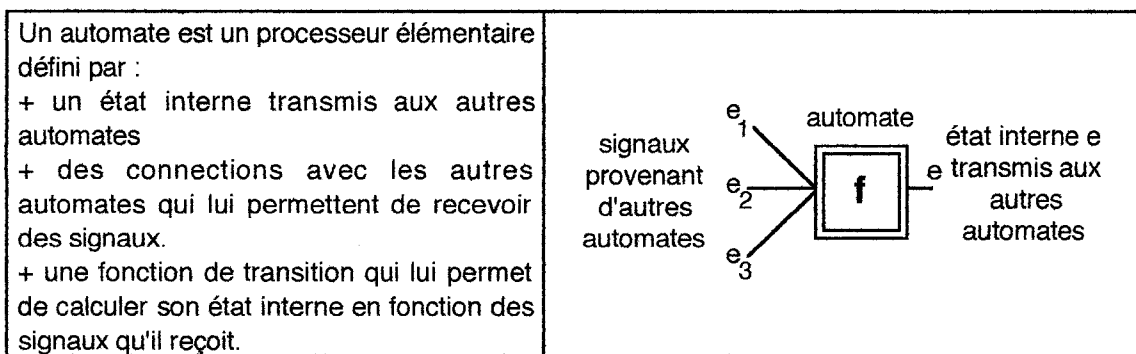
Un système connexionniste est ainsi un système compliqué formé de nombreux éléments en interaction. Ces éléments sont appelés des unités. Ils ont un comportement individuel en général très simple, ils s'influencent mutuellement selon leurs affinités. Le système admet un comportement global souvent d'une grande richesse qui s'interprète en termes de propriétés émergentes.

Un système connexionniste est ainsi capable d'une grande variété de comportements collectifs s'adaptant en fonction de l'expérience acquise (apprentissage), résistant aux perturbations (robustesse), restituant une information même incomplète (mémorisation).

Comme dans les méthodes différentielles, on part d'une description locale du système, en termes de changements à court terme des états des unités sous l'effet de leurs influences mutuelles. On attend du formalisme, la description globale du système, de son comportement en termes de propriétés émergentes.

3.5.1.1. Les unités constituant un système connexionniste

3.5.1.1.1 L'AUTOMATE

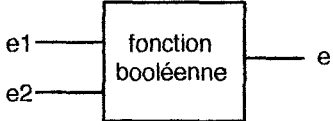
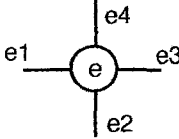


Historiquement deux types de tels automates sont apparus à peu près à la même époque :

- + L'automate booléen conçu vers 1940 par VON NEUMANN, et
- + Le neurone formel conçu dès 1943 par MAC CULLOCH et PITTS.

3.5.1.1.2 L'UNITÉ BOOLÉENNE DE VON NEUMANN.

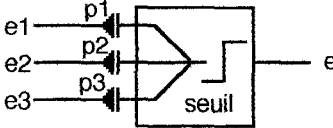
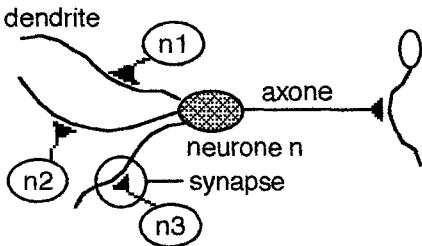
C'est un automate imaginé en vue de simuler des systèmes auto reproducteurs. Il s'agissait pour VON NEUMANN de répondre à la question : "Est-il possible de construire un système artificiel capable de se reproduire ?"

Unité booléenne	Modélisation d'un nœud du maillage
	
<p>L'état e est calculé par $f(e_1, e_2)$ où f est une fonction booléenne, c'est à dire que les entrées e_1 et e_2, ne peuvent prendre que deux valeurs, ainsi que le résultat e.</p>	<p>L'unité reçoit les influences e_1, e_2, e_3, e_4, de ses quatre voisins et retourne sa réaction e à leur sollicitations. L'état de chaque nœud du maillage évolue en fonction des états de ses voisins.</p>

Le nombre des fonctions booléennes possibles est fonction du nombre k des entrées : il est égal à 2^k . Chacune de ces fonctions booléennes peut être définie par sa table de vérité et codée sur un code à k bits.

3.5.1.1.3 LE NEURONE FORMEL DE MAC CULLOCH ET PITTS.

C'est un automate inspiré du fonctionnement d'un neurone réel.

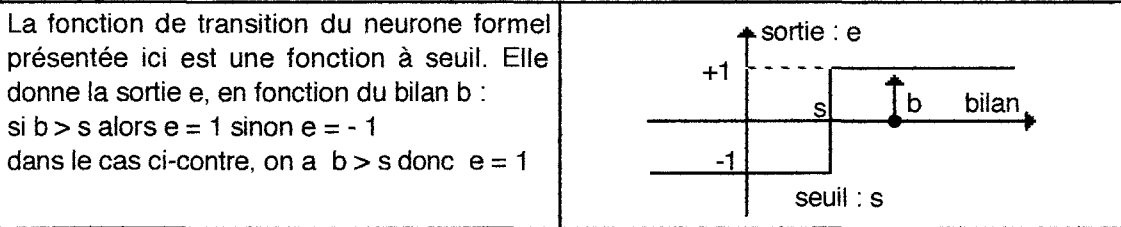
Unité : Neurone formel	Cellule : Neurone réel
 <p>si $e_1 \cdot p_1 + e_2 \cdot p_2 + e_3 \cdot p_3 > s$ alors $e = 1$ sinon $e = -1$</p>	
<p>p_1, p_2, p_3 sont des nombres réels qui représentent l'efficacité synaptique du contact entre unités. + si le nombre est positif alors l'efficacité synaptique est activante, d'autant plus que le nombre est grand. + Si le nombre est négatif alors l'efficacité synaptique est inhibante d'autant plus que le nombre est grand. + Si le nombre est nul alors l'unité est non connectée. L'unité neurone formel réalise la somme des produits des états des unités en entrée par les poids synaptiques (ici : $e_1 p_1 + e_2 p_2 + e_3 p_3$). Si ce total donne un bilan positif alors l'unité passe à l'état actif (+1) sinon en état passif (-1). Un bilan positif est un bilan qui dépasse le seuil S</p>	<p>Différents neurones sont connectés au neurone n, au moyen de synapses. Ces synapses ont une efficacité variable. Par exemple le neurone n_1 est activant du neurone n, si son efficacité synaptique est positive ; il sera inhibant du neurone n, si son efficacité synaptique est négative. Le neurone intègre les effets activants et inhibants pour en faire un bilan en fonction duquel il se mettra en état éveillé ou endormi, pour transmettre ou non un signal aux autres neurones auxquels il est connecté.</p>

3.5.1.1.4 LES FONCTIONS DE TRANSITION DES AUTOMATES.

Elles peuvent prendre différentes formes :

A. Modèles à seuil :

1. Les Fonctions à seuil :

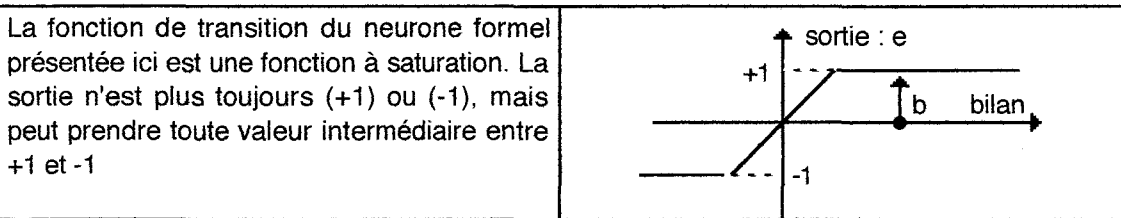


Des neurones formels ne pouvant prendre que l'état (1) ou l'état (-1) sont dits des neurones binaires.

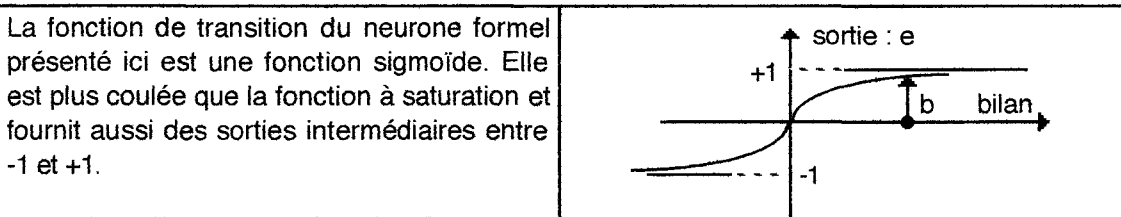
Les états d'un neurone binaire peuvent aussi être codés (1) et (0).

B. Modèles continus :

2. Les Fonctions à saturation :



3. Les Fonctions sigmoïdes :



Des neurones pouvant prendre toute valeur intermédiaire entre -1 et 1 sont dits analogiques ou continus.

C. Modèle linéaire :

4. La Fonction identique :

La sortie est tout simplement le bilan des entrées. La fonction de transition n'apporte aucune distorsion.

D. Modèle stochastique :

5. Fonction probabiliste :

La sortie n'est pas déterminée par le bilan des entrées de manière sure. On évalue la probabilité pour que la sortie soit à 1 par une formule du type : $P(e=1) = \frac{1}{1 + e^{-\frac{b}{t}}}$ où b est le

bilan des entrées et t une température.

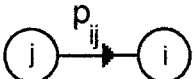
3.5.1.2 Description d'un système connexionniste

Les unités :

Elles sont repérées par un numéro de 1 à N. On note i, j, k un numéro particulier	
--	--

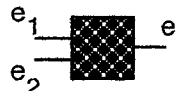
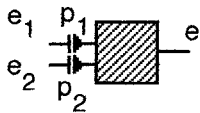
Les connexions :

Elles sont les supports des influences d'une unité sur une autre. Cette influence est notée : p_{ij} . Cette influence peut être variable ou fixe (elle est alors égale à 1 (influence) ou 0 (non influence)). On représente par un tableau carré N sur N ces N^2 influences

Ce p_{ij} est l'influence que fait subir l'unité source numéro i sur l'unité but numéro j	
---	--

Comportement individuel d'une unité :

Il est décrit par la fonction f_i de transition de l'unité numéro i .

Pour une unité booléenne : $e = f(e_1, e_2)$ 	Pour une unité de type neurone : $e = f(p_1 \cdot e_1 + p_2 \cdot e_2)$ 
--	--

Agencement des unités du réseau, dynamique de leur activation:

Il peut être à influence unidirectionnelle. C'est une organisation en couches qui correspond à une forme de raisonnement déductive, à une perception sans contexte, sans dépendance avec ce qui a pu précéder. La propagation des influences est unidirectionnelle, il y a déduction de la réponse à partir de la stimulation selon des interactions microscopiques entre les individus reliés.

Il peut être à influence bidirectionnelle. C'est une organisation à circuits qui correspond à une forme de raisonnement plus élaborée. Une "opinion" affichée présente un caractère d'hypothèse confirmée ou infirmée par les circuits de retour, réinjectée après révision éventuelle. Il y a convergence vers des associations cohérentes entre les entrées et les sorties. L'état interne du système est une sorte de mémoire du passé proche, du contexte. La mémoire à court terme est codée dans le niveau d'activation des unités cachées, la mémoire à long terme est codée dans les poids des influences, des connexions entre les unités. L'évolution du réseau est décrite par des trajectoires dans l'espace de ses états.

Fiche de description d'un système connexionniste :

Nom : [nom donné au système]

Architecture :

UNITES :

[Type d'unité]	[rôle du type d'unité]
état interne Xi:	[discret (-1 ou 1) (0 ou 1), continu, ...]
fonctions de transition fi :	[booléennes ou semi linéaires à seuil, ...]
nombre d'unités :	[noté : N]

CONNEXIONS :

influences :	[influences symétriques ou non , unidirectionnelle ou non]
agencement :	[connexions aléatoires ou cellulaires]

Dynamique :

EN EXPLOITATION :	[dynamique d'évolution du réseau]
-------------------	-------------------------------------

EN APPRENTISSAGE	[dynamique d'adaptation du réseau]
------------------	--------------------------------------

Inspiration : [modèle physique inspirant le système]

Notation : [notations utilisées dans la description]

Indicateur(s) associé(s) à un état du système :

Exemple de réseau d'unités : "neurones formels "

Considérons six unités, binaires à seuil nul, à la fois en entrée et en sortie, connectés selon le schéma ci-dessous :

La matrice des influences est :

1	2	0	0	0	-2	0
2	0	2	2	-2	0	-2
3	0	2	2	-2	0	-2
4	0	-2	-2	2	0	2
5	-2	0	0	0	2	0
6	0	-2	-2	2	0	2

n° des unités ↑
→ 1 2 3 4 5 6

A l'intersection de la ligne et de la colonne repérée par un numéro d'unité, on trouve le coefficient synaptique du lien entre ces deux unités.

On remarque dans ce cas particulier l'égalité des coefficients dans un sens et dans l'autre pour tous les couples d'unités.

Le schéma montre que ce réseau est formé de deux réseaux disjoints.

Réorganisons les lignes et les colonnes de la matrice pour faire apparaître cette division. La matrice ci-contre est clairement formée de deux sous matrices complètes simples (on dit plutôt blocs), l'une de 2 lignes sur 2 colonnes et l'autre de 4 lignes sur 4 colonnes.

La matrice des influences devient :

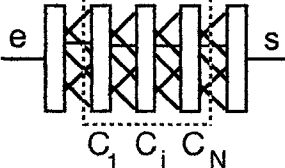
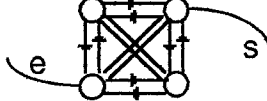
1	2	-2	0	0	0	0
5	-2	2	0	0	0	0
2	0	0	2	2	-2	-2
3	0	0	2	2	-2	-2
4	0	0	-2	-2	2	2
6	0	0	-2	-2	2	2

1 5 2 3 4 6

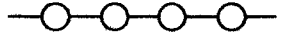
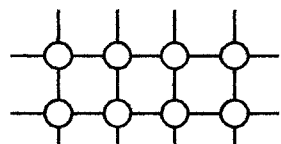
3.5.1.3 Choix possibles pour des systèmes connexionnistes

Pour construire un système connexionniste, on opère un certain nombre de choix architecturaux :

A. réseaux à fonction de transition non booléenne

	réseau à influence unidirectionnelle	réseau à influence bidirectionnelle
agencement	en couches	homogène à circuits
forme de raisonnement	déductive : la réponse est déduction du motif de stimulation.	convergente vers des associations cohérentes entre son entrée et sa sortie. Une "opinion" affichée en entrée présente un caractère d'hypothèse confirmée ou infirmée par les circuits en retour.
perception	sans contexte	avec contexte
états internes	sans influence sur tout nouveau motif présenté.	influence sous forme de mémoire du passé proche. Le niveau d'activation des unités cachées intervient pour la future association cohérente
mémoire à court terme	néant	par le niveau d'activation des unités cachées.
mémoire à long terme	dans le poids des connexions, des influences	dans le poids des connexions, des influences
schéma		

B. réseaux à fonction de transition booléenne

	réseau cellulaire	réseau aléatoire
agencement	Les unités sont les nœuds d'un maillage linéaire (trois entrées) ou plan (une ou huit entrées)	Les unités sont les nœuds d'un maillage linéaire (trois entrées) ou plan (une ou huit entrées)
dynamique	Chaque unité prend l'état conséquence des influences logiques de ses voisins	Chaque unité prend l'état conséquence des influences logiques de ses voisins par une fonction de transition déterminée aléatoirement
schéma	linéaire :  plan : 	aléatoire

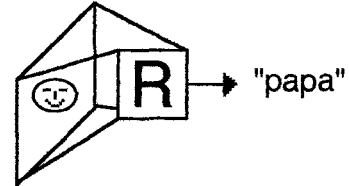
3.5.2. Ingénierie des systèmes connexionnistes

L'attitude de l'ingénieur est d'imaginer des procédés permettant de construire des systèmes réalisant un comportement donné ou le mieux adapté à une tâche donnée.

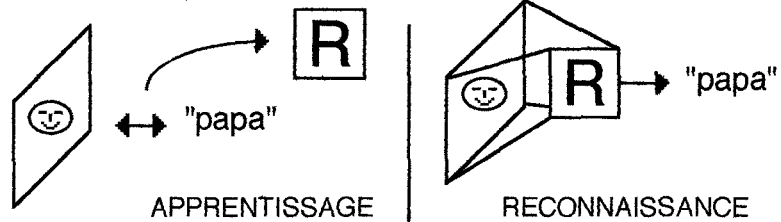
Question :

Comment construire un système sous forme de réseau connexionniste ?

+ Capable de résoudre certains problèmes déterminés :



+ Capable d'apprendre à résoudre certains problèmes déterminés :



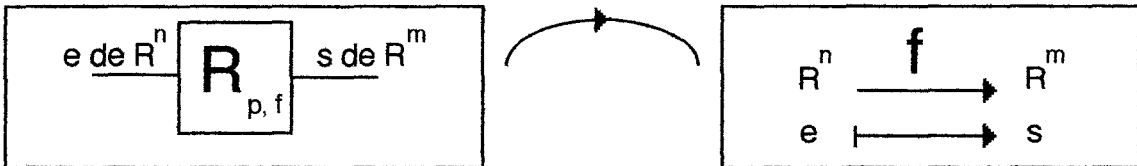
+ Deux questions réciproques l'une de l'autre se posent alors :

Question 1 :

Étant donné un système connexionniste $R_{p, f}$:

Quelle classe F de fonctions peut-il calculer ?

C'est réaliser la transformation décrite par le schéma ci-dessous :



On peut alors identifier trois sous questions relatives à cette question :

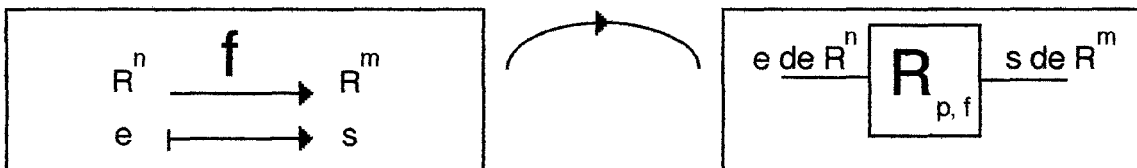
- 1.1 Quand p varie, quelle est la classe $F(R_{p, f})$?
- 1.2 Quand f varie, quelle est la classe $F(R_{p, f})$?
- 1.3 Quelle est la puissance de calcul d'une architecture de réseau ?

Question 2 :

Étant donnée une fonction f

Quelle classe de systèmes connexionnistes $CR_{p, f}$ réalise cette fonction ?

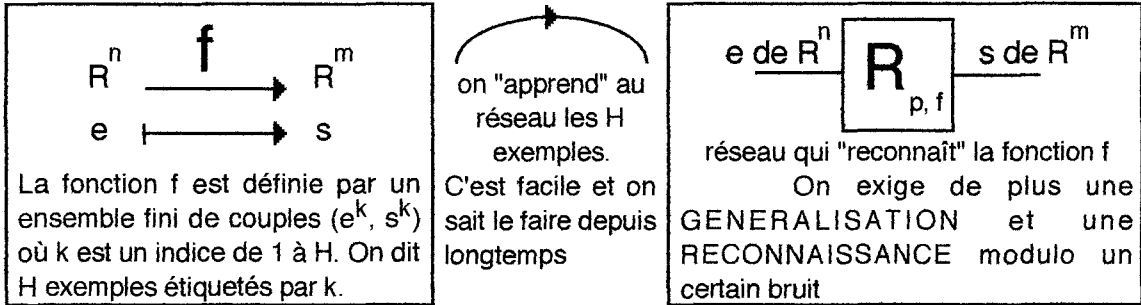
C'est réaliser la transformation décrite par le schéma ci-dessous :



Stratégie de détermination :

On se donne une architecture à priori et on modifie les influences p par apprentissage pour que f fasse partie de la classe du réseau.

Spécification du problème :



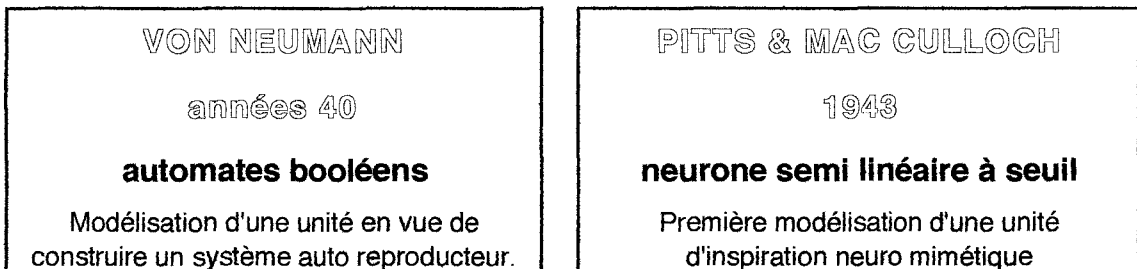
Question 3

Étant donnée une certaine "connaissance", quel est le "degré" de mémorisation, de délocalisation de cette connaissance ? En particulier peut-il y avoir mémorisation sans représentation ?

3.5.3 Panorama des systèmes connexionnistes :

Les bases ancestrales et les perspectives

3.5.3.1 Les racines

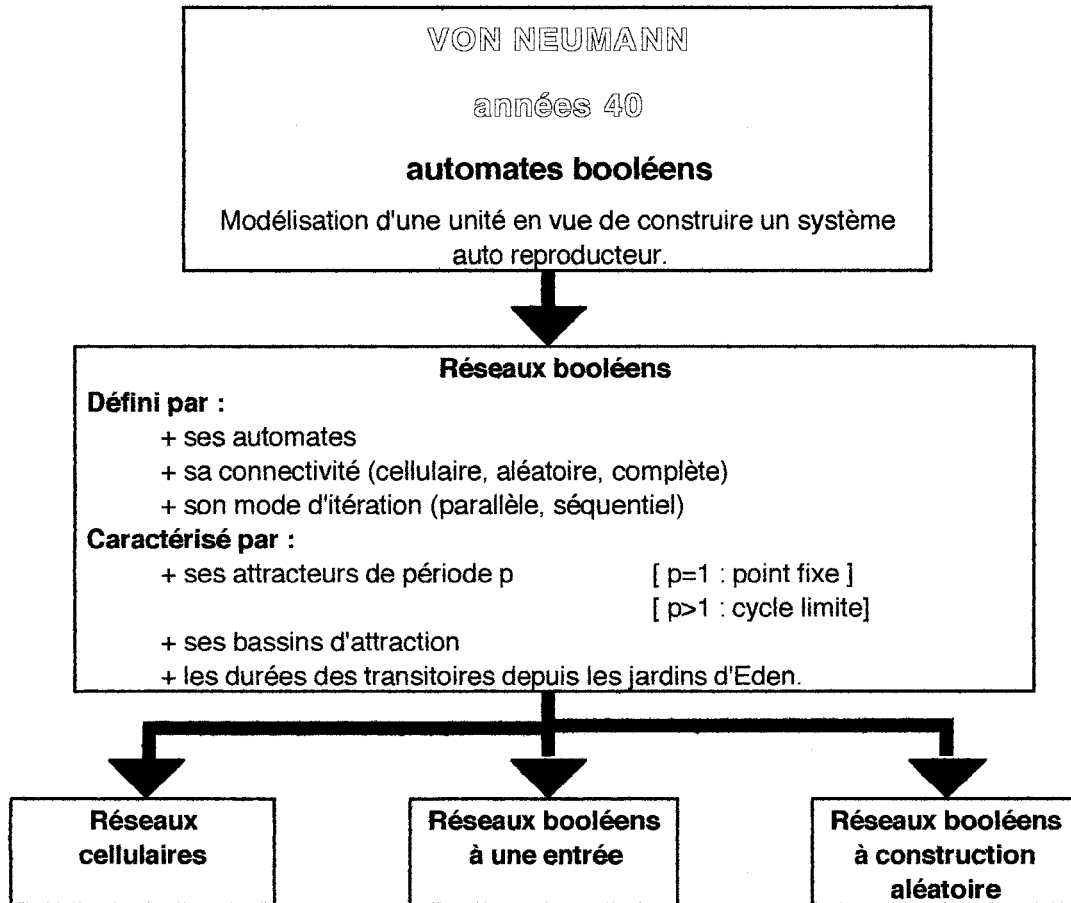


A peu près à la même période apparaissent deux modélisations d'unités de systèmes connexionnistes, qui reposent sur des inspirations différentes :

Pour VON NEUMANN, il s'agit de modéliser un comportement auto reproducteur.

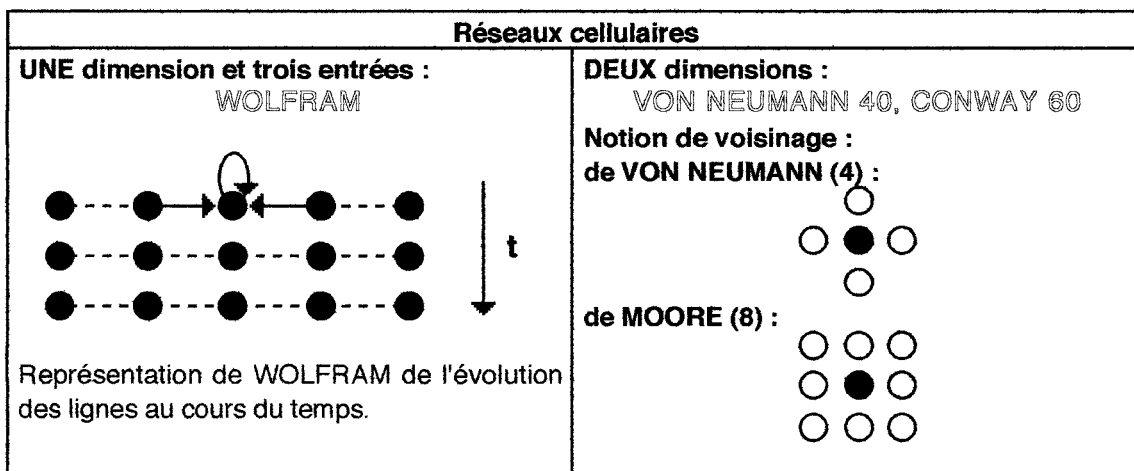
Pour MAC CULLOCH et PITTS, il s'agit de modéliser sommairement le fonctionnement d'un neurone réel.

3.5.3.2 Les descendances de la modélisation de VON NEUMANN



3.5.3.2.1 LES RÉSEaux CELLULAIRES

Réseaux cellulaires



3.5.3.2.2 LES RÉSEAUX BOOLÉENS À UNE ENTRÉE

Réseaux booléens à une entrée

Il apparaît des sous réseaux appelés CRABES, formés :

- + d'une tête
- + d'une boucle
- + de pattes

La boucle peut être frustrée (nombre impair de liaisons négatives) ou non frustrée.

3.5.3.2.3 LES RÉSEAUX BOOLÉENS À CONSTRUCTION ALÉATOIRE

Réseaux booléens à construction aléatoire

Les résultats de KAUFFMAN ne sont toujours pas démontrés dans le cas général. Seuls les cas extrêmes de connectivité un ou totale sont bien compris. Soit k la connectivité :

+ pour $k=1$ les réseaux sont constitués de crabes indépendants.

+ pour $k = N$

le nombre des attracteurs varie linéairement avec celui des unités

il existe en moyenne un cycle de longueur un

la période varie comme une exponentielle de N : $T = 0,63 \cdot 2^{\frac{N}{2}}$

L'attitude de l'ingénieur est d'imaginer des procédés permettant de construire des systèmes réalisant un comportement donné ou les mieux adaptés à une tâche donnée.

Nous prenons l'attitude inverse quand nous nous posons la question : Quelles sont les propriétés d'un système connexionniste fourni a priori ?

SYSTEME ETUDIE : Réseau booléen à construction aléatoire.

PROPRIETE ETUDIEE : La période des attracteurs.

Elle varie bien sur d'un réseau à un autre. Nous allons nous intéresser à son ordre de grandeur.

RESEAU ETUDIE : Les fonctions de transition des unités sont tirées au sort parmi toutes celles possibles. Les connexions entre les automates sont elles aussi tirées au sort. Un grand nombre de réseaux sont tirés au sort.

DETERMINATION DES ATTRACTEURS : Pour chaque réseau, on tire au sort une configuration initiale, on laisse évoluer vers un attracteur, on détermine alors la période de cet attracteur. On recommence sur un grand nombre de configurations initiales.

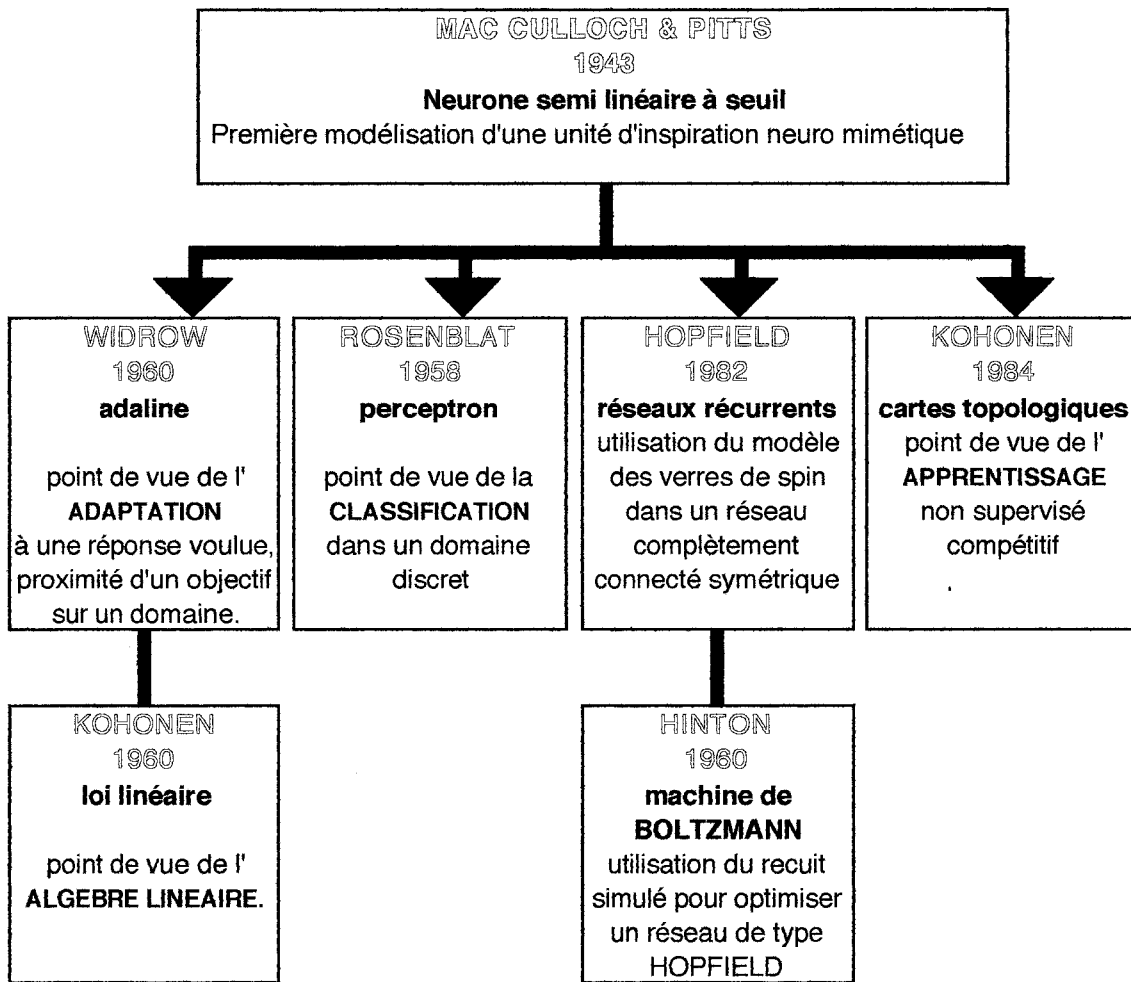
RESULTATS OBTENUS PAR KAUFFMANN :

Mise en évidence de deux régimes distincts selon la connectivité :

Connectivité 2 : la période moyenne varie comme la racine carrée du nombre N des unités du système. (Il en est de même pour le nombre des attracteurs).

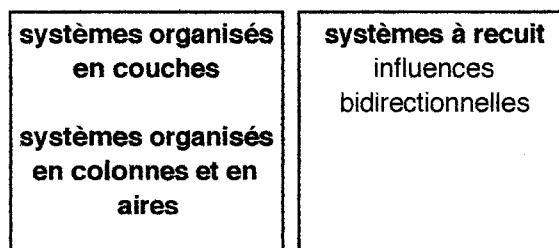
Dès que la connectivité atteint 4, la période varie comme une exponentielle du nombre N des unités.

3.5.3.3 Les descendances du modèle de MAC CULLOCH et PITTS



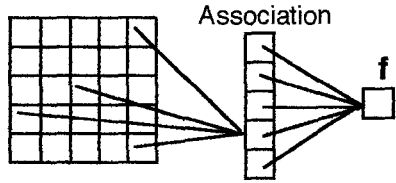
On distingue quatre descendances qui se caractérisent les unes par rapport aux autres par des points de vues différents.

Les évolutions les plus récentes :



3.5.3.3.1 LE PERCEPTRON DE ROSENBLATT [1958]

Présentation :

<p>PERCEPTRON : Un perceptron est formé de trois couches : la première appelée RETINE (ce sont des cellules d'entrée et non pas des automates) permet de présenter les exemples, la seconde, dite d'association permet de calculer des fonctions sur tout ou partie des unités de la rétine, la troisième enfin décide de la classification de l'exemple présenté sur la rétine.</p>	
---	--

Les fonctions de transition de la couche d'association sont fixées une fois pour toutes, ce sont souvent des fonctions booléennes non modifiables en apprentissage.

Le rôle de ces unités d'association est l'extraction des traits pertinents perçus sur la rétine. Seuls les poids synaptiques des connexions de l'unité de décision avec les unités d'association sont modifiables par apprentissage.

L'unité de décision est un automate à seuil. Dans un perceptron la décision est réalisée à partir de caractéristiques extraites de la rétine par des cellules d'association.

Si l'on établit un parallèle avec le domaine médical, la rétine code les symptômes, la couche intermédiaire les syndromes et la cellule de décision le diagnostic de la maladie.

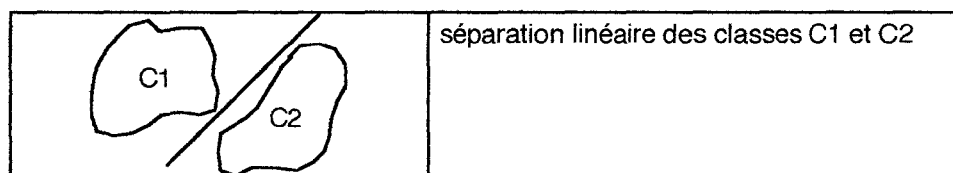
Le perceptron fonctionne sur le modèle de comportement : "stimulus réponse". Il "calcule" en sortie, dans le domaine de la classification d'objets, une réponse en fonction d'un stimulus présenté en entrée.

Problème posé :

Étant donnés deux ensembles C1 et C2 représentant deux classes d'exemples, on veut décider si un exemple E donné correspond à la classe C1 ou à la classe C2.

- si E est de la classe C1 alors on a une décision positive,
- si E est de la classe C2 alors on a une décision négative.

Le mécanisme de décision reposera sur une séparation linéaire des deux classes :



Faisabilité :

Le théorème du perceptron stipule qu'il existe une procédure d'apprentissage qui converge en un nombre fini d'étapes vers une solution quand elle existe, c'est à dire quand C1 et C2 sont linéairement séparables.

Dans le cas contraire, il n'existe même pas de solution approchée.

Description :

Nom : Perceptron de ROSENBLATT

Architecture :

UNITES :

Rétine	<i>Présentation des exemples</i>
---------------	----------------------------------

Association	
état interne X_i :	Booléen 0 ou 1
fonctions de transition f_i :	Spécifique à chaque unité
nombre d'unités :	Noté : N-1

Décision	
état interne X_i :	Booléen 0 ou 1
fonctions de transition f_i :	Semi linéaire à seuil
nombre d'unités :	Une

CONNEXIONS :	
influences :	Unidirectionnelles
agencement :	Une couche d'association entre les unités d'entrée et une unité de décision en sortie.

Dynamique :

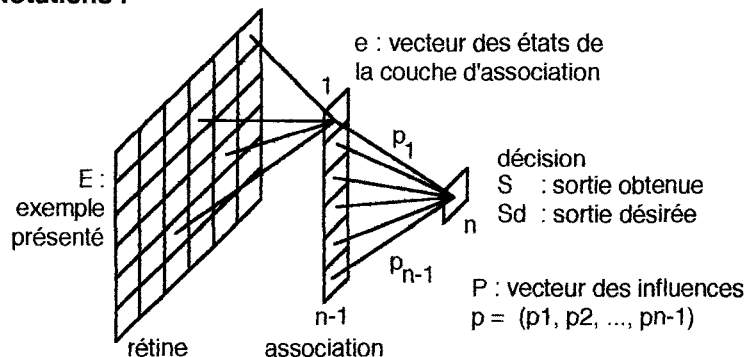
EN EXPLOITATION :	
	Application des fonctions de transition des unités

EN APPRENTISSAGE	
	Dynamique de ROSENBLATT

Inspiration :

simuler une assemblée de neurones

Notations :



Indicateur associé à un état du système :

critère : $J(P)$ tel que $J(P) = \sum - P \cdot e$: somme pour tous les exemples mal classés par le perceptron de vecteur d'influences P

Contrainte imposée au système :

si l'exemple E est de la classe C1 alors $S = P \cdot e = 1$
 si l'exemple E est de la classe C2 alors $S = P \cdot e = 0$

Dynamique de ROSENBLATT :

Apprentissage supervisé par correction d'erreur :

"La modification des influences est fonction de l'écart entre l'état désiré et celui constaté en sortie seuillée"

Dans ce cas la modification des influences par gradient s'écrit :

$$\Delta P_i = \mu(t) \cdot [S^d - S] \cdot e_i$$

$\mu(t)$ est une constante strictement positive, chiffrant l'intensité d'apprentissage à l'instant t.

On a $J(P) = \sum_{E \in M} (-P \cdot e)$, où M est l'ensemble des exemples mal classés par le perceptron

P

La ième composante du gradient de J sera : $\text{grad}(J^i) = \frac{\partial J}{\partial P_i} = -e^i$

donc $\text{grad}(J) = \sum_{E \in M} (-e)$

Seules les influences de la dernière couche du perceptron sont modifiables, la règle ci-dessus ne s'applique que pour les interactions entre les couches 2 et 3.

Notons H la fonction de HEAVISIDE : $H(x) = 1$ si $x \geq 0$ sinon 0.

L'algorithme sera donc :

1. Choisir un perceptron modélisé par un vecteur d'influence initial noté P^1
2. Tant que tous les exemples E ne sont pas bien classés :
 examiner un exemple E fournissant le vecteur d'état e
 faire évoluer : P^t en P^{t+1} par $P^{t+1} = P^t + \Delta P$ avec $\Delta P = [S^d - H(P^t \cdot e)] \cdot e$

On aura : $\Delta P_i = [S^d - H(P^t \cdot e)] \cdot e^i$

état désiré S^d	1	1	0	0
état obtenu $H(P^t \cdot e)$	1	0	1	0
évolution souhaitée	néant	augmenter les influences car il y a déficit de l'obtenu par rapport au désiré	diminuer les influences car il y a excédant de l'obtenu par rapport au désiré	néant

Ainsi une autre règle équivalente d'évolution pourrait être :

si E n'est pas bien classé
alors si E est de la classe C1
 alors $\Delta P = e$ { E est de la classe C1}
 sinon $\Delta P = -e$ { E est de la classe C2}
sinon $\Delta P = 0$

Exemple de perceptron :

Il y a 13 unités dans le réseau : Les unités d'entrée sont numérotées de 1 à 9 Il n'y a pas d'unité de commande L'unité de sortie est numérotée 13	Rétine (9 unités)	Association (3 unités)	Décision (1 unité)													
	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">2</td><td style="padding: 2px 5px;">3</td></tr> <tr><td style="padding: 2px 5px;">4</td><td style="padding: 2px 5px;">5</td><td style="padding: 2px 5px;">6</td></tr> <tr><td style="padding: 2px 5px;">7</td><td style="padding: 2px 5px;">8</td><td style="padding: 2px 5px;">9</td></tr> </table>	1	2	3	4	5	6	7	8	9	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 5px;">10</td></tr> <tr><td style="padding: 2px 5px;">11</td></tr> <tr><td style="padding: 2px 5px;">12</td></tr> </table>	10	11	12	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 5px;">13</td></tr> </table>	13
	1	2	3													
	4	5	6													
7	8	9														
10																
11																
12																
13																

La matrice P des influences sera ici $P = (p_{10}, p_{11}, p_{12})$ en notation simplifiée. p_{10} au lieu de $p_{13,10}$.

Quelques éléments de théorie sur le perceptron :

Théorème 1:

Toute fonction booléenne est calculable par un perceptron.

Idée de preuve : Comme toute fonction booléenne est décomposable en une disjonction de conjonctions, il suffit de simuler les "et" à l'aide des unités d'association de la première couche et le "ou" à l'aide de l'unité de décision de la dernière couche.

Théorème 2 : [MINSKY PAPERT]

Un perceptron à diamètre limité ne peut décider si un motif est simplement connexe ou non.

Définition : Un perceptron à diamètre limité chiffré par "d" est un perceptron tel que toute unité d'association possède la propriété suivante : L'unité d'association n'est reliée qu'à des unités de la rétine distantes d'au plus "d" unités.

Technique de preuve : par l'absurde

Preuve : Supposons qu'un perceptron à diamètre limité k soit capable de reconnaître des formes simplement connexes. Soit k le côté d'un carré correspondant au diamètre du perceptron :

Soient les quatre figures suivantes dont les longueurs sont supérieures à k :



figure 1



figure 2



figure 3



figure 4

Répartissons les unités d'association en catégories :

Catégorie 1 : unités d'association reliées à au moins une unité du bord gauche de la rétine.

Catégorie 2 : unités d'association reliées à au moins une unité du bord droit de la rétine.

Catégorie 3 : unités d'association reliées à aucune unité du bord de la rétine.

Étudions la réaction, le comportement du perceptron vis à vis de la rétine de la figure 1 :
La réponse du perceptron est NON, car la figure n'est pas simplement connexe (la figure est en noir sur fond blanc). Soit S_1 la somme pondérée des sorties des unités d'association de la catégorie 1, S_2 celle des unités de catégorie 2, et S_3 celle des unités de catégorie 3. Soit s le seuil de l'unité de décision. La réponse de l'unité de décision étant NON on a $S_1 + S_2 + S_3 < s$

Étudions la réaction, le comportement du perceptron vis à vis de la rétine de la figure 2 :
Seules les sommes pondérées des unités de catégorie 1 vont varier. Soit S'_1 cette nouvelle somme.
Comme le perceptron doit répondre OUI on a $S'_1 + S_2 + S_3 > s$ donc $S'_1 > S_1$

Étudions la réaction, le comportement du perceptron vis à vis de la rétine de la figure 3 :
C'est comme pour le cas précédent mais pour les unités de catégorie 2 donc $S'_2 > S_2$

Étudions la réaction, le comportement du perceptron vis à vis de la rétine de la figure 4 :
Quel doit être la réaction du perceptron :
Il va recevoir les influences : $S'_1 + S'_2 + S_3$. Or comme $S'_1 + S_2 + S_3 > s$ et $S'_2 > S_2$ on en déduit que $S'_1 + S'_2 + S_3 > s$ donc que le perceptron va répondre OUI. Or il faut qu'il réponde NON.

Ce qui établit par l'absurde que le perceptron à diamètre limité ne peut reconnaître des figures simplement connexes.

Théorème 3:

Convergence du perceptron.

Notons si l'exemple E est de la classe C_1 : $Z = e$

et si E est de la classe C_2 alors $Z = -e$.

Une mauvaise classification entraîne la modification : ajouter $\mu \cdot Z$ à P .

La récurrence est alors :

$P_0 = 0$ et pour tout t : $P_{t+1} = P_t + \mu \cdot Z_t$ et $P_t = \mu \cdot \sum Z_i$

(la somme pour i de 1 à t)

D'une correction à la suivante la norme du vecteur P des influences devient :

$$\| P_{t+1} \|^2 = \| P_t \|^2 + 2 \cdot \mu \cdot P_t \cdot Z_t + \mu^2 \cdot \| Z_t \|^2$$

Comme le produit scalaire $P_t \cdot Z_t$ est toujours négatif quand il y a correction et que μ est positif, on a la certitude que

$$\| P_{t+1} \|^2 \leq \| P_t \|^2 + \mu^2 \cdot \| Z_t \|^2$$

Les exemples à classer étant bornés en norme, notons $M = \max \{ \| Z_t \|^2 \}$

De proche en proche, on s'assure donc que la norme quadratique de P ne diverge pas plus vite que linéairement avec le nombre chiffré t de corrections :

$$\| P_t \|^2 \leq t \cdot \mu^2 \cdot M$$

Supposons qu'il existe au moins un vecteur P d'influences qui classe correctement tous les exemples.

Les produits scalaires $\langle P, z \rangle$ sont tous positifs, en excluant la nullité limite pour la classe 1.
Leur minorant est $m = \min \{ \langle P, z \rangle \}$

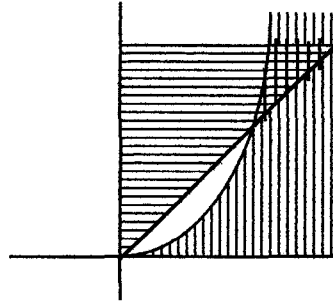
D'après la récurrence on a : $P_t^T \cdot P_t = \mu \sum P_i^T \cdot Z_i \geq \mu \cdot t \cdot m$

d'après l'inégalité de SCHWARZ qui exprime que

$$(P_t^T \cdot Q)^2 \leq \|P_t\|^2 \cdot \|Q\|^2$$

$$\text{on a : } \frac{t^2 \cdot m^2}{\|P\|^2} \leq \|P_t\|^2 \leq t \cdot M$$

Coincée entre une croissance au moins quadratique et au plus linéaire, la norme du vecteur des influences P_t ne peut pas croître indéfiniment



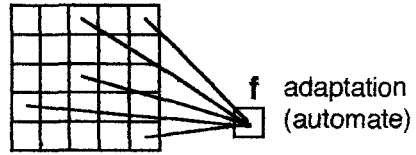
Inconvénients :

L'algorithme du perceptron converge lentement et s'arrête dès qu'une solution est trouvée. C'est généralement l'hyperplan tangent aux classes C1 et C2. Cette solution n'est pas optimale.

3.5.3.3.2 L'ADAPTATEUR LINÉAIRE ADALINE DE WIDROW [1960]

Présentation:

C'est un automate réalisant un bilan d'influences exercées par les unités d'une rétine, capable de s'adapter pour répartir une série d'exemples en deux groupes. La capacité d'un ADALINE est limitée aux problèmes linéairement séparables, c'est à dire pour lesquels les deux groupes à reconnaître peuvent être séparés par une droite, un plan, un hyperplan dans l'espace de leur représentation.



A première vue, Perceptron et ADALINE semblent se ressembler, il y a cependant des différences notoires exprimées ci-dessous.

Description :

Nom :

ADALINE de WIDROW - HOFF

Architecture :

UNITES :

Rétine	<i>Présentation des exemples</i>
---------------	----------------------------------

Adaptation	
état interne X_i :	Continu
fonctions de transition f_i :	Identique
nombre d'unités :	Une

CONNEXIONS :	
influences :	Unidirectionnelles
agencement :	Toutes les unités d'entrée sont connectées à l'unité de sortie

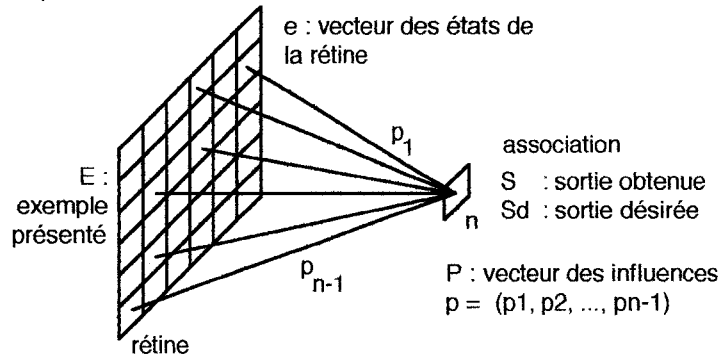
Dynamique :

EN EXPLOITATION :	Application d'une sommation
-------------------	------------------------------------

EN APPRENTISSAGE	Modification des influences par la dynamique de WIDROW HOFF
------------------	--

Notations :

- e vecteur des états du système
- e^T vecteur transposé du vecteur e
- P matrice des influences
- i, j, k numéro d'une unité
- e^d état désiré pour l'unité de sortie
- s état calculé pour l'unité de sortie



Indicateur associée à un état du système :

une énergie : $J(P, e)$

Inspiration :

bâtir des filtres linéaires en jouant sur les influences

Dynamique de WIDROW - HOFF :

"Les éléments en interaction se réarrangent afin de minimaliser l'énergie du système qu'ils constituent."

La loi de WIDROW-HOFF concerne les réseaux qui ne possèdent pas d'unités cachés (c'est à dire qu'il y a uniquement des unités d'entrée ou de sortie). Un exemple typique est l'ADALINE.

La modification des poids par gradient s'écrit :

$$\Delta P_i = 2 \cdot J(P) \cdot [S^d - \sum_k P_k \cdot e_k] \cdot e_i$$

Explication

On définit une énergie $J(P)$ comme une fonction d'erreur entre la sortie désirée S^d et la sortie calculée $P \cdot e$, par la somme cumulée des carrés des erreurs élémentaires pour chacune des unités numérotées i et pour chacun des exemples présentés .

On cherche à minimiser l'erreur $J(P)$ par une méthode de gradient.

On a alors les formules :

$$J(P) = (P \cdot e - S^d)^2 \text{ d'où } \text{grad } P(J) = 2 \cdot (P \cdot e - S^d) \cdot e^T$$

Ce qui donne alors la règle d'évolution de la matrice P :

$$\Delta P_i = 2 \cdot J(P) \cdot [S^d - \sum_k P_k \cdot e_k] \cdot e_i$$

Ici, on calcule un signal d'erreur fonction de l'écart entre la sortie désirée et la somme des $P_k \cdot e_k$ qui représente le bilan des influences subies par l'unité de décision

Dans la dynamique de ROSENBLATT, on calculait un autre signal d'erreur : l'écart entre la sortie désirée et la somme des $P_k \cdot e_k$ seuillée c'est à dire s

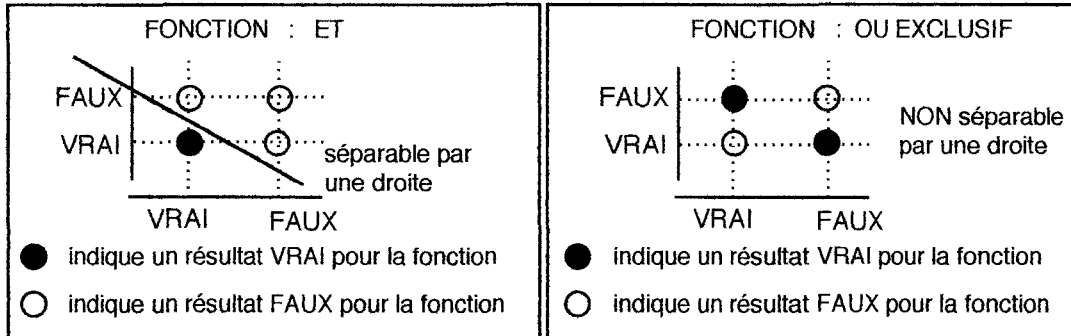
Contrairement à la loi de WIDROW, la loi du perceptron ne minimalisait pas une énergie et sa convergence n'est pas toujours assurée.

Deux avantages apparaissent pour la dynamique de WIDROW HOFF par rapport à celle du perceptron :

- + obtention d'une solution "approchée" quand le problème n'est pas linéairement séparable.
- + quand il y a une solution de séparabilité linéaire, elle est plus "robuste"

Exemple commun au perceptron et à l'ADALINE :

On veut classifier en deux groupes les réponses d'une fonction "et" et d'une fonction "ou exclusif".



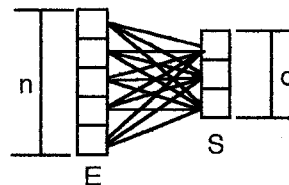
Ainsi, il apparaît que la "puissance de calcul" d'une unité à fonction de transition à seuil est celle d'une séparation linéaire.

3.5.3.3 LE FILTRE LINÉAIRE

Nous allons nous intéresser à un type de réseau dans lequel les unités sont continues, en privilégiant un point de vue algébrique linéaire.

Présentation :

FILTRE LINEAIRE : une sollicitation E en entrée codée sur n unités, produit une réaction S sur C unités de sortie selon une relation linéaire du type $S = A \cdot E$



Description :

Nom :

linéaire

Architecture :

UNITES :	
état interne X_i :	Continu
fonctions de transition f_i :	Identique
nombre d'unités :	noté : N

CONNEXIONS :	
influences :	Unidirectionnelles
agencement :	Des entrées vers les sorties

Dynamique :

†65

Application d'une sommation

EN APPRENTISSAGE

Imposition d'une capacité innée

Inspiration :

mathématique

Indicateur associé à un état du système :

néant

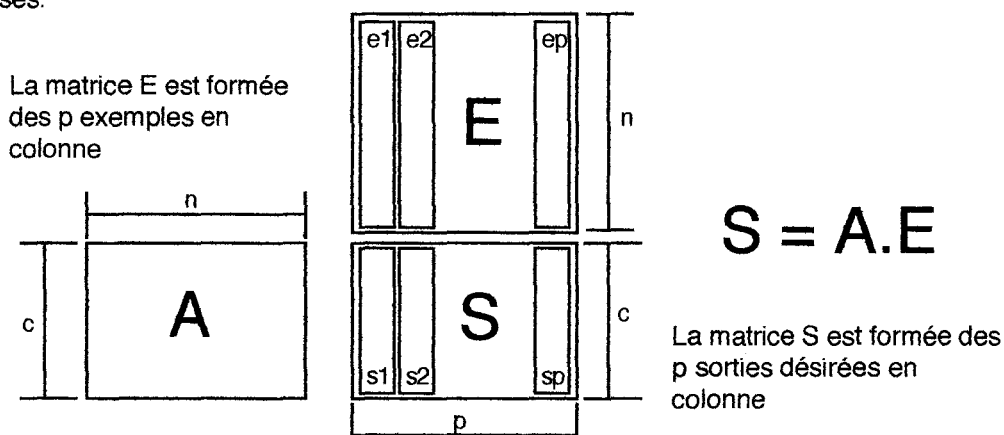
Loi linéaire :

"La sortie s est fonction linéaire de l'entrée e par une formule $s = A \cdot e$ "

La formule $s = A \cdot e$ exprime que la sortie s est le résultat de l'application de A à l'entrée e . Cette formule est dite linéaire car elle est de la forme $y = a \cdot x$ dont la représentation graphique mathématique est une ligne droite.

Problème posé :

On cherche à créer une mémoire associative linéaire classifiant "p" exemples prototypes en "c" classes.



- A est une matrice "c" lignes et "n" colonnes.
- E est une matrice "n" lignes sur "p" colonnes.
- S est une matrice "c" lignes sur "p" colonnes.

Ainsi on **force** le résultat $A \cdot e^1 = s^1$ pour l'exemple prototype numéro 1, mais aussi pour tous les autres. On "reconnait" la classe numéro 1 par s^1 parce que toutes ses composantes sont nulles sauf celle en position numéro 1, de même s^2 reconnaît la classe numéro 2 parce que toutes ses composantes sont nulles sauf celle en position numéro 2.

Le problème mathématique posé est la résolution de l'équation : $A \cdot E = S$ d'inconnue la matrice A.

A. Solution exacte : la matrice pseudo inverse de PENROSE.

L'équation $A \cdot E = S$, admet des solutions qui s'expriment à l'aide de la matrice pseudo inverse de E notée E^+ Ces solutions sont : $A = S \cdot E^+ + Q \cdot (I - E \cdot E^+)$, où Q est une matrice quelconque.

La solution de plus petite norme quadratique est : $A = S \cdot E^+$

Le calcul d'une matrice pseudo inverse est très complexe.

Différents algorithmes permettent de "calculer" la matrice pseudo inverse de la matrice E.

+ Algorithme de GREVILLE (voir paragraphe 3.5.4.2)

+ Algorithme de KALABA - RASAKHOO. (voir paragraphe 3.5.4.2)

B. Approximation proposée par KOHONEN

Dans le but de ne pas calculer la matrice pseudo inverse des exemples, KOHONEN propose de remplacer la solution $A = S \cdot E^+$ par $A = S \cdot E^T$ où E^T est la matrice transposée de E, qui se "calcule" simplement en échangeant les lignes et les colonnes de la matrice E.

Cette hypothèse simplificatrice suppose que les exemples soient bien indépendants les uns des autres, qu'ils soient suffisamment "distants".

Exemple :

On veut mémoriser les exemples prototypes :

$e^1 = (1, -1, -1, 1, -1, 1)$ et $e^2 = (1, 1, 1, -1, -1, -1)$ pour le réseau de six unités décrit en exemple de réseau d'automates au paragraphe de description des systèmes connexionnistes.

Question préalable :

" Ces deux "exemples" sont-ils assez distants l'un de l'autre pour ne pas être "confondus"?

Pour mesurer la "distance" entre ces deux exemples, on va utiliser la distance de HAMING notée HA. La distance $HA(e^1, e^2)$ se calcule en comptant le nombre d'unités dans e^1 et dans e^2 qui sont dans un état différent.

unité numéro :	1	2	3	4	5	6
configuration e^1 du réseau:.....	1	-1	-1	1	-1	1
configuration e^2 du réseau:.....	1	1	1	-1	-1	-1
différence :	N	O	O	O	N	O

(N veut dire non et O veut dire oui)

Il y a quatre différences donc $HA(e^1, e^2) = 4$.

La configuration e^1 du réseau sera mémorisée si $e^1 = A \cdot e^1$ c'est à dire si le réseau restitue en sortie le même état qu'en entrée, de même pour la configuration e^2

On a donc $A = e \cdot e^T$ soit : $A = e^{1T} \cdot e^1 + e^{2T} \cdot e^2$

Le produit $e^{1T} \cdot e^1$ est un tableau de 6 cases sur 6 cases. Le contenu de chacune de ces 36 cases se calcule par la multiplication des états de e^1 en colonne et de e^{1T} en ligne.

$$\begin{array}{c}
 e^{1T} \quad 1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \\
 \\
 \begin{array}{c}
 1 \\
 -1 \\
 e^1 \cdot -1 \\
 1 \\
 -1 \\
 1
 \end{array}
 \left| \begin{array}{cccccc}
 1 & -1 & -1 & 1 & -1 & -1 \\
 -1 & 1 & 1 & -1 & 1 & 1 \\
 -1 & 1 & 1 & -1 & 1 & 1 \\
 1 & -1 & -1 & 1 & -1 & -1 \\
 -1 & 1 & 1 & -1 & 1 & 1 \\
 1 & -1 & -1 & 1 & -1 & -1
 \end{array}
 \right.
 \end{array}$$

De même pour le produit $e^{2T} \cdot e^2$.

La somme des deux tableaux est le tableau obtenu en ajoutant les contenus des cases correspondantes des deux tableaux :

On arrive ainsi au tableau A :

$A = \begin{bmatrix} 2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & -2 & -2 & 2 & 0 & 2 \\ -2 & 0 & 0 & 0 & 2 & 0 \\ 0 & -2 & -2 & 2 & 0 & 2 \end{bmatrix}$	<p>Calcul de $A \cdot e^1 = s^1$</p> $ \begin{bmatrix} 2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & -2 & -2 & 2 & 0 & 2 \\ -2 & 0 & 0 & 0 & 2 & 0 \\ 0 & -2 & -2 & 2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -8 \\ -8 \\ 8 \\ -4 \\ 8 \end{bmatrix} $
--	---

Dans le cas de cette approximation, on seuille s^1 pour retrouver e^1

C. Approximation proposée par WIDROW - HOFF

Le problème de la recherche de cette matrice A a été résolu aux paragraphes précédents par une méthode itérative proposée par WIDROW - HOFF. La matrice A est obtenue à partir d'une matrice A1 qui minimise une fonction d'erreur J entre S (sortie obtenue par $A_k \cdot E$) et S_d (sortie désirée) par :

$$A_{k+1} = A_k - \mu_k \cdot (S_d - A_k \cdot E) \cdot e^T.$$

3.5.3.3.4 LE RÉSEAU DE HOPFIELD

Présentation :

HOPFIELD cherche à modéliser une mémoire adressable par son contenu : une forme mémorisée étant retrouvée par une stabilisation du réseau stimulé par une partie caractéristique de la forme. Son inspiration repose sur la physique statistique.

Description :

Nom :

Réseau récurrent de HOPFIELD

Architecture :

UNITES :	
état interne X_i :	discret 0 ou 1
fonctions de transition f_i :	semi linéaire à seuil u_i
nombre d'unités :	noté : N

CONNEXIONS :	
influences :	unidirectionnelles
agencement :	Complètement connecté, les unités n'étant pas reliées à elles mêmes. Les connexions sont SYMETRIQUES

Dynamique :

EN EXPLOITATION :	
	Choix aléatoire séquentiel des unités mises à jour selon l'une des dynamiques décrites ci-après.

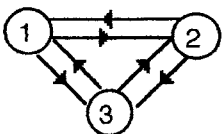
EN APPRENTISSAGE	
	Selon la dynamique de HEBB

Inspiration :

modélisation d'une mémoire adressable

Notation :

Définitions et Notations sur un exemple :
réseau totalement connecté de trois unités numérotées de 1 à 3



Vecteur d'état du réseau : $e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$
e est élément de $\{0, 1\}^3$

Vecteur d'influence des connections du réseau (poids) : $P = \begin{bmatrix} 0 & 1/2 & -1 \\ 1/2 & 0 & 1/4 \\ -1 & 1/4 & 0 \end{bmatrix}$

Vecteur des seuils des transitions : $U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$

On note : V^i la ième composante du vecteur V.

Indicateur associée à un état du système :

modélisé par une énergie : $J(P,e)$

$$J(P,e) = \frac{1}{2} \sum_{i \neq j} P^{ij} \cdot e^i \cdot e^j - \sum_i u^i \cdot e^i$$



l'énergie d'un état EST l'interaction entre deux spins MOINS l'intervention d'un champ extérieur

(comme P est symétrique, les interactions sont régulières)

Dynamique d'apprentissage du réseau de HOPFIELD : HEBB

Loi de HEBB [1949]:

"Si deux éléments sont actifs ensemble, alors leur liaison sera renforcée".

<p>Légende :</p> <ul style="list-style-type: none"> ● unité active ○ unité passive 	 <p>connexion renforcée</p>	 <p>connexion non modifiée</p>
--	--	--

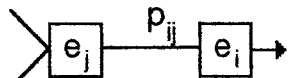
Cette loi est une loi "naturelle" bien connue des physiologistes.

Mathématisons la :

Notons Δp_{ij} la modification de l'influence p_{ij} entre l'unité source numéro j et l'unité but numéro i
 p_{ij} est l'influence exercée par l'unité numéro j sur l'unité numéro i.

notons e_j l'état de l'unité numéro j, e_i l'état de l'unité numéro i, et μ une constante chiffrant l'intensité de l'apprentissage, ou encore l'attention que porte le réseau à son environnement.

La loi de HEBB peut s'exprimer mathématiquement par la formule :

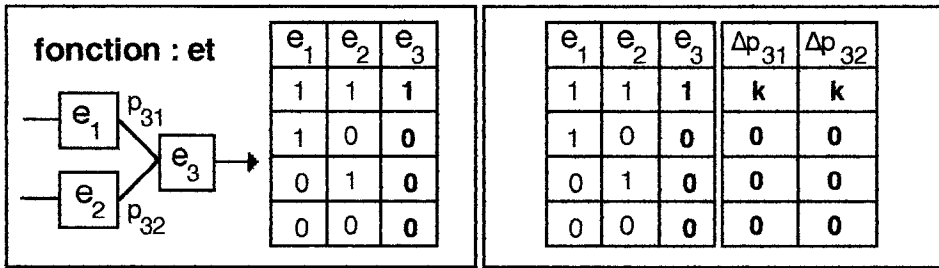
$\Delta P_{ij} = k \cdot e_j \cdot e_i$	
---	---

Mise en œuvre d'un apprentissage avec la loi de HEBB pour une fonction "et".

Calculons Δp_{31} et Δp_{32} dans chaque cas :

$$\Delta p_{31} = k \cdot e_1 \cdot e_3$$

$$\Delta p_{32} = k \cdot e_2 \cdot e_3$$

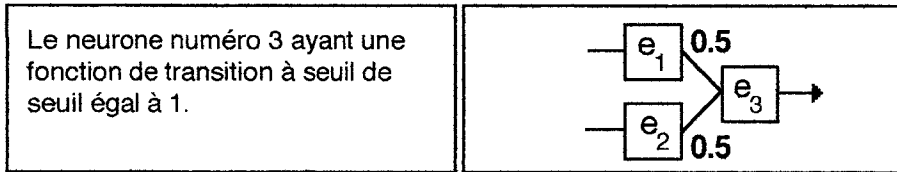


Le problème est :

Quelles valeurs faut-il donner à p_{31} et à p_{32} pour que le réseau calcule une fonction "et".

Si l'on part d'une configuration où les poids sont nuls, ceux-ci vont évoluer vers une configuration où ils seront égaux, puisque Δp_{31} et Δp_{32} sont toujours égaux .

On aura donc $p_{31} = p_{32}$, et le réseau sera :



Limitation :

Le fait de ne tenir compte que de la corrélation entre les unités d'entrée et de sortie est insuffisante, et de nombreuses fonctions ne sont pas "calculables" par la loi de HEBB, contrairement à la fonction "et" ci-dessus.

Résumé : Algorithme d'apprentissage

- 1• On commence avec un ensemble de connexions toutes nulles [Hypothèse de la table rase]
- 2• On force le système dans un état particulier e^S .
- 3• On examine tous les couples (i, j) d'unités et on augmente p_{ij} de Δp_{ij} selon la règle :

si $e_i^S = +1$ et $e_j^S = +1$	alors $\Delta p_{ij} = +1$	(unités simultanément actives)
si $e_i^S = +1$ et $e_j^S = -1$	alors $\Delta p_{ij} = -1$	(unités opposées)
si $e_i^S = -1$ et $e_j^S = +1$	alors $\Delta p_{ij} = -1$	(unités opposées)
si $e_i^S = -1$ et $e_j^S = -1$	alors $\Delta p_{ij} = +1$	(unités simultanément inactives)

(On pourrait modifier la règle ci dessus pour ne pas modifier l'influence p_{ij} lorsque les unités sont simultanément inactives.)

DYNAMIQUE n° 1 d'actualisation du réseau : [HOPFIELD 82]

A chaque moment t , l'unité à actualiser est choisie aléatoirement conformément au mécanisme asynchrone de "mise à feu" du neurone biologique.

Équation de récurrence :

$$e_{t+1}^i = H((P \cdot e_t - U)^i)$$

où H est la fonction de HEAVISIDE : $H(x) = \text{si } x \geq 0 \text{ alors } 1 \text{ sinon } 0$;

On a un comportement déterministe dans une dynamique aléatoire.

Algorithme simulant la conduite du réseau non bruité :

tant que pas stable

faire

+ choisir avec la probabilité uniforme l'unité à actualiser

+ actualiser cette unité en appliquant la récurrence

fait

Utilisation du réseau en mémoire associative :

Le problème à résoudre est de trouver la matrice P pour que les minima locaux de l'énergie $J(P)$ soient prédéterminés et correspondent aux motifs à apprendre.

DYNAMIQUE n° 2 d'actualisation du réseau : [HOPFIELD 85]

On introduit du bruit sous forme d'une suite de variables aléatoires indépendantes. On note (B_t) cette suite. (t est un entier naturel).

Équation de récurrence :

$$e_{t+1}^i = H((P \cdot e_t - U)^i - K \cdot B_t)$$

où K est un facteur de bruit.

Équation de la probabilité de l'état d'une unité à actualiser :

$$p(e_{t+1}^i = 1) = p(B_t \leq \frac{(P \cdot e_t - U)^i}{K}) = g_k((P \cdot e_t - U)^i)$$

où g_k est la fonction de répartition de la loi gaussienne d'écart type k .

Rappel :

$$g_k(x) = \frac{1}{2} (1 + \text{Erf}(\frac{x}{k}))$$

$$\text{et } \text{Erf}(x) = \text{si } x > 0 \text{ alors } \frac{2}{\sqrt{2\pi}} \int_0^x e^{-\frac{t^2}{2}} dt \text{ sinon } -\text{Erf}(-x)$$

Algorithme de Monte CARLO simulant la conduite du réseau bruité :

tant que pas stable

faire

+ choisir avec la probabilité uniforme l'unité numéro i, à actualiser

+ calculer $g_k ((P \cdot e_t - U)^i)$

+ tirer au hasard l'état suivant de l'unité numéro i en fonction de la probabilité calculée

fait

DYNAMIQUE n° 3 d'actualisation du réseau : [GLAUBER - Métropolis]

Dynamique de GLAUBER [Modèle de relaxation isotherme]:

On remplace la fonction g_k de la dynamique numéro 2 par la fonction h_T

$$h_T(x) = \frac{1}{1 + e^{-\frac{x}{T}}} \text{ A la place d'un facteur de bruit } k, \text{ on a une température } T.$$

On note T une température et (K_t) une suite de variables aléatoires (t est un entier naturel).

Équation de récurrence :

$$e_{t+1}^i = H((P \cdot e_t - U)^i - T \cdot K_t)$$

Équation de la probabilité de l'état d'une unité à actualiser:

On note e' un état du réseau qui diffère de l'état e du réseau uniquement par l'état de l'une de ses composantes.

On écrit : $e' \approx e$

Quand $e_{t+1}^i = 1$ et $e_t^i = 0$ on a $\Delta J = J(e_{t+1}) - J(e_t) = (P \cdot e_t - U)^i$

Quand $e_{t+1}^i = 0$ et $e_t^i = 1$ on a $\Delta J = - (P \cdot e_t - U)^i$

on a donc $p(e'/e) = \frac{1}{N} h_T(J(e') - J(e))$ où $e' \approx e$

$$\text{et } p(e/e) = 1 - \frac{1}{N} \sum_{e' \approx e} h_T(J(e') - J(e))$$

On peut ici exprimer la probabilité de transition en fonction de la variation d'énergie.

D'autre part :

$$p(e_{t+1}^i = 1) = p(K_t \leq \frac{(P \cdot e_t - U)^i}{T}) = h_T((P \cdot e_t - U)^i)$$

Dynamique de Métropolis [1950] [Modèle de relaxation atomique]:

On remplace la fonction h_T par une fonction h'_T telle que $h'_T(x) = \inf(1, e^{-\frac{x}{T}})$.

Ces deux dynamiques ont la même probabilité stationnaire qui est :

$$p_T(e) = \frac{e^{-\frac{J(e)}{T}}}{Z(T)} \text{ et } Z(T) = \sum_e e^{-\frac{J(e)}{T}}$$

Second aspect de représentation de connaissances : une étude ce cas : le dessin au trait.

C'est la distribution de GIBBS associée à la fonction d'énergie J . La fonction de répartition $Z(T)$ est une constante de normalisation. Cette distribution est la distribution d'entropie maximum parmi toutes les distributions ayant même énergie moyenne.

DYNAMIQUE n° 4 d'actualisation du réseau : [KIRPATRICK 83]

Algorithme du recuit simulé :

initialiser

répéter

RELAXER : laisser évoluer selon la dynamique de Métropolis ou de GLAUBER un temps suffisamment long

REFROIDIR : réduire la température T suivant une loi géométrique de raison 0,9.

jusqu'à ce que le réseau soit gelé (Refroidissement n'entraînant aucun changement des unités)

3.5.3.3.5 LA MACHINE DE BOLTZMANN

Présentation :

La machine de BOLTZMANN introduit une innovation par rapport au réseau de HOPFIELD, en distinguant deux catégories d'unités : des unités visibles et des unités cachées.

Description :

Nom :

machine de BOLTZMANN

Architecture :

UNITES :	
état interne X_i :	discret 0 ou 1
fonctions de transition f_i :	semi linéaire à seuil
nombre d'unités :	noté : N

CONNEXIONS :	
influences :	unidirectionnelles
agencement :	totalemment connecté à influences symétriques. Une unité ne s'influence pas elle même

Dynamique :

EN EXPLOITATION :	
	Choix aléatoire séquentiel des unités mises à jour selon l'une des dynamiques décrites pour les réseaux de HOPFIELD

EN APPRENTISSAGE	
	Apprentissage supervisé de BOLTZMANN

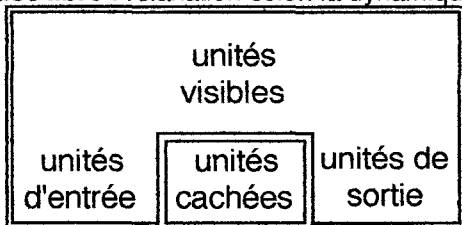
Inspiration :

Rapprocher la probabilité d'évolution en phase libre de celle d'évolution en phase contrainte.

Notation :

phase contrainte : relaxation selon la dynamique de Métropolis des seules unités cachées, les unités visibles étant fixées.

phase libre : relaxation selon la dynamique de Métropolis de toutes les unités.



$e = (e^{visi}, e^{cach})$: vecteur d'état du réseau

e^{visi} : vecteur d'état des unités visibles

e^{cach} : vecteur d'état des unités cachées

Dynamique d'apprentissage supervisé de BOLTZMANN

Loi de BOLTZMANN :

"On cherche à équilibrer la probabilité d'évolution en phase libre avec la probabilité d'évolution en phase contrainte".

On note $P_{\text{contraint}}$ ($e_{\text{cach}} = u / e^{\text{visi}} = v$) la probabilité en phase contrainte que le vecteur d'état e_{cach} des unités cachées du réseau soit égal à u , sachant que le vecteur d'état e^{visi} des unités visibles du réseau est égal à v .

On veut que $P_{\text{contraint}}(e_{\text{cach}} = u / e^{\text{visi}} = v) = P_{\text{libre}}(e_{\text{cach}} = u / e^{\text{visi}} = v)$

Mettons en place une distance G entre ces deux lois de probabilité. Notre objectif sera de minimaliser la distance : $G(P_{\text{contraint}}, P_{\text{libre}})$

Définition : $G(P_1, P_2) = \sum_e P_1(e) \cdot \log\left(\frac{P_1(e)}{P_2(e)}\right)$

Remarque : G prend le sens d'une entropie relative de la théorie de l'information.

La loi de BOLTZMANN peut alors être vue comme minimalisant un indicateur qui est une entropie relative.

Exprimons le gradient de G par rapport à une influence p_{ij}

$$\frac{\partial G(P_{\text{contraint}}, P_{\text{libre}})}{\partial p_{ij}} = - \sum_e P_{\text{contraint}}(e) \cdot \frac{1}{P_{\text{libre}}(e)} \cdot \frac{\partial P_{\text{libre}}(e)}{\partial p_{ij}}$$

Si la loi de probabilité d'évolution en phase libre suit une loi de GIBBS on a :

$$P_{\text{libre}}(e) = \frac{e^{-\frac{J(e)}{T}}}{Z(T)} = \frac{e^{-\frac{1}{T} \left(\frac{1}{2} \sum_{i \neq j} p_{ij} e_i e_j - \sum_i u_i e_i \right)}}{Z(T)} \quad (\text{voir énergie du réseau de HOPFIELD})$$

d'où :

$$\frac{\partial P_{\text{libre}}(e)}{\partial p_{ij}} = \frac{1}{T} \left[\sum_u P_{\text{libre}}(u, e) u_i e_j - \sum_v P_{\text{libre}}(v) \cdot \sum_{r, s} P_{\text{libre}}(r, s) r_i s_j \right]$$

$$\nabla(\partial G(P_{\text{contraint}}, P_{\text{libre}}); \partial p_{ij}) = \nabla(-1; T) \left[\nabla_{su}(u, v); \nabla(P_{\text{contraint}}(v); P_{\text{libre}}(v)) P_{\text{libre}}(u, v) u_i v_j - \sum_v P_{\text{contraint}}(v) \cdot \sum_{r, s} P_{\text{libre}}(r, s) r_i s_j \right]$$

$$\frac{\partial G(P_{\text{contraint}}, P_{\text{libre}})}{\partial p_{ij}} = \frac{-1}{T} \left[\sum_{u, v} P_{\text{contraint}}(u, v) u_i u_j - \sum_{r, s} P_{\text{libre}}(r, s) r_i r_j \right]$$

$$\frac{\partial G(P_{\text{contraint}}, P_{\text{libre}})}{\partial p_{ij}} = \frac{-1}{T} \left[P_{\text{contraint}}(e_i=1, e_j=1) - P_{\text{libre}}(e_i=1, e_j=1) \right]$$

On renforcera donc la liaison entre les unités numéros i et j si la probabilité en phase contrainte est supérieure à la probabilité en phase libre.

3.5.3.4 Les perspectives actuelles

Réseaux en couches unidirectionnels linéaires

Détermination d'une matrice P des influences, de moindre norme :

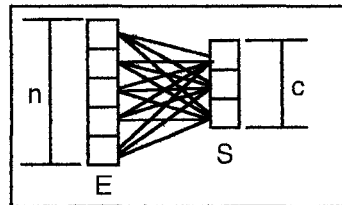
- par projection orthogonalisation :
Algorithme de GRAMM SCHMIDT
- par pseudo inverse de PENROSE :
Algorithme de GREVILLE
Algorithme de KALABA RASAKHOO
- par rétro propagation du gradient :
Algorithme de Le CUN

Le problème posé :

On dispose de p exemples clefs E_k et de p correspondants désirés D_k .

On veut une association linéaire : pour tout k de 1 à p : $D_k = A \cdot E_k$

E_k est codé sur n unités d'entrée, et D_k est codé sur c unités de sorties :



Notons $X = [E_1 , \dots , E_p]$ la matrice p colonnes et n lignes dont les colonnes sont les exemples.

Notons $Y = [D_1 , \dots , D_p]$ la matrice p colonnes et c lignes dont les colonnes sont les valeurs cibles associées aux clefs.

On veut trouver une matrice A telle que $Y = A \cdot X$ satisfaisant aux contraintes suivantes :

+ elle minimise la norme quadratique de A notée $J(A)$:

$$J(A) = \sum_{k=1}^p \| D_k - A \cdot E_k \|^2$$

+ elle minimise la norme euclidienne de A notée $E(A)$

$$E(A) = \sum_{i=1}^c \sum_{j=1}^n a_{ij}^2 = \text{tr}(A^T \cdot A)$$

(ce qui impose de ne pas mémoriser des associations parasites non voulues)

Sous ces contraintes, on prouve que la solution A est unique.

3.5.3.4.1 PROJECTION ORTHOGONALISATION (GRAMM SCHMIDT)

L'idée est d'orthogonaliser les exemples E_k .

Notons P_k la matrice de projection sur le sous espace engendré par les exemples de 1 à k.

Algorithme de génération des matrices P_k

$P_k = 0$: {matrice nulle}

Pour k de 1 à p {nombre d'exemples }

faire

$N_k = E_k - P_{k-1} \cdot E_k$ { N_k est l'innovation, la nouveauté introduite par l'exemple numéro k }

si $N_k \neq 0$

alors $P_k = P_{k-1} + (N_k \cdot N_k^T) / \|N_k\|^2$

sinon $P_k = P_{k-1}$ { pas de nouveauté }

Algorithme de génération des matrices A_k

$A_1 = (D_1 \cdot E_1^T) / \|E_1\|^2$

Pour k de 2 à p {nombre d'exemples }

faire

$S_k = A_{k-1} \cdot E_k$ { sortie obtenue pour l'exemple numéro k }

$R_k = D_k \cdot S_k$ { reliquat de D_k non expliqué par A_{k-1} }

$N_k = E_k - P_{k-1} \cdot E_k$ { N_k est l'innovation, la nouveauté introduite par l'exemple n° k }

}

si $N_k \neq 0$ { il y a une innovation introduite par E_k }

alors $P_k = P_{k-1} + (N_k \cdot N_k^T) / \|N_k\|^2$

$A_k = A_{k-1} + (R_k \cdot N_k^T) / \|N_k\|^2$ { on associe le reliquat à l'innovation }

sinon

si $R_k \neq 0$

alors { pas d'innovation N_k et on a un reliquat R_k de D_k non expliqué par A_{k-1} }

{ E_k est une combinaison linéaire des exemples de E_1 à E_{k-1} et D_k n'est }

{ pas la combinaison linéaire correspondante, on redistribue l'erreur }

{ dans la direction S_k }

$B_k^T = S_k^T \cdot A_{k-1} \cdot E_{k-1}$

$A_k = A_{k-1} + [\|S_k\|^2 / (\|S_k\|^4 + \|B_k\|^2)] \cdot R_k \cdot S_k^T \cdot A_{k-1}$

$P_k = P_{k-1}$

sinon { il n'y a pas d'innovation, ni de reliquat R_k non expliqué }

$A_k = A_{k-1}$

$P_k = P_{k-1}$

fait

3.5.3.4.2 CALCUL D'UNE PSEUDO INVERSE (PENROSE)

Solution au problème posé :

1. On "calcule" la matrice A par un procédé exact de calcul par la formule :

$$A = D \cdot E^+ \text{ où } E^+ \text{ est la matrice pseudo inverse de } E$$

2. On présente un exemple "e" à reconnaître.

3. On "calcule" le vecteur sortie "s" correspondant à $s = A \cdot e$

+ si "e" fait partie de l'ensemble d'apprentissage alors "e" sera parfaitement classifié par "s".

+ si "e" ne fait pas partie de l'ensemble d'apprentissage alors l'exemple "e" sera attribué à la classe qui correspond à la plus grande composante du vecteur de sortie "s".

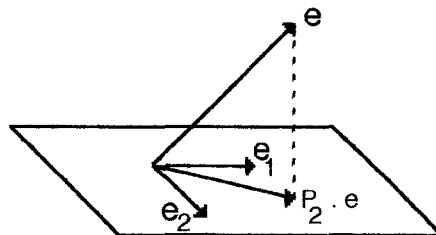
Interprétation du procédé solution

Pour mieux visualiser ce qui se passe, prenons le cas où le codage des exemples se fait sur 3 unités. On a alors un espace de dimension $n=3$.

L'ensemble d'apprentissage est composé de deux exemples e_1 et e_2 .

La sortie s est la projection orthogonale de l'entrée e sur le plan défini par les vecteurs exemples e_1 et e_2 .

En effet la matrice pseudo inverse s'interprète comme une matrice de projection orthogonale.



Théorie sur les matrices pseudo inverse

On ne considère que des matrices à coefficients réels (la théorie prend en compte les coefficients complexes)

Soit A une matrice n lignes et c colonnes

Définition :

A^+ est la matrice pseudo inverse de A **ssi** $A \cdot A^+ \cdot A = A$ et $A^+ \cdot A \cdot A^+ = A^+$ et ($A \cdot A^+$ et $A^+ \cdot A$ sont symétriques)

Conséquence :

si A est une matrice carrée non singulière **alors** $A^+ = A^{-1}$

Propriétés :

toute matrice A admet une pseudo inverse unique.

si les colonnes de A sont linéairement indépendantes **alors** $A^+ = (A^T A)^{-1} \cdot A^T$

si les lignes de A sont linéairement indépendantes **alors** $A^+ = A^T \cdot (A \cdot A^T)^{-1}$

si μ est un réel **alors** **si** $\mu \neq 0$ **alors** $\mu^+ = \mu^{-1}$ **sinon** $\mu^+ = 0$

si a est un vecteur **alors** **si** $a \neq 0$ **alors** $a^+ = \frac{a^T}{(a^T \cdot a)}$ **sinon** $a^+ = 0^T$

si $A = \text{diag} (a_1, a_2, \dots, a_n)$ **alors** $A^+ = \text{diag} (a_1^+, a_2^+, \dots, a_n^+)$

$$A^+ = \lim_{\mu \rightarrow 0} (A^T \cdot A + \mu^2 I)^{-1} \cdot A^T$$

si $S \cdot E^+ \cdot E = S$

alors l'équation $S = A \cdot E$ d'inconnue A admet toutes les solutions de la forme : $A = S \cdot E^+ + Q \cdot (I - E \cdot E^+)$ où Q est une matrice quelconque.

sinon l'équation $S = A \cdot E$ d'inconnue A admet les solutions approchées de la forme : $A = S \cdot E^+ + Q \cdot (I - E \cdot E^+)$ minimisant la norme quadratique de la matrice d'erreur $S - A \cdot E$ (Q est une matrice quelconque).

La solution approchée de plus petite norme est $A = S \cdot E^+$

Algorithmes de détermination d'une matrice pseudo inverse

1 . Algorithme de GREVILLE.

Soit à déterminer la matrice pseudo inverse d'un matrice notée A_n formée de n colonnes.

On note $A_k = [A_{k-1} | a_k]$ une matrice A_k

A_{k-1} est la matrice formée des k-1 premières colonnes de la matrice A_k

a_k est le vecteur colonne numéro k de la matrice A_k

initialisation : $A_1 = [a_1]$ c'est un vecteur

donc si $a_1 \neq 0$ alors $A_1^+ = a_1^T / (a_1^T a_1)$ sinon $A_1^+ = 0^T$

formule de récurrence :

si $\|(I - A_{k-1} \cdot A_{k-1}^+) a_k\|^2 \neq 0$

alors
$$P_k = \frac{(I - A_{k-1} \cdot A_{k-1}^+) a_k}{\|(I - A_{k-1} \cdot A_{k-1}^+) a_k\|^2}$$

sinon
$$P_k = \frac{(A_{k-1}^+)^T \cdot A_{k-1} a_k}{1 + \|A_{k-1}^+ a_k\|^2}$$

puis

$$A_k^+ = \begin{bmatrix} A_{k-1}^+ (I - a_k P_k^T) \\ \hline P_k^T \end{bmatrix}$$

2. Algorithme de KALABA RASAKHOO

Soit A une matrice de m lignes et de n colonnes de rang r

initialisation :

$P_1 = Id$ et $\delta_0 = \text{trace}(A^T \cdot A)$

formule de récurrence : pour k de 1 à (n-1)

$P_{k+1} = \delta_0 Id + A^T \cdot A \cdot P_k$

$\delta_{k+1} = \text{trace}(A^T \cdot A \cdot P_{k+1}) / (k+1)$

3.5.3.4.3 RÉTRO PROPAGATION DU GRADIENT (LE CUN)

Description du réseau multicouche:

Nom :

multi couches

Architecture :

UNITES :	
état interne X_i :	continu
fonctions de transition f_i :	sigmoïde
nombre d'unités :	noté : N

CONNEXIONS :	
influences :	unidirectionnelles
agencement :	en couches

Dynamique :

EN EXPLOITATION :	
	application des transitions

EN APPRENTISSAGE	
	GBP : rétro propagation du gradient

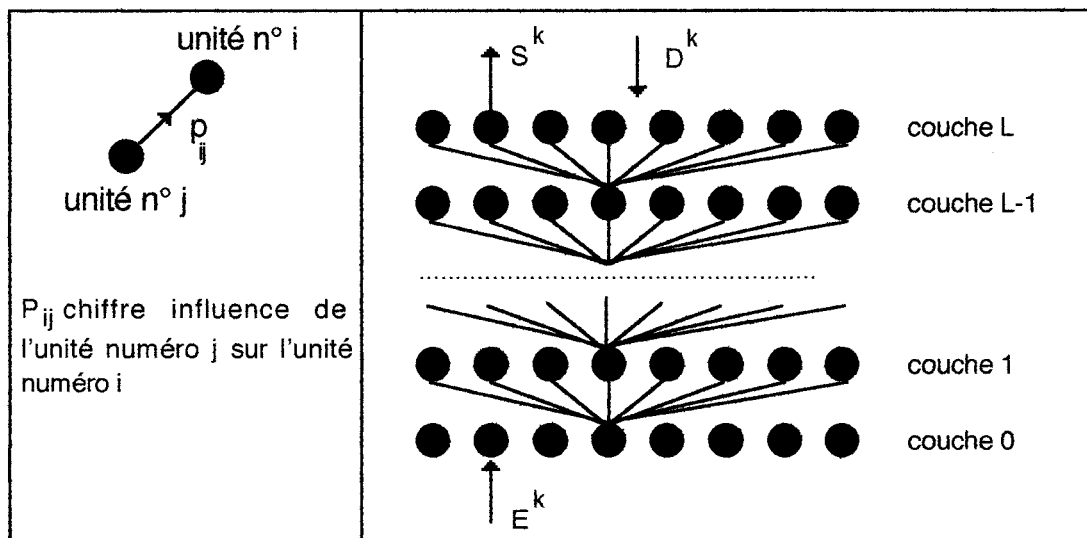
Inspiration :

méthode de gradient

Indicateur associé à un état du système :

fonction de coût cumul quadratique des erreurs minimalisée par gradient

Notations, schéma :



Le réseau est structuré en couches numérotée de 0 à L.
 Les flèches indiquent le sens de la propagation des états lors du calcul de la sortie du réseau.
 E^k est l'exemple numéro k présenté en entrée du réseau, S^k est la sortie calculée correspondante, D^k est la sortie désirée, $S^k - D^k$ est donc l'erreur qui sera rétro propagée dans le sens inverse des flèches

problème posé :

On dispose d'un ensemble de K associations d'une forme E^k avec une sortie désirée D^k . On a donc un ensemble de K couples (E^k, D^k) . Il s'agit d'apprendre au réseau les K associations avec une capacité de généralisation suffisante.

solution utilisant l'algorithme de rétro propagation du gradient.

On note e_i l'état de l'unité numéro i , f_i la fonction de transition sigmoïde de l'unité n° i . A_i l'influence des autres unités sur l'unité n° i : c'est la somme des $p_{ij} e_j$. Le nouvel état de l'unité n° i sera donc $f_i(A_i)$

La fonction d'erreur sera : $J(A) = \sum_{k=1}^K \|C^k - D^k\|^2$

La règle d'évolution des influences p_{ij} entre les unités j et i sera :

$$p_{ij}(t+1) = p_{ij}(t) - \mu \frac{\partial J}{\partial p_{ij}}$$

L'apprentissage serait alors un apprentissage "global" sur l'ensemble des K exemples. On va choisir une méthode de gradient décomposé pour pouvoir adapter les influences petit à petit sur chaque association entre un exemple et une sortie désirée.

On notera J^k l'erreur commise sur l'association concernant l'exemple numéro k :

$$J^k = \|S^k - D^k\|^2$$

La règle d'évolution sera alors : $p_{ij}(t+1) = p_{ij}(t) - \mu_t \frac{\partial J^k(t)}{\partial p_{ij}}$

μ_t est choisi comme un petit nombre réel positif qui chiffre l'intensité de l'apprentissage. Il évoluera comme une suite décroissante de réels positifs modélisant une intensité d'apprentissage de plus en plus faible.

Le problème de l'algorithme de rétro propagation du gradient provient de la non existence d'études théoriques affirmant la convergence de l'algorithme. Des choix appropriés d'architectures de réseau, d'ordres de présentation des exemples, et de suites d'intensités d'apprentissages μ ont permis d'atteindre effectivement un minimum souvent nul pour la fonction d'erreur J. Ces choix sont très empiriques.

Le calcul de $\frac{\partial J^k(t)}{\partial p_{ij}}$ s'effectue par $\frac{\partial J^k(t)}{\partial p_{ij}} = \frac{\partial J^k(t)}{\partial A_i} \cdot \frac{\partial A_i}{\partial p_{ij}} = y_i \cdot e_j$
 en notant $y_i = \frac{\partial J^k(t)}{\partial A_i}$

le calcul de $\frac{\partial J^k(t)}{\partial p_{ij}}$ se ramène donc au calcul du y_i associé à chaque unité numéro i du réseau.

Le calcul des y_i sera le suivant :

A. Pour les unités numéro i de la couche de sortie :

$$y_i = \frac{\partial J^k(t)}{\partial A_i} = \frac{\partial}{\partial A_i} \sum_h (S_h^k - D_h^k)^2 = 2 \cdot (S_i^k - D_i^k) \cdot f'(A_i)$$

La sommation est effectuée pour toutes les unités numérotées h de la couche de sortie.

B. Pour les unités numéro i qui sont pas sur la couche de sortie.

Notons y_q la dérivée partielle $\partial J^k / \partial A_q$ pour toutes les unités numéro q qui reçoivent une influence de l'unité numéro i. On aura :

$$y_i = \frac{\partial J^k(t)}{\partial A_i} = \sum_q \frac{\partial J^k(t)}{\partial A_q} \cdot \frac{\partial A_q}{\partial A_i} = \sum_q \frac{\partial J^k(t)}{\partial A_q} \cdot \frac{\partial}{\partial A_i} \left(\sum_j p_{qj} \cdot e_j \right)$$

$$y_i = \sum_q y_q \cdot p_{qi} \cdot \frac{\partial f_i(A_i)}{\partial A_i} = f'_i(A_i) \sum_q y_q \cdot p_{qi}$$

Algorithme GBP :

1. Calcul de la sortie C correspondant à l'entrée E :

de la couche numéro zéro à la couche numéro L :

si l'unité numéro i est en couche 0

alors $e_j = E_j$

sinon $e_j = f_i(A_i)$ avec $A_i = \sum_j p_{ij} \cdot e_j$

2. Calcul des y_i pour chaque unité numéro i

de la couche numéro L à la couche numéro un :

si l'unité numéro i est en couche L (de sortie)

alors $y_i = 2 \cdot (S_i - D_i) \cdot f'_i(A_i)$

sinon $y_i = f'_i(A_i) \cdot \sum_j p_{ij} \cdot y_j$

3. Adaptation des influences p_{ij} .

$$p_{ij}(t+1) = p_{ij}(t) - \mu(t) \cdot y_j \cdot e_j$$

$\mu(t)$ est une suite de réels positifs convergeant vers zéro ou un très petit réel positif.

Réseaux récurrents bidirectionnels

Mise en équilibre des unités en vue de satisfaire un objectif avec un minimum de remise en cause

3.5.3.4.4 RÉSEAUX RÉCURRENTS DÉTERMINISTES BIDIRECTIONNELS (GUIVARCH)

Description :

Nom :

réseau récurrent bidirectionnel

Architecture :

UNITES :	
unités d'entrée rétiné :	<i>présentation des exemples</i>

unités cachées :	<i>modélisent la mémoire à court terme du système</i>
état interne Xi:	continu ou discret
fonctions de transition fi :	semi linéaire
nombre d'unités :	noté : NC

unité de sortie :	<i>fournissent le résultat visible de l'activité</i>
état interne Xi:	continu ou discret
fonctions de transition fi :	semi linéaire ou spéciale
nombre d'unités :	noté : NS

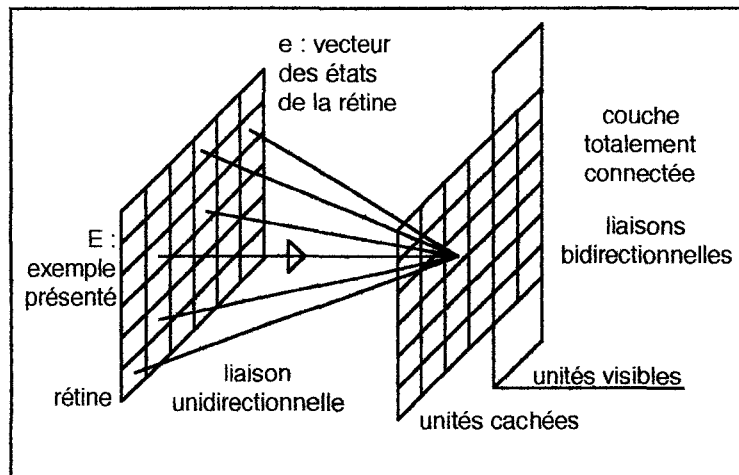
CONNEXIONS :	
influences :	bidirectionnelles (couche cachée <-> sortie)
agencement :	une couche d'entrée vers les unités cachées et de sortie

Les principes d'exploitation ou d'apprentissage :

+ s'adaptent aussi bien

- à des unités prenant leurs valeurs dans un domaine continu que discret
- à la satisfaction d'une égalité à une valeur cible en sortie
- à une maximalisation ou minimalisation de valeurs cibles inconnues
- à des sorties témoins du respect d'une inégalité.

Notations :



$e_{i,t}$ état d'activation de l'unité numéro i à l'instant t ; e_{v_i} état voulu pour l'unité numéro i

$$u_{i,t} = \sum_k p_{ikt} \cdot e_{kt} \text{ bilan des influences reçues par l'unité numéro } i$$

$$f_i(u_{i,t}) = e_{i,t+1} \text{ fonction de réponse de l'unité numéro } i$$

Dynamique :

EN EXPLOITATION :	Mise en équilibre des unités avec ou sans contexte
EN APPRENTISSAGE	Dynamique de GUIVARCH

Dynamique d'apprentissage :

Réseaux prenant leurs valeurs dans un domaine continu, soumis à un apprentissage supervisé portant sur des cas exemplaires, modifiant les influences locales entre les unités pour imposer une sortie ayant une valeur cible bien définie en réponse à une sollicitation d'entrée.

- Un objectif double :**
1. Avoir toutes les unités en équilibre
 2. Avoir les sorties au niveau désiré

Un indicateur : Une distance par rapport à l'objectif

Une dynamique : Répartir itérativement déséquilibres et erreurs par coopération entre les unités

Indicateurs d'écart à l'objectif :

pour les unités cachées : le déséquilibre courant : $\Delta e_{i,t} = e_{i,t} - f_i(u_{i,t})$

pour les unités de sortie : l'erreur par rapport au niveau désiré :

$$\Delta e_{i,t} = e_{v_i} - f_i(u_{i,t})$$

Pénalité de distance :

Cumul quadratique des indicateurs pour chaque unité.

Trois formes équivalentes pour une pénalité $C_{i,t}$ pour une unité n° i :

forme 1 :

$$2.C_{i,t} = \frac{(e_i - f_i)^2 + 2 \cdot \mu_i \cdot a_i (e_{vi} - f_i) + a_i^2 \cdot (e_{vi} - f_i)^2}{1 + 2 \cdot \mu_i \cdot a_i + a_i^2}$$

avec $\mu_i^2 < 1$ qui garantit une forme quadratique définie positive

forme 2 :

$$2.C_{i,t} = \frac{(e_i - e_{ei})^2 + a_i^2 \cdot (1 - \mu_i)^2 \cdot (e_{vi} - f_i)^2}{1 + 2 \cdot \mu_i \cdot a_i + a_i^2}$$

en posant : $e_{ei} = f_i + \mu_i \cdot a_i (f_i - e_{vi})$

forme 3 :

$$2.C_{i,t} = (f_i - e_{ri})^2 + a_i^2 \cdot (1 - \mu_i)^2 \cdot \frac{(e_i - e_{vi})^2}{(1 + 2 \cdot \mu_i \cdot a_i + a_i^2)}$$

en posant : $e_{ri} = \frac{(1 + \mu_i \cdot a_i) \cdot e_i + a_i \cdot (a_i + \mu_i) \cdot e_{vi}}{1 + 2 \cdot \mu_i \cdot a_i + a_i^2}$

Pour une unité cachée, μ_i et a_i sont nuls.

$e_i - e_{ei}$ est l'excès d'activation de l'unité numéro i

$e_{ri} - f_i$ est le déficit de stimulation de l'unité numéro i

L'apprentissage d'un exemple de référence consiste à réduire le cumul des pénalités de chaque unité selon l'importance π_i qu'on leur accorde : $C(t) = \sum_i (\pi_i C_i(t))$. Pour tout i $\pi_i > 0$

Conduite dynamique de l'apprentissage :

1ère phase de descente de C(t)

On agit sur les niveaux d'activation des unités sans modifier les valeurs p_{jk} des influences.

Les unités cachées conservent un déséquilibre résiduel.

Les unités de sortie conservent un déséquilibre et une erreur résiduels.

Stratégie :

On abaisse C(t) en agissant sur les activations de manière coopérative entre les unités candidates à une modification. La modification d'une unité n'est pas égoïste, elle se préoccupe des conséquences de son évolution sur les unités qui sont sous son influence.

Dans l'hypothèse d'une descente en gradient de C(t), on a :

$$e_{i,t+1} - e_{i,t} = -\beta_i \frac{\partial C(t)}{\partial e_{i,t}} = \underbrace{-\beta_i \cdot \left[\pi_i \frac{e_i - e_{ei}}{1 + 2 \cdot \mu_i \cdot a_i + a_i^2} \right]}_{\substack{\text{Marche directe} \\ \text{vers l'équilibre} \\ \text{S'AIDER SOI MEME}}} + \underbrace{\sum_{j \neq i} \pi_j (f_j - e_{rj}) \frac{\partial f_j}{\partial e_i}}_{\substack{\text{Satisfaction des unités qui} \\ \text{aspirent à faire } f_j = e_{rj} \\ \text{AIDER LES AUTRES}}}$$

Au terme de la descente en gradient, on aura un gradient nul donc :

$$\pi_i \frac{e_i - e_{ei}}{1 + 2 \cdot \mu_i \cdot a_i + a_i^2} = \sum_{j \neq i} \pi_j (f_j - e_{ij}) \frac{\partial f_j}{\partial e_i}$$

L'excès d'activation
accepté par l'unité n° i,
compte tenu de son
importance π_i
= La somme des déficits
de stimulation des
unités influencées par
l'unité n° i compte tenu
de leurs sensibilités
aux influences π_j

Les paramètres μ_i et a_i contrôlent la manière dont les unités de sortie dérivent vers leur niveau cible :

si $\mu_i = 0$ alors on a $e_{ei} = f_i$ c'est à dire que l'unité n° i ne prend pas en charge sa mise en équilibre, elle rétro propage une demande de poussée des autres plus ou moins intense selon la valeur de $|a_i|$. Elle demande aux autres de la mettre en équilibre.

si $(1 + \mu_i \cdot a_i) = 0$ et $a_i > 1$ alors l'unité se fixe sur son niveau d'équilibre et rétro propage une demande d'adaptation. Elle fixe son exigence d'équilibre et demande aux autres de s'adapter à son bon vouloir.

Soit f le temps final de la descente de la première phase.

2ème phase de descente de $C(t)$:

On adapte les influences p_{ij} . Ces influences pondèrent la transmission de deux quantités :

En sens direct : l'activation e_j de l'unité n° j influençant l'unité n° i.

En sens rétrograde : la demande rétrograde de i vers j : $(\mu_i \cdot (e_{ei} - f_i) \cdot f'_i(u_i))$

Une variation dp_{ij} n'agit sur $C(f)$ que par ces deux quantités.

Dans le sens direct, la sensibilité de $C(f)$ est donnée par :

$$\frac{\partial C_i(f)}{\partial p_{ij}} = \frac{\partial C_i(f)}{\partial f_i} \frac{\partial f_i}{\partial p_{ij}} = - (e_{ei} - f_i) f'_i e_j$$

Dans le sens rétrograde, la sensibilité de $C(f)$ est donnée par :

$$\frac{\partial C_j(f)}{\partial p_{ij}} = \frac{\partial C_j(f)}{\partial (e_j - e_{ej})} \frac{\partial (e_j - e_{ej})}{\partial p_{ij}}$$

La dérivée partielle du premier facteur est extraite de la seconde forme de $C(f)$, la dérivée partielle du second facteur est obtenue à partir de l'égalité manifestant la fin de la descente en gradient.

La correction de p_{ij} s'effectue classiquement à l'opposé du gradient et on a :

$$p_{ij,t+1} = p_{ij,t} + s_{ij} \cdot (e_{ei} - f_i) \cdot f'_i \cdot e_{ej}$$

Réseaux organisés en colonnes et en aires
Modélisation selon les travaux de BURNOD

3.5.3.4.5 RÉSEAUX ORGANISÉS EN COLONNES ET EN AIRES (BURNOD)

Présentation :

Le système modélise une colonne corticale, architecture modulaire du cortex humain qui intègre le fonctionnement d'une centaine de neurones.

Architecture :

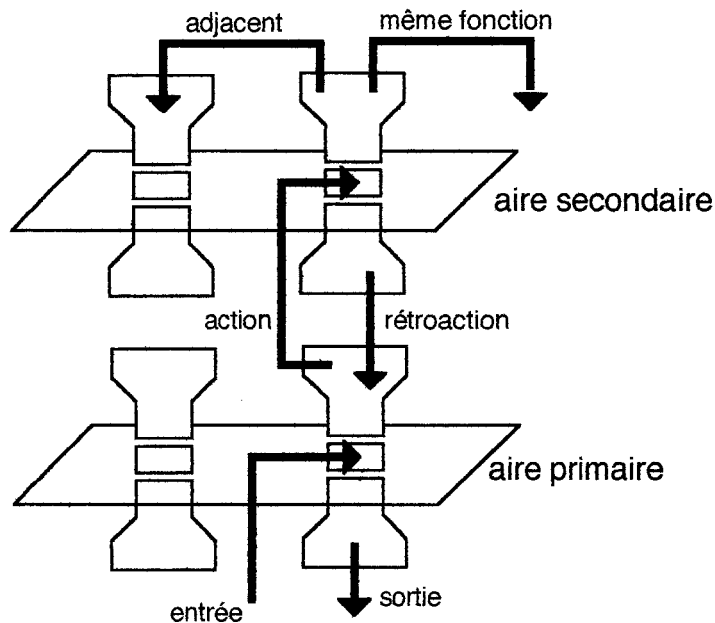
Une colonne corticale possède six couches, regroupées en trois dans la modélisation.

Couche haute (1 à 3 du cortex) : elle reçoit les entrées corticales et effectue une sortie vers les autres régions corticales.

Couche médiane (4 du cortex) : elle reçoit les stimuli extérieurs au cortex ou provenant de l'aire primaire.

Couche basse : (5 et 6 du cortex) : elle effectue des sorties vers les structures cibles.

Précisons par un schéma cette architecture :



États d'activation d'une colonne :

Ils sont ternaires : notés E_0 , E_1 et E_2 .

État E_0 : activité très faible

(phase d'inhibition)

État E_1 : couche haute activée moyennement : recherche d'un état stable

(phase d'attention)

État E_2 : effet d'avalanche déclenché. Les couches basses et hautes sont fortement actives.

L'état stable est atteint

(phase d'action)

Fonctionnement : mise en forme et transmission de l'information :

Activité forte E_2 d'une colonne :

Correspond à une comparaison de l'activité de colonnes connectées. Les différences d'état seront amplifiées, d'où :

+ inhibition des colonnes connectées d'état E_1

+ réactivation des colonnes connectées d'état E_2 et modifications synaptiques.

Activité thalamique :

Elle est le résultat d'une fonction de transfert spatiale calculée localement sur un champ récepteur par des inter neurones périphériques inhibiteurs et des centraux excitateurs.

Sur une colonne d'une aire sensorielle où la couche médiane est très développée, elle aura un effet d'activation forte E_2 de toute la colonne.

Sur une colonne d'une aire motrice, elle n'aura pas d'effet.

Sur une colonne d'une aire associative, il y aura passage en état E_1 de la colonne.

Activité thalamique plus activité d'attention E_1 d'une colonne.

Il y aura amplification du phénomène. La colonne passe en état E_2

Apprentissage et évolution:

Règles d'apprentissage :

Elles mémorisent les similitudes et les différences de fonctionnement d'une colonne avec ses voisines : couplage, découplage entre colonnes.

+ Deux colonnes toujours activées ensemble vont mémoriser une relation privilégiée qui induira un déclenchement inconditionnel quand l'une est à E_1 et l'autre à E_0 .

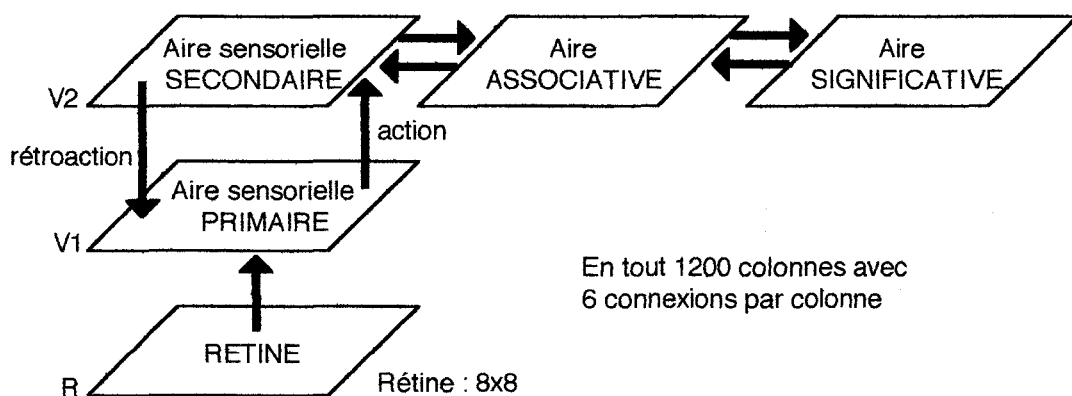
+ Deux colonnes jamais coactives vont mémoriser une relation d'inhibition.

+ Deux colonnes parfois en états (E_2 et non E_2) vont mémoriser une relation de mise en attention E_1

Règles de propagation :

Le réseau évolue à partir d'une situation thalamique donnée (monde extérieur) et de buts corticaux recherchés (monde intérieur) pour produire des actions thalamiques favorisant la résolution du problème.

Exemple de réseau :



Une fonction de transfert thalamique effectue une détection de segments d'orientation variable sur des champs récepteurs se recouvrant partiellement.:



AIRE V1 : Elle distribue l'information à sa surface de manière rétinotopique et code les variations d'orientation par rapport à la rétine.

AIRE V2 : même type de traitement.

Cette cascade de détections locales fait apparaître des ensembles de colonnes en déclenchement inconditionnel représentant les caractéristiques invariantes du monde extérieur.

AIRE SIGNIFIANTE : Les colonnes codent des noms ou des concepts "ceci est un rond" ou "ceci est un carré".

Apprentissage :

Toutes les colonnes de l'aire signifiante sont à E_0 sauf la colonne concept à reconnaître mise à E_2 . On laisse le système évoluer pour mémoriser les relations entre les colonnes.

Reconnaissance :

Toutes les colonnes de l'aire signifiante sont à E_1 (état d'attention). On présente une forme sur la rétine. La solution est trouvée lorsqu'une colonne de l'aire signifiante a atteint un état stable E_2 .

<p>Réseaux auto adaptatifs Modélisation selon les travaux de KOHONEN</p>

3.5.3.4.6 RÉSEAUX AUTO ADAPTATIFS (KOHONEN)

Description :

Nom :

réseau auto adaptatif de KOHONEN

Architecture :

UNITES :	
unités d'entrée rétiné :	<i>présentation des exemples</i>
unité de sortie :	<i>fournissent le résultat visible de l'activité</i>
état interne Xi:	Positif
fonctions de transition fi :	sigmoïde
nombre d'unités :	noté : NS

CONNEXIONS :	
influences :	unidirectionnelles de la couche d'entrée vers la couche de sortie bidirectionnelles pour la couche de sortie
agencement :	une couche d'entrée vers les unités de sortie

Dynamique :

EN EXPLOITATION :	
	elle est modélisée par l'équation différentielle : $\frac{dS}{dt} = f(E, S, M, N)$

EN APPRENTISSAGE	
des connexions externes rétiné – réseau	elle est modélisée par : $\frac{dM}{dt} = g(E, S, M)$
des connexions internes réseau lui-même	elle est modélisée par : $\frac{dN}{dt} = h(S, M)$

Pour les cartes adaptatrices, il n'y a pas d'apprentissage des connexions internes (on verra plus loin qu'elles sont la discrétisation d'un "chapeau mexicain")

Notations :

Soit e l'état d'une unité de la couche S de sortie totalement connectée.

Équations d'état d'une unité :

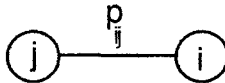
$$\frac{de}{dt} = u - p(e) \quad u \rightarrow \boxed{} \rightarrow e \quad p(e) \text{ est une perte non linéaire rendant compte du phénomène de saturation}$$

unité stabilisée $\frac{de}{dt} = 0 \Leftrightarrow u = p(e) \Leftrightarrow e = p^{-1}(u) \quad p^{-1}$ fonction sigmoïde

Évolution des influences locales :

$$\frac{dp_{ij}}{dt} = K \cdot e_i \cdot e_j - \text{oub}(e_i) \cdot p_{ij}$$

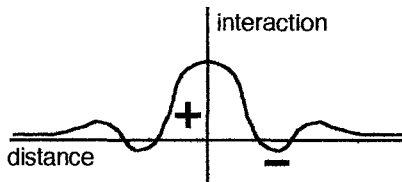
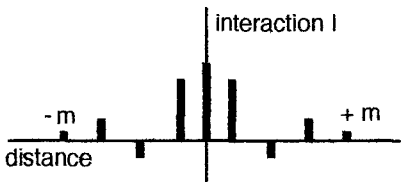
\uparrow phénomène d'attention = intensité d'apprentissage
 \uparrow phénomène d'oubli
 pour tout $x : \text{oub}(x) \geq 0$



Inspiration :

Système de perception (vue, ouïe) des mammifères, codant sur des unités voisines des sensations semblables.

Modélisation :

Inspiration particulière	Réalisation correspondante
Dans un système de perception, certaines cellules nerveuses ont des rôles bien identifiables. Par exemple, dans le système visuel du chat, certains neurones réagissent à la présence d'un trait incliné dans une partie spécifique du champ visuel.	Réseau à réponses localisées : Chaque unité du réseau correspond à une stimulation spécifique.
Dans un système de perception, on identifie un mécanisme d'interaction latérale dépendant de la distance entre les neurones de type chapeau mexicain : 	Réseau en amas d'unités. Le réseau s'organise en amas d'unités excitées dans un voisinage de l'unité la plus active.  <p>I_k est une discrétisation du chapeau mexicain limitée à un voisinage $[-m, +m]$</p>

Règle d'évolution de l'état e_i de l'unité numéro i :

$$e_i(t+1) = \text{sigmoïde} \left(\underbrace{\sum_{j \text{ de la rétine}} p_{ij} \cdot e_j(t)}_{\text{influence de la rétine}} + \sum_{k=-m}^{+m} I_k e_{i+k}(t) \right)$$

Dynamique d'apprentissage :

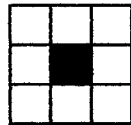
Elle va porter uniquement sur les connexions externes

1. Sélection de l'unité qui réagit le plus au type de signal d'entrée

Les vecteurs ligne de la matrice M peuvent être interprétés comme une image de l'unité numéro i dans l'espace des signaux d'entrée

	<p>L'unité numéro C réagissant le plus à une entrée E donnée sera donc celle qui a une image $M.C$ la mieux "corrélée" à E. Il suffit donc de trouver l'unité numéro C telle que :</p> $\ E - M.C\ = \min_i \ E - M.i\ $
--	--

2. Augmenter l'activation de cette unité et des unités du groupe qui l'entoure



Voisinage V_C de l'unité $n^\circ c$
 Ce voisinage est assez large au début pour se rétrécir après

si j est dans V_C
 alors $e_j = 1$
 sinon $e_j = 0$

On dispose d'une fonction d'oubli notée OUB telle que
 $OUB(1) = k(t)$ et $OUB(0) = 0$

si i est dans V_C alors $\frac{dp_{ij}}{dt} = OUB(e_i) \cdot [e_i - p_{ij}]$

Réseaux simpliciaux

Modélisation selon les travaux de A. COSNES

3.5.3.4.7 RÉSEAUX SIMPLICIAUX (COSNES)

Présentation :

Dans Matière à Penser (Éditions Odile JACOB Paris 89) Jean Pierre CHANGEUX et Alain COSNES proposent une modélisation de certains phénomènes cognitifs par des complexes simpliciaux.

Un complexe simplicial est un ensemble fini de points appelés sommets, chaque sommet représente un neurone pris en un agrégat assez complexe.

Un complexe simplicial possède une dimension notée N .

Le complexe simplicial est structuré par N ensembles appelé "Delta 1", ..., "Delta N ".

"Delta 1" est un sous ensemble de l'ensemble des couples de sommets. Les éléments de "delta 1" seront appelés arêtes, chaque arête représentant une connexion entre deux neurones.

Plus généralement "Delta N " est un sous ensemble de l'ensemble des $(N+1)$ uples de sommets.

Ainsi "Delta 2" peut être interprété comme le sous ensemble des triangles.

Les "Delta i " doivent bien sûr être compatibles entre eux.

Ainsi si ABC est un triangle de "Delta 2" alors AB, BC et AC doivent être des arêtes de "Delta 1".

La topologie simpliciale a pour objet l'étude des invariants topologiques des objets "complexes simpliciaux".

Elle pourrait ainsi être un moyen pour coder la notion de forme privée de la partie quantitative de sa géométrie dans un réseau neuronal.

En ce qui concerne la mémoire à long terme, J.P. CHANGEUX et A. COSNES proposent une modélisation par complexes simpliciaux hyperboliques.

La condition pour qu'un complexe simplicial de dimension 2 soit hyperbolique est que tout sommet d'un triangle soit commun à sept triangles différents.

Un complexe simplicial hyperbolique généralise la notion d'arbre.

3.5.4. Exemples de machines connexionnistes

1. Des réseaux pour diagnostiquer
2. Des réseaux pour lire un texte écrit
3. Des réseaux pour apparier des images
4. Des réseaux pour restaurer une image bruitée sous la forme la plus probable
5. Des réseaux pour prévoir un moment optimal

3.5.4.1 DIAGNOSTIQUER, systèmes experts et machines connexionnistes

Les systèmes experts connaissent actuellement deux goulots d'étranglement :

– le goulot de VON NEUMANN, qui tient à la lenteur des méthodes utilisées, séquentielles et difficilement parallélisables, donnant lieu souvent à une explosion combinatoire lorsque les données à traiter sont trop nombreuses.

– le goulot de FEIGENBAUM, qui tient au fait que les connaissances du système doivent être données de façon explicite par le programmeur et qu'il n'existe aucune possibilité d'apprentissage à partir d'exemples, de plus les connaissances d'un expert sont souvent difficilement formalisables.

exemple : [D'après Y. LE CUN]

Problème :

Apprendre à un réseau à produire automatiquement un diagnostic de douleurs abdominales. La base de données fournie par le service de J.F. BOISVIEUX, Hôpital de la pitié, comporte 5700 cas. Chaque cas est décrit par 132 signes (symptômes) partiellement remplis et un diagnostic parmi 22 possibles. Les diagnostics sont les suivants, entre parenthèses le pourcentage de ce diagnostic dans la base de données.

Affection pulmonaire (0,31)	Anévrisme de l'aorte (0,45)	Appendicite (26,84)
Cancer sans occlusion (0,92)	Cholécistite (10,72)	Colique néphrétique (3,40)
Douleur gynéco-fonctionnelle (0,64)	Douleur non spécifique (24,19)	Fausse couche (0,17)
Foyer suppuré (0,31)	Grossesse extra-utérine (0,95)	Hernie étranglée (0,31)
Infarctus du mésentère (1,06)	Infection urinaire (1,46)	Kyste ovarien (1,51)
Occlusion org. primitive (8,34)	Pancréatite (3,5)	Perforation ulcère (3,87)
Péritonite primitive ou intestinale (1,39)	Poussée d'ulcère (2,34)	Salpingite (2,48)
Sigmoïdite (2,13)		

Plus de la moitié de ces cas sont diagnostiqués comme appendicite ou douleur non spécifique. L'appendicite constitue un diagnostic très grave à ne pas porter, et les médecins préfèrent le porter à tort que de manquer un vrai cas.

Solution :

Un réseau formé de quatre couches :

+ une couche d'entrée de 200 unités codant les données représentant les symptômes
+ une première couche cachée de 100 unités connectées chacune à 40 unités de la précédente couche.

+ une seconde couche cachée de 50 unités connectées chacune à 40 unités de la précédente couche.

Les unités de ces couches cachées peuvent être interprétées comme codant des diagnostics intermédiaires ou syndromes.

+ une couche de sortie de 22 unités représentant les diagnostics chaque unité étant connectée aux 50 unités de la couche précédente.

Les liens sont interprétables en termes de causalité. Par exemple l'unité de décision "Appendicite" sera reliée à quatre unités syndromes : "inflammation, inflammation digestive, péritonite, occlusion".

Résultats :

Ce réseau obtient des performances de 70% de réussite en apprentissage et de 60% en généralisation. 4000 cas ont été utilisés pour l'apprentissage à partir d'exemples, les 1700 restant pour la généralisation. L'unité diagnostic s'interprète comme une sorte de probabilité de diagnostic. Un système expert donne un taux de succès de 60%. Un tel réseau fonctionne donc comme un système expert avec des performances comparables, mais il est actuellement incapable d'argumenter sa décision. La faiblesse de la généralisation provient sans doute du nombre trop élevé de connexions (7100).

3.5.4.2 RECONNAITRE, lire un texte écrit

exemple : [D'après T. SEJNOWSKI]

Problème :

Apprendre un réseau à convertir du texte anglais en parole.

La difficulté de ce problème tient aux règles de prononciation et aux nombreuses exceptions.

Solution :

Le réseau est organisé en 309 unités organisées en trois couches :

+ une couche d'entrée qui examine 7 caractères du texte à la fois pour tenir compte du contexte. Soit 203 unités organisées en 7 groupes de 29 unités codant les caractères.

+ une couche de 80 unités cachées

+ une couche de 26 unités de sortie codant les phonèmes

Soit au total 18 629 connexions modifiables par l'algorithme de rétro propagation du gradient, par comparaison des caractères en entrée aux phonèmes désirés en sortie.

Résultats :

Après une nuit d'apprentissage, soit la présentation d'environ 50 000 mots environ, le programme "prononçait" correctement 95% des mots d'un texte d'apprentissage et 90% des mots d'un texte nouveau.

La connexion en sortie d'un synthétiseur des phonèmes, produisait des paroles évoquant les légères erreurs d'un enfant débutant.

L'étude des unités cachées a permis de mettre en évidence des phénomènes d'auto organisation du réseau en apprentissage.

3.5.4.3 ASSOCIER, appairer des images

exemple : [d'après KELLER et FOGELMAN SOULIE]

Problème :

Appairer des images de gels d'électrophorèse permettant l'analyse de la composition en protéines d'un liquide.

L'image est formée de "spots" dont la position et l'intensité sont caractéristiques d'une protéine et de sa concentration.

Solution :

Un réseau de KOHONEN type "cartes topologiques"

Ce réseau est progressivement adapté à l'image d'une gel et sert de référence pour l'appariement de gels ultérieurs.

3.5.4.4 RESTAURER l'image la plus probable, à partir d'une forme bruitée.

exemple : [d'après S. et D. GEMAN]

Problème :

Restaurer l'image la plus probable, à partir d'une donnée bruitée.

Solution :

Un réseau complètement connecté de 64 sur 64 unités, codant les pixels d'une image. L'image la plus probable correspondant à une distribution de probabilité optimisée, modélisable par une technique de recuit simulé. On part d'une image initiale et on modifie les pixels et contours par des itérations successives suivant une règle de Monte CARLO. La température décroît régulièrement jusqu'à ce que l'image restaurée soit satisfaisante.

Résultats :

Les images restaurées sont d' "assez bonne" qualité, même avec un bruitage conséquent.

3.5.4.5 PREVOIR un moment optimal

exemple : [d'après BAILEY, THOMPSON, FEINSTEIN]

Problème :

Prédire le meilleur moment pour utiliser des options d'achat sur des actions en Bourse, à partir des mouvements des prix.

Solution :

Un réseau organisé en trois couches d'unités continues entre -0,5 et 0,5:
+ une couche d'entrée de 12 unités codant des facteurs comme le prix de l'option d'achat, la volatilité et le prix de l'action, le volume échangé, le prix d'exercice, le temps d'expiration, les taux d'intérêt, ...
+ une couche d'unités cachées de 9 unités
+ une couche de sortie de 2 unités codant pour l'une la tendance attendue et pour l'autre une prévision du prix de l'option d'achat
Apprentissage par rétro propagation du gradient chaque exemple étant présenté 1000 fois

Résultats :

Sur les exemples appris : donne 90% de réponses correctes
Sur des exemples non appris donne 80% de réponses correctes.
Les résultats obtenus sont un peu meilleurs qu'avec les méthodes statistiques

3.5.5. Conclusion sur les modèles connexionnistes

Dans un système classique, les données sont stockées dans des cases numérotées bien identifiées, le calcul consiste à transférer les contenus des boîtes X et Y supposées représenter des opérandes et d'une boîte F supposée représenter une instruction, dans un processeur P qui va produire un résultat $R = F(X, Y)$ qui sera transféré dans une boîte T. Il y a dans ce processus modification de l'état électrique d'un certain nombre de transistors. Les fonctions de stockage (matériel) et de manipulation (logiciel) sont nettement séparées. Le calcul binaire n'a aucune résistance à la perturbation ou au flou. Le programme fonctionne sur un jeu de données typées et bien spécifiées selon une procédure analysable en un petit nombre fini d'étapes, traitant un problème bien posé, incapable d'initiative spontanée.

Dans un système expert, la représentation des connaissances est localisée. Un atome de connaissance est représenté par un objet précis (une règle de réécriture, un fait, un octet en mémoire). Les systèmes experts tendent de lever l'une des contraintes des systèmes classiques en séparant faits et règles, un système de déductions ou d'inférences logiques tirant les conséquences ou les inférences qui s'imposent. Le problème doit rester bien posé et s'exprimer en un nombre fini de faits et règles. Les caractéristiques du monde extérieur restent énormité, ambiguïté, variabilité. Nous sommes tous experts par exemple en reconnaissance d'images. Instantanément nous reconnaissons un visage ami. Sommes nous capables d'exprimer les "règles" et "faits" de cette reconnaissance?

Dans un système connexionniste, l'approche est délibérément orientée vers le traitement parallèle et distribué des connaissances, elle repose sur l'émergence de concepts et de comportements adaptés à un domaine donné, à partir d'assemblées d'unités primitives en interactions locales simples.

CERTES

- C'est un système difficile à construire. On trouve encore peu de "chips" implantant un réseau.
- C'est un système qui n'est utilisé pour l'instant que dans des applications restant de dimension modeste, ne permettant pas d'expliquer les résultats fournis, n'apportant pas de grandes avancées par rapport aux méthodes classiques.

MAIS

- C'est un système adaptatif, car il possède une capacité d'apprentissage lui permettant de tenir compte de nouvelles contraintes, certains réseaux possèdent même une capacité d'auto organisation.
- C'est un système dans lequel la représentation de l'information est distribuée, chaque cellule peut ne représenter qu'une caractéristique partielle d'une information et peut participer à la représentation de plusieurs entités d'information.
- C'est un système facile à simuler : en le programmant pour une petite application simple ou en utilisant un simulateur et une carte adaptatrice dans le cas contraire.
- C'est un système parallèle, son architecture étant conçue comme un ensemble d'unités fonctionnant simultanément.
- C'est un système capable de généralisation, propriété émergente apparaissant comme conséquence de l'apprentissage.

BIBLIOGRAPHIE

vision :

- [BALL 82] BALL D.H., BROWN C.M.
" Computer Vision "
Prentice-Hall inc. 1982.
- [BERT 90] BERTRAND G.
" Voisinages, dilatations, distances "
Publication Laboratoire Intelligence Artificielle et Analyse Images ESIEE 1990.
- [CELEU 89] CELEUX G, DIDAY E, GOVAERT G, LECHEVALLIER Y,
RALAMBONDRAINY H
**" Classification automatique des données : environnement statistique et
informatique "**
DUNOD 1989.
- [DIXO 86] DIXON M.J., GEGORY P.J.
" Hybrid expert system in image analysis "
SPIE Proceedings "Applications of Artificial Intelligence IV" , vol 657:9-12. 1986.
- [FIOR 89] FIOROT J.C., JEANNIN P.
" Courbes et surfaces rationnelles. Application à la C.A.O. "
MASSON 1989.
- [GALL 88] GALLESIO E., SERFATI V., Boi L., ZAVIDOVIQUE B.
" An object to capture some fuzzy expertise "
SPIE Proceedings "Applications of Artificial Intelligence IV" , vol 657:26-33. 1986.
- [GARB 87] GARBAY C.
**" Quelques propositions pour la réalisation d'un système expert de
segmentation d'images "**
TS, 4 (3) 229-238 1987.
- [HALA 85] HALARIC R.M., SCHAPIRO L.G.
" Images segmentation techniques "
CVGIP, vol 29:100-132. 1985.
- [HANS 78] HANSON A.R., RISEMAN E.M..
" Vision : a computer system for interpreting scenes "
Computer vision systems pp 303-333, Academic press 1978
- [HAYE 85] HAYES ROTH B.
" A blackboard model of control "
Artificial Intelligence, vol 26, pp 251-321 1985.
- [KOHL 87] KOHL C.A., HANSON A.R., RISEMAN E.M..
" A goal directed intermediate level executive for image interpretation "
Proceedings of the tenth International Joint Conference on Artificial Intelligence
vol2 pp 811-814 IEEE Computer Society Press 1987.
- [LUX 85] LUX A.
" Algorithmique et contrôle en vision par ordinateur "
Thèse d'État, Université Joseph FOURIER et INPG (Grenoble) 1985.
- [NAZI 84] NAZIF A.M., LEVINE M.D.
" Low level image segmentation : an expert system "
IEEE Trans on PAMI-6 : 555-577 1984.

- [PAIR 88]..... PAIR C., MOHR R., SCHOTT R.
" Construire les algorithmes : les améliorer, les connaître, les évaluer"
DUNOD 1988.
- [PAVL 82]..... PAVLIDIS T.
" Algorithms for Graphics and, image Processing"
Springer-Verlag Berlin Heidelberg New York 1982.
- [PORQ 86]..... PORQUET C.
" Le metalangage AIRELLE comme outil de développement de maquettes d'interpréteurs d'images"
Thèse de doctorat Université de Caen 1986.
- [MARR 82]..... MARR D.
" Vision"
W.H. FREEMAN and Company New York 1982 ISBN 0-7167-1567-8
- [MATS 88]..... MATSUYAMA T.
" Expert system for image processing knowledge based composition of image analysis processes"
Proceedings of the 9 international conference on pattern recognition pp125-133
IEEE
- [MOHR 88] MOHR R.
" Sur l'appariement Modèle - Perception"
Actes du 2nd Atelier scientifique TIPI chap XXIX (CNRS, LIFIA, IOTA, LSS) 1988.
- [SANF 83] SANFELIU A., FU K.
" A distance measure etwen attributed relational graphs for pattern recognition"
IEEE Tr on SMC vol SMC-13 n°3 mai juin 1983
- [SERR 84] SERRA J..
" Structures syntaxiques en morphologie mathématique"
Premier colloque image Biarritz 1984
- [SERR 86] SERRA J..
" Introduction to mathematical morphology"
Computer vision graphics and Image processing 35, pp 283-305, 1986

connexionnisme :

- [ACKEY 85]..... ACKLEY D.H. - HINTON G.E. - SEJNOWSKI T.J.
A learning algorithm for Boltzmann machines
Cognitive Science 9 : 147 - 169
- [FUKUS 83]..... FUKUSHIMA K. - MIYAKE S. - ITO T.
Neocognitron : a neural network model for a mechanism of visual pattern recognition
IEEE Transactions on Systems, Man and Cybernetics SMC-13 : 826 -834
- [HEBB 49] HEBB D.O.
How we know universals : the perception of auditory and visual forms
New York Wiley I "the first stage in perception : growth of the assembly " : xi-xix, 60-78.
- [HOPFI 82] HOPFIELD J.J.
Neural Networks and physical systems with emergent computational abilities
Proceedings of the National Academy of Sciences 79 : 2554 - 2558
- [KIRPA 83] KIRPATRICK S. - GELATT C.D. - VECCHI M.P.

Optimization by simulated annealing

Science 220 : 671 - 680

[KOHON 72]..... KOHONEN T.

Correlation matrix memories

IEEE Transactions on Computers C-21 : 353-359

[KOHON 82]..... KOHONEN T.

Self-organized formation of topologically correct feature maps

Biological Cybernetics 43 : 59 - 69

[KOHON 84]..... KOHONEN T.

Self organisation and associative memory

Berlin Singer - Verlag

[LECUN 87]..... LE CUN Y.

Modèles connexionnistes de l'apprentissage

Thèse de doctorat Université de PARIS VI

[MINSK 69]..... MINSKY M. - PAPERT S.

Perceptrons

Cambridge MA: MIT Press

[PERET 84]..... PERETTO P. - NIEZ J.J.

Stochastic dynamics of Neural Networks

IEEE Transactions on systems, Man and Cybernetics SMC-16 : no 1

[PERSO 86]..... PERSONNAZ L. - GUYON I. - DREYFUS G.

Collective computational properties of neural networks

New Learning Mechanisms. Physical Review A, vol 34 n°3

[PITTS 43]..... PITTS W. - Mc CULLOCH W.

A logical calculus of the ideas immanent in nervous activity

Bulletin of Mathematical Biophysics 5: 115-133

[ROSEN 58]..... ROSENBLATT F.

The perceptron : a probabilistic model for information storage and organization in the brain.

Psychological Review 65: 386-408

[RUMEL 6]..... RUMELHART D.E. - HINTON G.E. - WILLIAMS R.J.

Learning representations by back-propagating errors

Nature 323 : 533 - 536

[SEJNO 86]..... SEJNOWSKI T.J. - ROSENBERG C.R.

NETtalk : a parallel network that learns to read aloud

The Johns Hopkins University Electrical Engineering & Computer Science Techn. Report

[WIDRO 60]..... WIDROW B. - HOFF M.E.

Adaptative switching circuits

Ire Wescon Convention Record New York : 96 -104

Partie 2 - Chapitre 4.

Apprentissage

où l'on classifie les modèles

Partie 2 - Chapitre 4:

Le but de ce chapitre est de présenter la notion d'apprentissage.

Nous établissons tout d'abord un bref panorama historique de la notion d'apprentissage.

Nous en déduisons cinq définitions de l'apprentissage qui s'apparentent à autant de points de vue.

Nous les passons alors successivement en revue.

01. De la notion naïve de l'apprentissage à l'approche formelle

Traditionnellement l'apprentissage prend l'apparence d'un fait à décrire précisément tel qu'il est : répétition éventuellement renforcée de bastonnade, habitude, dressage, enrichissement presque continu de la connaissance à partir de l'expérience et de la réflexion sur la réflexion, ...

Savoir comment l'apprentissage est possible et se réalise, devient à l'ordre du jour lorsque la psychologie scientifique pose comme objet d'étude les conditions et les lois de l'apprentissage.

La première moitié de ce siècle consacre la domination de la psychologie du comportement (Behaviorisme et Néo behaviorisme) sur la psychologie de la forme.

La seconde moitié de ce siècle, voit émerger la psychologie cognitive, après un engouement temporaire pour les modèles stochastiques de l'apprentissage.

Ensuite, les théories neuro mimétiques (connexionnisme), contestent la conception cognitiviste .

TIBERGIEN [TIBER87] résume ces différentes approches dans un tableau:

	Comportement	Macro - cognition		Micro - cognition	
	psychologie behavioriste	psychologie néo behavioriste	psychologie de la forme gestalt	psychologie cognitive	connexionnisme
Quoi?	connexion stimuli - réponse		bonnes formes	représentations nœud, règle, schéma	configurations de poids synaptiques
Comment?	association répétition inhibition excitation généralisation différentiation	attente, espérance, renforcement "imaginatif"	structuration, contexte	création de nœuds de règles,... activation, compactage, contrôle	activation excitation, inhibition résonance.
Pourquoi?	loi de l'effet (plaisir, déplaisir)	curiosité, incertitude, intentionnalité	équilibre	intentionnalité, incertitude	convergence
Quand?	maturation et périodes critiques	maturation et périodes critiques	maturation et périodes critiques	développement de méta connaissances	maturation nerveuse, interaction

TURING, lui, dans ses articles pionniers (1947, 1950, 1952) présente l'apprentissage comme une caractéristique essentielle de toute machine voulant dépasser une stricte fonction mécanique. Il dégage trois modèles : les modèles intellectuel, génétique et culturel.

Le modèle intellectuel établit une analogie entre l'acquisition d'une machine et le psychisme d'un sujet, il se place ainsi dans le cadre de la psychologie du développement.

Le modèle génétique établit une analogie entre l'évolution des êtres vivants et l'évolution "biologique" des idées qui naissent, meurent, se reproduisent par croisement et mutation de gènes.

Le modèle culturel établit une analogie entre l'apprentissage et l'évolution historique et sociale des idées.

Plus récemment d'autres approches apparaissent:

Le "computo représentationnisme" assimile l'apprentissage à **un calcul** parce que la notion de calcul effectif est à la fois universelle et non triviale et propose une grille d'interprétation sévère et acceptable. C'est cette hypothèse qui identifie la pensée à un calcul que D. ANDLER appelle "computo représentationnisme".

D. MARR propose trois niveaux d'explication des phénomènes :

- 1 • mécanique ou biologique,
- 2 • fonctionnel ou algorithmique,
- 3 • formel.

S'intéresser à l'activité mentale comme un calcul sur des représentations symboliques formelles, c'est donc s'intéresser à un niveau supérieur d'explication.

P.M. CHURCHLAND propose quatre problématiques pour l'analyse des phénomènes de connaissance :

- 1 • la question ontologique (existence et nature des processus mentaux)
- 2 • la question sémantique (origine des significations)
- 3 • la question épistémologique (possibilité de connaître les états mentaux)
- 4 • la question méthodologique (moyens d'investigation rationnels des états et processus mentaux)

S'intéresser à l'activité mentale comme un calcul sur des représentations symboliques formelles, amènera donc à se poser ces quatre questions.

Du point de vue informatique, l'apprentissage formel est envisagé comme une modification répétée de l'état d'un dispositif calculatoire appelé "apprenti", modification elle-même d'ordre calculatoire.

Un tel apprentissage suppose définir :

- un objet d'apprentissage
- un protocole d'apprentissage
- des critères d'apprentissage, c'est à dire d'identification par l'apprenti de ce qui est à apprendre.

Nous proposons maintenant cinq points de vue sur l'apprentissage qui focalisent cinq définitions :

Apprentissage (au sens béhavioriste) : Modification stable du **comportement** imputable à l'**expérience**.

Apprentissage (au sens gestaltiste) : Modification stable de l'**organisation**, imputable à la **capacité à percevoir globalement des objets**

Apprentissage (au sens cognitiviste) : Modification stable des **structures mentales internes**, imputable à une **assimilation - accommodation**.

Apprentissage (au sens connexionniste) : Modification stable des **poids des connexions entre les unités**, imputable à une adaptation à une réponse imposée

Apprentissage (au sens formel) : Modification stable d'un **modèle mathématique sophistiqué**, imputable à l'élaboration d'un **calcul**

Dans cette perspective :

apprendre c'est converger (cadre des probabilités)

apprendre c'est décoder (cadre de la cryptographie)

apprendre c'est comprimer (cadre de la réduction de la quantité d'information pour une augmentation de sa qualité)

Nous nous proposons maintenant de passer en revue ces différentes sortes d'apprentissage.

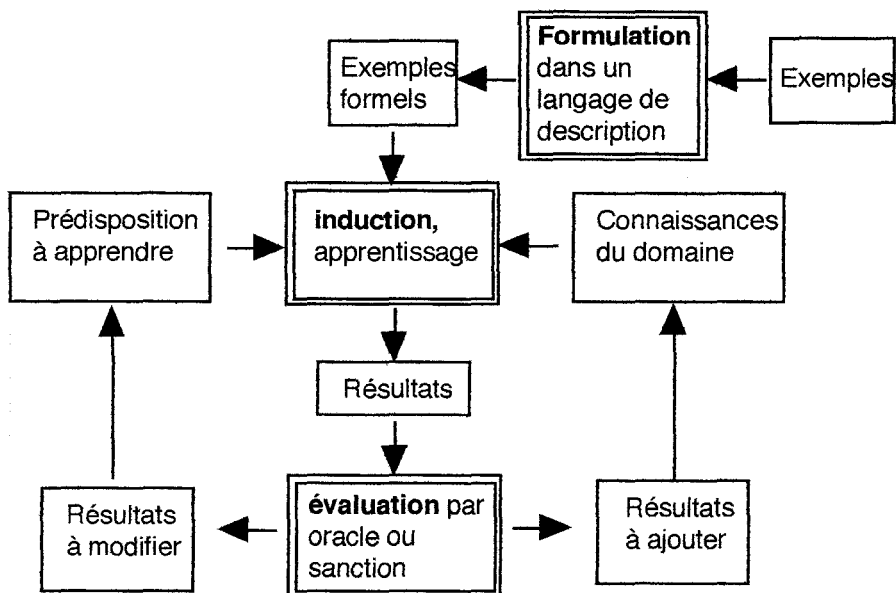
02. Apprentissage comportemental (théorie du comportement)

Modification stable du **comportement** imputable à l'expérience.

Modèle comportemental d'un mécanisme d'apprentissage :

Connaissances du domaine, prédisposition à apprendre et ensemble d'exemples formalisés sont trois catégories de connaissances utilisées par le système d'apprentissage pour fournir par induction un résultat évalué pour le réinvestir soit en ajout dans les connaissances du domaine, soit en modification dans les prédispositions à apprendre.

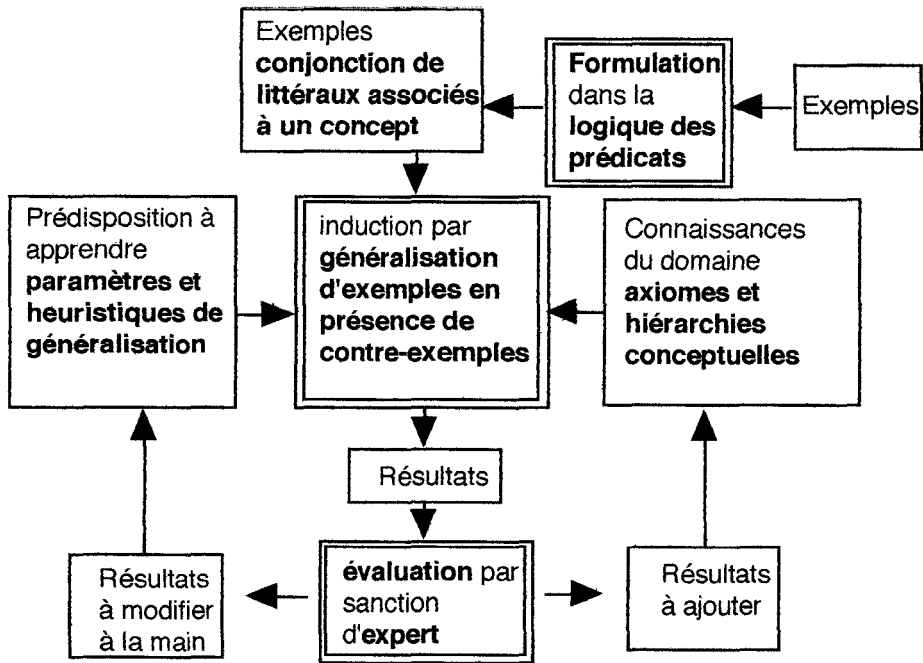
Ce modèle est schématisé ci-dessous :



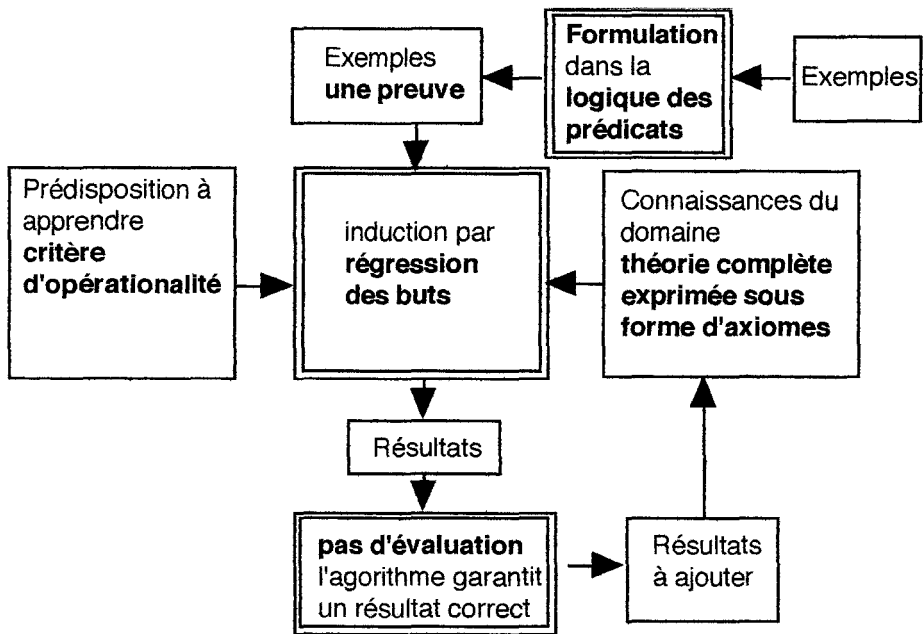
Illustrons ce modèle dans quatre domaines d'induction :

- ceux d'un apprentissage à partir de la détection de régularités,
- à partir d'explications,
- à partir de la décomposition en composantes principales,
- à partir d'un système connexionniste.

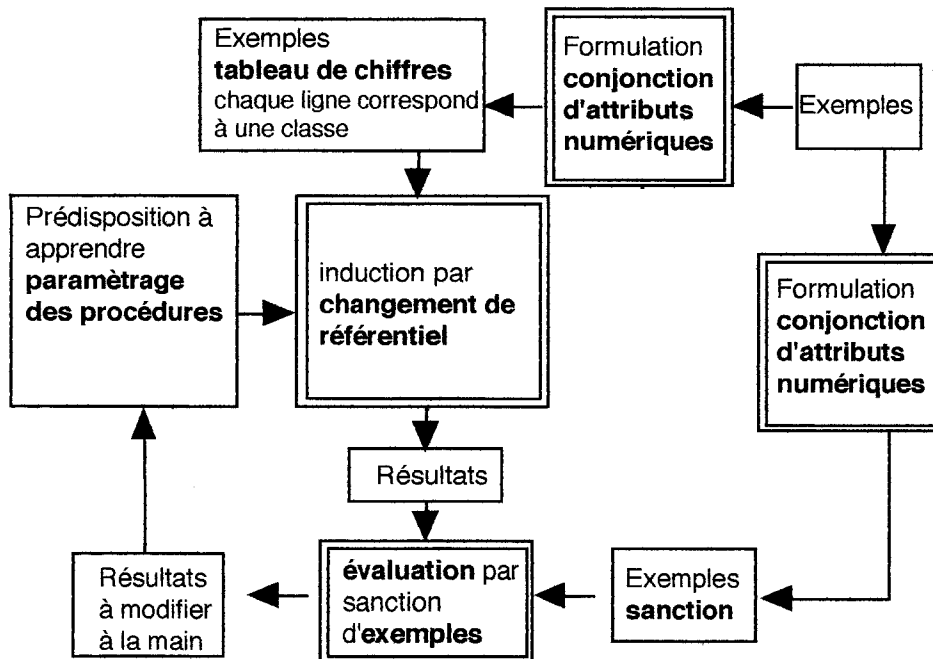
Exemple de ce modèle dans le cas d'un apprentissage par détection de régularités, de ressemblances



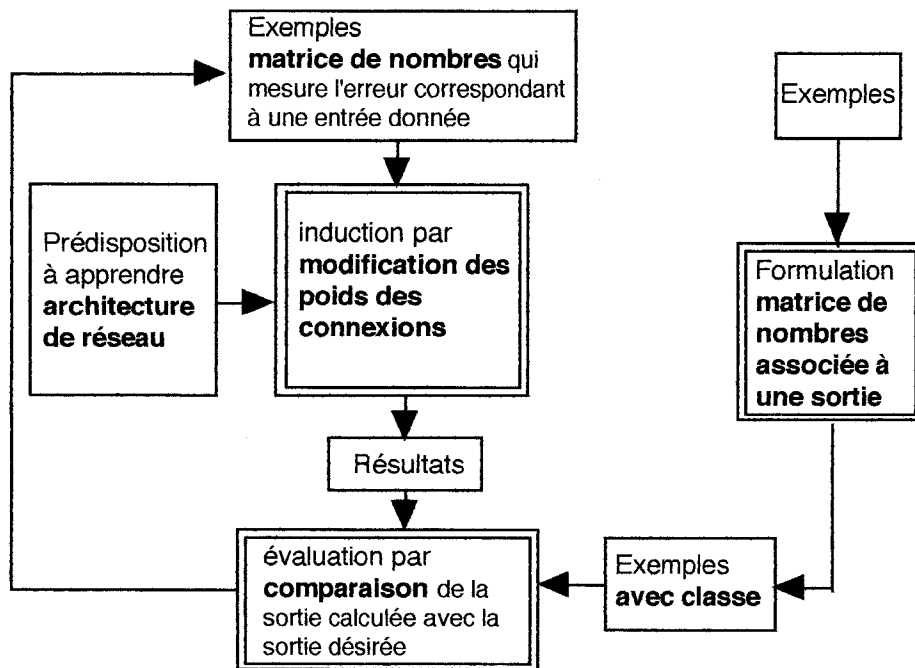
Exemple de ce modèle dans le cas d'un apprentissage à partir d'explications



Exemple de ce modèle dans le cas d'un apprentissage par décomposition en composantes principales



Exemple de ce modèle dans le cas d'un apprentissage par système connexionniste



Conclusion :

Le modèle comportemental exploite trois types de données :

- les exemples formalisés dans un langage de description.
- les connaissances de l'apprenti dans le domaine d'apprentissage.
- les prédispositions à apprendre de l'apprenti.

Le modèle comportemental met en œuvre trois types de fonctions :

- la formalisation des exemples, par un expert.
- l'apprentissage inductif, par l'apprenti.
- l'évaluation de l'apprentissage, par l'apprenti lui-même ou par oracle.

Nous constatons donc, que plus ou moins selon les cas, l'apprenti est assisté dans sa démarche d'apprentissage par des "personnages" extérieurs.

03. Apprentissage gestaltiste

(théorie de la forme)

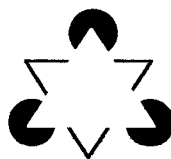
Modification stable de *l'organisation* imputable à la *capacité à percevoir globalement les objets*

L'apprentissage est une amélioration de l'organisation, apparaissant comme une propriété émergente de perceptions de formes, à partir d'une capacité à distinguer les objets, qui résulte d'une structuration accomplie par le système nerveux. Ce modèle pose en principe :

A. Le tout et les parties :

- 1 • la perception d'une image est immédiate et intuitive, à la fois sensible et cognitive.
- 2 • toute perception est globale : perception d'une forme structurée dont on ne détaille pas les éléments.
- 3 • le tout domine et hiérarchise les parties.

exemple : la composition ci-contre est perçue comme un triangle (illusion de KANIZSA)



- 4 • le tout ne peut être assimilé à la simple addition de ses parties : les relations entre les éléments sont plus importantes que les éléments eux mêmes.

B. Le fond et la forme :

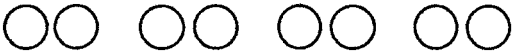





- 1 • tout champ perceptif se dissocie en un fond et une forme.
- 2 • la nature du fond peut influencer les caractères de la figure :

exemple : les parallèles ne sont plus perçues parallèles mais bombées. (illusion de HERING)



- 3 • un ensemble d'éléments est perçu comme une forme en raison des lois suivantes :
 - 3.1 : loi de petitesse : une petite forme se démarque d'un fond plus large.
 - 3.2 : loi du contour : une forme à contour fermé se démarque mieux qu'une forme non clôturée.
 - 3.3 : loi de simplicité : la forme simple l'emporte sur la forme compliquée.

- 3.4 : loi de régularité et de symétrie : une forme se démarque par une répartition régulière ou symétrique de ses éléments.
- 3.5 : loi de différenciation : une forme à structuration originale se démarque mieux.
- 3.6 : loi de ségrégation des unités :

• critère de proximité :	
• critère de région commune :	
• critère de ressemblance :	
• critère de continuité :	
• critère de connexité :	
• critère de fermeture :	

C. prégnance de la forme :

Quand, dans une image plusieurs formes peuvent être distinguées, l'une d'elles l'emporte par sa **prégnance**, c'est à dire *sa propriété à capter l'attention*.

Une forme est prégnante quand, mieux que d'autres, elle obéit aux lois énoncées ci-dessus, et notamment, à celles de simplicité, de régularité et de symétrie. Ces lois de prégnance seraient en accord avec celles de la nature : simplicité, équilibre, relative régularité. D'où l'hypothèse controversée de l'isomorphisme : les mêmes lois régneraient dans l'univers naturel extérieur et dans l'univers perceptif et mental.

04. Apprentissage cognitif

(théories piagétienne de connaissance)

Modification stable des **structures mentales internes**, imputable à une **assimilation - accommodation**.

Les modèles piagetiens identifient quatre concepts : le concept d'assimilation, d'accommodation, d'équilibration et, de structure logico algébrique, au travers de deux principes :

- 1 • Tout schème (structure) tend à **assimiler** (c'est à dire à s'incorporer les éléments extérieurs compatibles avec sa nature).
- 2 • Tout schème (structure) tend à **s'accommoder** aux éléments qu'il assimile (c'est à dire à se modifier pour tenir compte de la nouveauté sans perdre la mémoire du passé).

Un stade est un système général de pensée obéissant au principe d'équilibration. Piaget distingue ainsi quatre stades : le stade sensori-moteur (4 à 18 mois); le stade relationnel (18 mois à 5 ans); le stade dimensionnel (5 à 10 ans); le stade vectoriel ou abstrait (11 à 18 ans) L'évolution cognitive apparaît comme étant la complexification des mécanismes de traitement de l'information.

Les modèles suivants visent à décrire les niveaux successifs d'organisation des connaissances et les mécanismes de transition d'un niveau d'organisation à un autre.

4.1 Modèle de SIEGLER

Ce modèle, appelé "par élaboration de règles"

- identifie des états de connaissance et pose comme principe que les conduites sont régies par un ensemble de règles (si condition alors action) décisionnelles hiérarchisées représentées sous forme d'un arbre de décision.

4.2 Modèle de PASCUAL LEONE

Ce modèle, appelé "des opérateurs constructifs"

- identifie des schèmes subjectifs disponibles dans le répertoire de l'apprenti.
- identifie un système méta-constructif formé de quatre types d'opérateurs s'appliquant aux schèmes activés
 - l'opérateur M : espace mental.
 - l'opérateur L : apprentissage des structures logiques.
 - l'opérateur C : apprentissage de la différenciation des schèmes.
 - l'opérateur A : facteurs affectifs de motivations, de personnalité.

4.3 Modèle de Case

Ce modèle, appelé "des structures de contrôle"

- identifie trois représentations de base :
 - représentation de la situation problème (encodage),
 - représentation des objectifs à atteindre,
 - représentation des stratégies d'exécution.
- identifie quatre stades d'équilibration piagetiens :
 - stade de la structure de contrôle sensori-motrice,
 - stade de la structure de contrôle relationnelle,
 - stade de la structure de contrôle dimensionnelle,
 - stade de la structure de contrôle vectorielle ou abstraite.

4.4 Modèle de FISCHER

Ce modèle, appelé "des compétences spécifiques"

- identifie des classes de compétences hiérarchiquement ordonnées en dix niveaux,
- identifie deux concepts :
 - le concept de niveau optimal (la capacité limite de traitement de l'information),
 - le concept de règle de transformation hiérarchisée :
 - règle macro génétique (changement inter niveaux),
 - règle micro génétique (changement intra niveaux).

4.5 Modèle de HALFORD

Ce modèle, appelé "approche par correspondance structurale" :

- identifie quatre niveaux généraux du développement définis en termes de correspondance structurale entre le système symbolique S (représentation du problème) et le système environnement E (énoncé du problème).

La notion de correspondance structurale est une règle ou une procédure permettant d'établir une double correspondance :

- entre les éléments du système symbolique S et les éléments du système environnement E.
- entre les relations du système S et les relations du système E.

Conclusion :

Ces modèles posent comme hypothèse que le développement se caractérise par un nombre fini d'états d'équilibration. Le passage d'un état à un autre est le résultat d'une assimilation d'une nouveauté, d'une innovation qui peut remettre en cause l'équilibre atteint par un processus d'accommodation.

05. Apprentissage connexionniste

(théorie connexionniste)

Modification stable des **poids des connexions entre les unités**, imputable à une adaptation à une réponse imposée

On définit l'apprentissage comme la propriété pour un système de modifier son comportement en fonction de son histoire, c'est à dire comme la capacité d'acquérir de nouveaux comportements ou d'en modifier d'anciens par des expériences répétées.

Le problème de l'apprentissage pour un réseau de neurones formels, appelés unités, est celui de la modification des poids synaptiques des connexions, formalisant les influences réciproques, dans le but de faire réaliser au système la tâche souhaitée.

5.1 Formes d'apprentissage

+ Imposition d'une capacité (capacité innée)

Il n'y a pas de dynamique d'apprentissage à proprement parler. Les interactions entre les unités sont fixées en fonction de propriétés dont on veut doter le réseau. C'est à dire que c'est par un calcul a priori qu'elles vont être déterminées, et non pas par une évolution étape par étape.

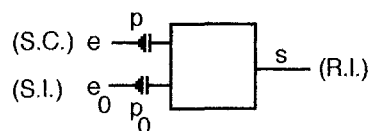
+ Apprentissage supervisé (capacité enseignée)

C'est un apprentissage avec professeur. Les interactions entre les unités se modifient progressivement pour amener le réseau à reproduire une réponse voulue à une sollicitation donnée.

Exemple d'apprentissage supervisé : conditionnement classique (type réflexe de PAVLOV).

On associe un stimulus conditionnel (SC) à un stimulus inconditionnel (SI) qui déclenche automatiquement une réponse inconditionnelle (RI).

On peut modéliser ce comportement de conditionnement par une règle de HEBB :
 $\Delta p = k \cdot e \cdot s$
 p_0 est une constante



Lorsqu'il y a effet de contexte, on peut modéliser le comportement par une règle de RESCOLA-WAGNER. On a alors des stimuli parasites (SP) qui viennent perturber l'apprentissage. Supposons qu'il y en ait trois notés (SP1), (SP2), (SP3). Notons p_1, p_2, p_3 les poids synaptiques correspondant à ces stimuli parasites, e_1, e_2, e_3 les entrées parasites correspondantes et s_1, s_2, s_3 les sorties. On aura la règle de RESCOLA WAGNER :

$$\Delta p_1 = k \cdot e_1 \cdot (s - (s_2 + s_3)) \text{ et } \Delta p_2 = k \cdot e_2 \cdot (s - (s_1 + s_3)) \text{ et } \Delta p_3 = k \cdot e_3 \cdot (s - (s_2 + s_1))$$

+ Apprentissage non supervisé (capacité acquise)

C'est un apprentissage sans professeur. Le réseau extrait de son environnement les informations qui lui permettent de réaliser des buts internes.

Exemple d'apprentissage non supervisé : apprentissage compétitif

L'idée de l'apprentissage compétitif est de renforcer pour un exemple présenté au réseau, les liens entre l'unité du réseau qui a le mieux reconnu cet exemple, et les unités d'entrée du réseau activées par l'exemple. Ainsi plus une unité du réseau a gagné pour un exemple, plus elle gagnera encore à l'avenir pour le même exemple. Les différentes unités du réseau tendent alors à se spécialiser dans des types de reconnaissances "antagonistes".

5.2 Mécanismes d'apprentissage

+ L'auto association

Le réseau mémorise des exemples qui lui sont présentés de façon répétitive. On cherche actuellement des procédures permettant de décider comment choisir les exemples et dans quel ordre les présenter au réseau.

Quand on présente une partie significative de l'exemple, ou un exemple proche, (dessin de forme floue ou incomplète par exemple), le réseau retrouve l'exemple appris : c'est l'auto association. Le réseau a appris s'il fournit en réponse, la sollicitation proposée.

+ L'hétéro association

Le réseau mémorise des couples d'exemples (E, S) qui lui sont présentés de façon répétitive. Il apprend à reproduire le second exemple quand on lui présente le premier. Quand on présente un exemple E, le réseau restitue la sortie S qui lui est associée. Par exemple à la présentation d'une forme, le réseau répond par le nom de cette forme.

- *la classification* (cas particulier d'hétéro association)

Les exemples sont classés dans des catégories. Le réseau mémorise des couples formés d'un exemple et de la catégorie de cet exemple qui lui sont présentés de façon répétitive. Il apprend à identifier la catégorie quand on lui présente l'exemple.

+ La détection de régularité

On dispose d'un ensemble d'exemples, qui sont présentés au réseau avec une probabilité donnée. Le réseau est supposé découvrir les exemples statistiquement les plus importants de l'ensemble.

5.3 Tactiques d'apprentissage

+ Apprentissage non supervisé compétitif

loi de HEBB [Voir réseau de HOPFIELD]

"Si deux éléments sont actifs ensemble, alors leur liaison sera renforcée".

+ Apprentissage supervisé fondé sur l'optimisation d'un indicateur

Cet indicateur peut être fondé sur une interprétation physique : une énergie, une entropie, une inertie, un potentiel.

On cherche à minimaliser cet indicateur

Indicateurs :

Soit un réseau constitué d'unités numérotés de 1 à N. On notera e_i l'état de l'unité numéro i et e_j l'état de l'unité numéro j et P_{ij} l'influence synaptique de la connexion entre l'unité numéro j et l'unité numéro i. On distingue dans ce réseau des unités d'entrée et des unités de sortie. On notera e_i^s l'état observé pour une unité de sortie numéro i et e_i^d l'état désiré pour cette même unité. On notera e_i^e l'état observé pour une unité d'entrée. On notera E_s l'ensemble des indices des unités de sortie.

On définit une énergie $J(P)$ du système connexionniste, fonction des poids P de ses connexions :

$$J(P) = \frac{1}{2} \sum_{i \in E_s} (s^d - s^c)^2$$

Cette énergie va être minimalisée par une méthode dite de gradient (ou encore dite méthode de ligne de plus grande pente, trajet suivi par exemple par une bille le long d'un plan incliné lorsqu'elle diminue son énergie potentielle en roulant vers le bas)

On calcule pour chaque influence synaptique P_{ij} la variation d'énergie qu'entraîne la variation de ce poids.

C'est à dire les dérivées partielles de l'énergie par rapport aux poids synaptiques : $\frac{\partial J}{\partial P_{i,j}}$

On modifie les poids synaptiques selon la ligne de plus grande pente par :

$\Delta P_{i,j} = -k \cdot \frac{\partial J}{\partial P_{i,j}}$ où k est une constante strictement positive qui modélise l'intensité de

l'apprentissage ou l'attention.

LOIS itératives :

unités prenant leurs valeurs dans un domaine continu :

Pour réaliser des adaptateurs, estimateurs, filtres, ...

But : minimalisation d'un indicateur somme quadratique des erreurs entre la sortie et le désiré

loi de WIDROW - HOFF

loi de Le CUN rétro propagation du gradient

unités prenant leurs valeurs dans un domaine binaire :

Pour réaliser des détecteurs, classificateurs, ...

But : minimalisation d'un indicateur somme quadratique des erreurs entre la sortie seuillée et le résultat désiré

loi de ROSENBLATT

LOIS stochastiques :

En exploitation :

lois d'HOPFIELD [82] [85], GLAUBER, Métropolis, KIRPATRICK

But : minimalisation d'un indicateur de type énergie inspiré de l'interaction entre les atomes d'un matériau magnétique désordonné (verres de spins) [Théories des transitions, des champs moyens, des répliques]

loi de BOLTZMANN

En apprentissage :

But : minimalisation d'un indicateur de type entropie relative pour rapprocher les probabilités d'évolution d'un réseau de BOLTZMANN, en phase contrainte et en phase libre.

Apprentissage par imposition d'une capacité innée

On calcule une corrélation linéaire entre une entrée et une sortie, et l'on impose les influences ainsi obtenues.

LOI linéaire:

"La sortie désirée D_s est fonction linéaire de l'entrée E par une formule $D = A \cdot E$ "

La matrice A de plus petite norme peut être calculée par $A = D \cdot E^+$

Conclusion :

L'apprentissage d'un système connexionniste, caractérisé par une matrice d'influences, cherche à modeler l'espace des configurations possibles pour y "creuser des trous", minima d'un indicateur, éventuellement interprétable en d'autres termes, témoin d'un concept à mémoriser.

06. Apprentissage formel

(théorie formelle)

Modification stable d'un **modèle mathématique sophistiqué**, imputable à l'élaboration d'un **calcul**

L'apprentissage est envisagé comme une modification répétée de l'état d'un dispositif calculatoire appelé "apprenti", modification elle-même d'ordre calculatoire.

Le présupposé de l'approche formelle est de déclarer que des niveaux d'explication formels et fonctionnels sont pertinents pour étudier un phénomène comme l'apprentissage, et fonctionnent de manière autonome.

Si apprendre c'est d'une manière ou d'une autre calculer, alors une approche de l'apprentissage doit s'appuyer sur la notion de calcul. Le cadre de la théorie de la complexité est un cadre candidat à une modélisation possible.

Si apprendre c'est d'une manière ou d'une autre converger, alors une approche de l'apprentissage doit s'appuyer sur la notion de convergence. Le cadre de la théorie des probabilités est un cadre candidat à une modélisation possible..

Le modèle de VALIANT [VALIA 84] ou théorie de l'apprentissage **pac** (probably approximately correct) que nous allons présenter maintenant respecte cette double filiation.

Définitions :

Notons F une classe de fonctions f d'un domaine X vers un codomaine Y $X \xrightarrow{f} Y$

D1 On appelle **exemple étiqueté** tout élément du produit cartésien : $X \times Y$

D1 bis On appelle **exemple étiqueté par f** tout élément $(x, f(x))$

D2 On appelle **échantillon** toute suite S (éventuellement infinie) d'exemples étiquetés.

D3 Un échantillon S est dit compatible avec une classe F de fonctions si et seulement si il existe une fonction f de F telle que pour tout couple (x, y) de S on ait $y = f(x)$

D4 On appelle **classe de concepts d'une classe F de fonctions de X vers $\{0;1\}$** ,

la classe C de parties du domaine X définie par :

$$C = \{ c : c \subseteq X \text{ et } \exists f \in F \text{ telle que } \forall x \in X \text{ on ait } (x \in c) \Leftrightarrow (f(x) = 1) \}$$

Les fonctions f de la classe F sont utilisées pour approcher un "phénomène".

Si le phénomène à modéliser se décrit à un moment donné par le couple (x, y) , la fonction f qui "l'approche" fournit le couple $(x, f(x))$.

Nous allons envisager l'erreur de modélisation entre y et $f(x)$ par, ce que la théorie de la décision appelle une fonction de perte.

D5 On appelle **fonction de perte** L_f toute fonction définie sur le produit cartésien $X \times Y$ à valeurs réelles positives et bornées par une quantité M .

$$\begin{array}{l} X \times Y \xrightarrow{L_f} [0, M] \\ (x, y) \longmapsto L_f(f(x), y) \end{array}$$

On note $\Omega(X)$ la tribu des boréliens définie sur X . $(X, \Omega(X))$ est ainsi un espace probabilisable, qui peut être probabilisé par une probabilité de densité D_X . La constitution d'un échantillon d'exemples étiquetés par une fonction f , peut alors se réaliser par une suite de tirages aléatoires dans X .

L'espace probabilisable $(X \times Y, \Omega(X) \times \Omega(Y))$ peut être muni d'une probabilité de densité D par $P_D\{A\} = P_{D_X}\{\{x \text{ tels que } (x, f(x)) \in A\}\}$

La fonction d'erreur L_f apparaît ainsi comme une variable aléatoire bornée sur $X \times Y$, qui admet ainsi des moments à tous les ordres.

D6 On appelle **erreur de f** relativement à D et L_f le moment d'ordre 1 de L_f (ou espérance $E_D(L_f)$).

La valeur optimale de cette erreur notée $\text{opt}(D, F)$ est définie par :

$$\text{opt}(D, F) = \min_{f \in F} E_D(L_f)$$

D7 On appelle **fonction de prédiction A** toute fonction définie sur $X \times S$ et à valeurs dans Y qui fournit comme image de (u, S) l'hypothèse v associée à u dans S

$$\begin{array}{l} X \times S \xrightarrow{A} Y \\ (u, S) \longmapsto v \\ \text{on } a(u, v) \in S \end{array}$$

La qualité d'une fonction de prédiction A est relative à la donnée d'une mesure D sur $X \times Y$ et d'une fonction de perte L_f et d'un échantillon S de S et dépend donc de leur choix pertinent.

D7 bis On note $A[S]$ la fonction définie sur X et à valeurs dans Y qui fournit comme image de u l'hypothèse v associée à u dans S

$$\begin{array}{l} X \xrightarrow{A[S]} Y \\ u \longmapsto v \\ \text{on } a(u, v) \in S \end{array}$$

D8 On appelle **distance de HAUSSLER** entre les deux fonctions f et g de F toute fonction d_k définie sur $F \times F$ et à valeurs dans les réels positifs définie comme ci-contre

$$\begin{array}{l} F \times F \xrightarrow{d_k} [0, +\infty] \\ (f, g) \longmapsto d_k(f, g) = \frac{|E_D(L_f) - E_D(L_g)|}{k + E_D(L_f) + E_D(L_g)} \end{array}$$

D9 Un problème d'apprentissage est défini par un quintuplet (X, Y, F, L, D) où X est le domaine du problème, Y le codomaine du problème, F une classe de fonctions de X vers Y , D une classe de mesures de probabilité sur $(X \times Y, \Omega(X \times Y))$, L une fonction de perte bornée par M .

D10 Une fonction de prédiction A résout un problème d'apprentissage (X, Y, F, L, D) si et seulement si pour tout k strictement positif et pour toute précision de prédiction α strictement comprise entre 0 et 1, et pour tout taux de fiabilité δ strictement compris entre 0 et 1, il existe une fonction m de \mathbf{R}^3 dans \mathbf{R} telle que pour toute mesure D de D , on ait : si S est un échantillon formé de $m(k, \alpha, \delta)$ couples tirés aléatoirement selon D alors on a :

$$P_D^{m(k, \alpha, \delta)} \{ d_k(\text{opt}(D, F), \text{err}_D(A[S])) \geq \alpha \} \leq \delta$$

La notion d'apprentissage est ainsi définie comme une convergence en probabilité, car \mathbf{A} définit une suite \mathbf{Z}_n de variables aléatoires par :

$$\mathbf{Z}_n(S) = \text{err}_D(\mathcal{A}[S \uparrow n])$$

Donc une fonction de prédiction \mathbf{A} résout un problème d'apprentissage si et seulement si pour toute mesure D , la suite \mathbf{Z}_n converge en probabilité vers :

$$\text{opt}(D, F)$$

La complexité en échantillon d'un problème d'apprentissage

D11 On appelle complexité en échantillon d'une fonction de prédiction \mathbf{A} , la fonction $m_{\mathbf{A}}$ de \mathbf{R}^3 dans \mathbf{R} qui vérifie :

$$m_{\mathbf{A}}(\alpha, k, \delta) = \min \{ n \text{ tels que } \max_D \{ \mathbb{P}_{D^n} \{ d_k(\text{err}_D(\mathcal{A}[S]), \text{opt}(D, F)) > \alpha \} \} < \delta \}$$

D12 On appelle complexité en échantillon d'un problème (X, Y, F, L, D) , le minimum de la complexité en échantillon des fonctions de prédiction \mathbf{A} qui résolvent ce problème :

$$m_F(\alpha, k, \delta) = \min_{\mathbf{A}} m_{\mathbf{A}}(\alpha, k, \delta)$$

Cherchons quelle est la complexité en échantillon d'un apprentissage d'une classe finie C de concepts lorsque :

la distance est dL^1 telle que : $dL^1(f, g) = |E_D(L_f) - E_D(L_g)|$

et la fonction de perte L^0 .

Un concept est défini par une classe F de fonctions de X dans $\{0; 1\}$

Lemme :

Soit un problème défini sur un domaine X par une classe F de fonctions f de X dans $\{0; 1\}$ et la fonction de perte L^0 définie par $L^0(a, a) = 0$ et si $a \neq b$ alors $L^0(a, b) = 1$ et une classe D de distributions de probabilités sur $X \times \{0; 1\}$

Soient deux réels ε et δ vérifiant $0 < \varepsilon < 1$ et $0 < \delta < 1$

et τ une fonction de $]0, 1[\times]0, 1[$ dans \mathbf{N}

Si

pour toute valeur de ε et δ , pour toute fonction f de F , et pour toute distribution D définie sur $X \times Y$ on a

1. les fonctions indicatrices des concepts de C qui vérifient :

$$\mathbb{P}_D \tau(\varepsilon, \delta) \{ \exists c \in C \text{ tel que } |\text{err}_D(f_c) - \widehat{\text{err}}_S(f_c)| > \frac{\varepsilon}{3} \} < \frac{\delta}{2}$$

et

2. la fonction de prédiction \mathbf{A} renvoie un concept de C qui vérifie :

$$\mathbb{P}_D \tau(\varepsilon, \delta) \{ |\widehat{\text{err}}_S(\mathcal{A}[S]) - \widehat{\text{opt}}(S, F)| > \frac{\varepsilon}{3} \} < \frac{\delta}{2}$$

Alors

pour toute distribution D sur $X \times Y$ on a :

$$\mathbb{P}_D \tau(\varepsilon, \delta) \{ |\text{err}_D(\mathcal{A}[S]) - \text{opt}(S, F)| > \varepsilon \} < \delta$$

C'est à dire la fonction de prédiction **A** résout le problème de prédiction $(X, \{0; 1\}, F, L^0, D)$ avec une complexité en échantillon $O(\tau)$.

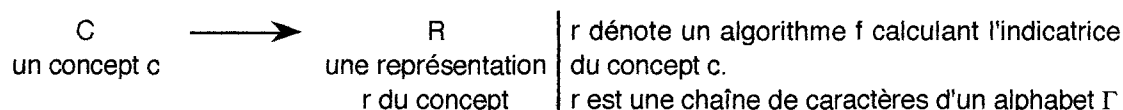
S'il est possible d'exhiber une fonction τ telle que les conditions du lemme soient remplies, alors une fonction de prédiction **A** qui produit un concept de C dont l'erreur empirique est optimale à $\varepsilon / 3$ (c'est à dire qui minimise approximativement l'erreur empirique) résout le problème à l'aide d'échantillons de taille $\tau(\varepsilon, \delta)$

La complexité calculatoire d'un problème d'apprentissage

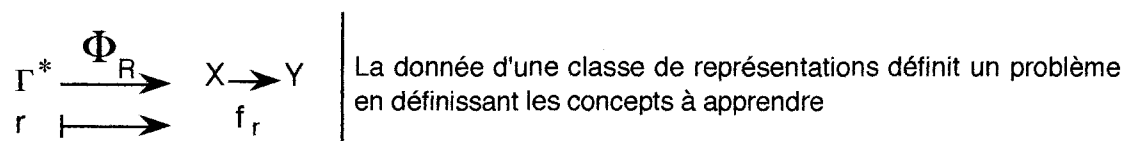
Dans un cadre calculatoire, le problème d'apprentissage satisfait aux contraintes suivantes :

- le domaine X et le codomaine Y sont discrets.
- les fonctions de F peuvent être représentées par un ensemble d'algorithmes.
- les fonctions de prédiction L sont calculables par des algorithmes.

Notion de représentation R des concepts d'un problème d'apprentissage.



L'ensemble R des représentations des concepts de C est une classe de représentation.



Exemples de classes de représentation :

- AEF : Automates à états finis
- FB : formules booléennes
- CIRC : circuits booléens

Notion d'évaluation :

Le problème d'évaluation $E(R)$ associé à la classe de représentation R est défini par :

- les instances (r, u, v) avec $r \in R, u \in X, v \in Y$
- les questions : v est-il un préfixe de $\Phi_R[r](u)$?

Exemples :

Le problème de l'évaluation $E(\text{AEF})$ des automates à états finis est soluble en espace logarithmique.

Le problème de l'apprentissage $(\Sigma^*, \{0,1\}, \text{AEF}, L^0, D)$ des automates à états finis est statistiquement insoluble, mais tout problème $(\Sigma^{[m]}, \{0,1\}, \text{AEF}^{[s]}, L^0, D)$ pour $m > 0$ et $s > 0$ est d'après le lemme trivialement de complexité en échantillon polynomiale en m et s .

L'algorithme de prédiction A pour la classe de représentation R est polynomial si et seulement si il existe un polynôme p tel que quel que soit l'échantillon $s \in S_l$ d'exemples étiquetés de $X^{[m]} \times Y$; quels que soient ϵ et δ strictement positifs, l'algorithme A se termine après au plus: $p(l, m, 1/\epsilon, 1/\delta)$ étapes.

Problème de prédiction :

Un problème de prédiction est défini par un quintuplet (X, Y, F, R, Φ) tel que :

- $X = \Sigma^*$ est le domaine du problème
- Y est le codomaine
- F est la classe des fonctions : $X \xrightarrow{f} Y$
- R est la classe des représentations écrites sur Γ^*

• Φ est la fonction : $\Gamma^* \xrightarrow{\Phi} X \rightarrow Y$

Un problème de prédiction est polynomial

si et seulement si :

il existe un polynôme p et un algorithme de prédiction A tels que :

pour tout s entier majorant la taille de la représentation à prédire

pour tout r de $R^{[s]}$

pour tout m

pour toute mesure de probabilité D induite par r et une probabilité DX à support sur $\Sigma^{[m]}$

pour tous les taux d'erreur et de fiabilité ϵ et δ strictement compris entre 0 et 1:

si

A reçoit un échantillon S constitué de $l = p(m, s, 1/\epsilon, 1/\delta)$ couples $(x_i, \Phi[r](x_i))$

où les $(x_i, \Phi[r](x_i))$ sont tirés selon D,

alors

$$P_S \{ er_D (A[S]) \geq \epsilon \} < \delta$$

Nous allons maintenant résumer dans un tableau le modèle de l'apprentissage formel, en l'illustrant d'un exemple : celui de l'apprentissage des rectangles du plan.

Problème P d'apprentissage pour un système artificiel (au sens de HAUSSLER)	Définitions	Exemple
<i>c'est quoi ?</i>	c'est : un quintuplet : $P = (X, Y, F, L, D)$ où	Problème de l'apprentissage RECT des rectangles du plan
1 l'espace des données du problème	X : est le domaine du problème	X est le plan euclidien \mathbb{R}^2
2 l'espace des résultats du problème	Y : est le codomaine du problème	Y est { oui, non } : le point (c, l) de \mathbb{R}^2 est-il dans le rectangle r ?
3 les concepts du problème	F : est une classe de fonctions f de X dans Y	F classe de fonctions notées r : modélisant les rectangles du plan définis par les coordonnées des extrémités de l'une de leurs diagonales $\mathbb{R}^2 \xrightarrow{r} \{\text{oui, non}\}$ $(c, l) \mapsto r(c, l)$ si $c_A \leq c \leq c_B$ et $l_A \leq l \leq l_B$ alors $r(c, l) = \text{oui}$ sinon $r(c, l) = \text{non}$
4 la variabilité du problème <i>remarque</i> : on peut se limiter à D_X densité de probabilité sur $(X, \Omega(X))$ car $(X \times Y, \Omega(X \times Y))$ est alors muni de D $P_D\{A\} = P_{D_X}\{x / (x, \chi(x)) \in A\}$ x est la fonction caractéristique du concept à apprendre	D : est une classe de densités de probabilités sur l'espace : $(X \times Y, \Omega(X \times Y))$	D : densités de probabilité, uniformes sur le plan.
5 la tolérance admise du problème on définit une erreur : $\text{err}(f) = E_D(L)$ (espérance) on définit un optimum : $\text{opt}(D, F) = \min_{f \in F} E_D(L)$	L : est une fonction de perte relative à f, bornée par M $X \times Y \xrightarrow{L} [0, M]$ $(x, y) \mapsto L(f(x), y)$ <ul style="list-style-type: none">perçue comme une variable aléatoire bornée sur $X \times Y$ qui admet ainsi des moments à tous les ordreschiffre l'erreur de modélisation entre y et f(x).	L fonction de perte standard si $r(c, l) = y$ alors $L((c, l), y) = 1$ sinon $L((c, l), y) = 0$ c'est à dire si la question r(c, l) : le point (c, l) est-il dans le rectangle r donne la réponse y alors la fonction L donne 1 sinon elle donne 0.
<i>comment est-il résolu?</i> apprendre c'est converger pour une suite S appelée échantillon, d'hypothèses de l'apprenti formulées à partir de présentations d'exemples. Lorsque la taille de l'échantillon croît, la suite des hypothèses de l'apprenti doit probablement converger vers une hypothèse approximativement correcte.	il est résolu par : une fonction de prédiction A appelée apprenti sur l'échantillon S $X \xrightarrow{A S} Y$ $u \mapsto v \quad \text{ou} \quad X \times S \xrightarrow{A} Y$ $(u, s) \mapsto v$ v est tel que $(u, v) \in S$ un problème d'apprentissage est donc résolu par un apprenti modélisé par une fonction	L'apprenti est modélisé par une fonction A de prédiction : $\mathbb{R}^2 \xrightarrow{A S} \{\text{oui, non}\}$ $(c, l) \mapsto A[S](c, l)$ une solution au problème d'apprentissage du rectangle r avec les données D, α et δ est un rectangle r' tel que $P_D\{r \Delta r'\} \leq 1 - \alpha$

<p>la fonction A modélise l'apprenti</p> <p>L'apprenti connaît :</p> <ul style="list-style-type: none"> • le taux de fiabilité δ • le taux d'erreur à respecter $1-\alpha$ ou la précision de prédiction α 	<p>A résout le problème P</p> <p><i>si et seulement si</i></p> <p>pour tout mesure D, la suite Z_n des variables aléatoires définies par :</p> $Z_n(S) = \text{err}_D(A[S n])$ <p>converge uniformément en probabilité vers $\text{opt}(D, F)$</p>	<p>L'apprenti tire aléatoirement des points du plan selon une mesure D de probabilité et détermine grâce à un "oracle" si les points appartiennent ou non au rectangle à apprendre; c'est à dire, il génère un échantillon S.</p> <p>On prouve que la constitution d'un échantillon de taille $\lceil \frac{4}{1-\alpha} \ln \frac{4}{\delta} \rceil$ assure l'apprentissage d'un rectangle r</p>
<p>autrement dit :</p> <p>pour tout paramètre d'écart de HAUSSLER k strictement positif, pour toute précision de prédiction α comprise entre 0 et 1, pour tout taux de fiabilité δ compris entre 0 et 1, il existe une fonction m de \mathbb{R}^3 dans \mathbb{R}, telle que pour toute mesure D de probabilité, tout échantillon S de $m(k, \alpha, \delta)$ couples tirés aléatoirement selon D, on ait :</p> $P_{D^{m(k, \alpha, \delta)}} \{ d_k(\text{optimum}, \text{erreur}(A[S])) \geq \text{précision} \} \leq \text{fiabilité}$		
<p>d_k est la distance de HAUSSLER définie par :</p> $d_k(f, g) = \frac{ E_D(L_f) - E_D(L_g) }{k + E_D(L_f) + E_D(L_g)}$	<p>la complexité en échantillon d'un apprenti A est la fonction m_A définie par</p> $m(k, \alpha, \delta) = \min_n \max_D (P_{D^n} \{ d_k(\text{opt}(D, F), \text{err}_D(A[S])) \geq \alpha \} \leq \delta)$ <p>la complexité en échantillon d'un problème P est le minimum de la complexité en échantillon des apprentis A résolvant le problème</p>	

<i>Quelles sont ses caractéristiques ?</i>	Le problème ...	
<p>1 la nature du terrain d'apprentissage</p> <p>une des qualités de ce protocole sera sa reproductibilité</p>	<p>est soumis à un protocole</p> <p>un protocole est articulé en deux composantes :</p> <ul style="list-style-type: none"> • l'état initial (une configuration stable), c'est à dire les compétences initiales de l'apprenti • une interaction dans une durée de l'apprenti avec son environnement 	<p>par exemple pour les interactions :</p> <ul style="list-style-type: none"> • un dressage • un processus d'équilibration relaxation • une induction • un processus de transmission
<p>2 la nature du succès</p> <p>Le critère d'apprentissage ne sert pas à évaluer l'apprentissage, mais à permettre la définition de problèmes d'apprentissage justiciables de traitements et de démonstrations formels.</p>	<p>dispose d'un critère de succès :</p> <p>un critère d'apprentissage définit les suites d'hypothèses qui peuvent être produites par un apprenti qui a effectivement appris l'objet à apprendre, il peut ne pas être vérifiable expérimentalement mais seulement être utilisé pour une preuve formelle.</p>	<p>par exemple :</p> <ul style="list-style-type: none"> • le critère d'identification finie. • fréquence d'apparition de certains événements • détection de certaines régularités
<p>3 la nature de ce que l'on cherche à apprendre</p>	<p>possède un objet d'apprentissage :</p>	<p>par exemple :</p> <ul style="list-style-type: none"> • des catégories perceptives techniques : triangles, cercles,... • des fonctions • des grammaires...

07. Apprentissage formel inductif

(théorie de l'inférence inductive)

Modification stable d'une **loi générale**, imputable à l'observation d'**exemples particuliers**, dans un processus d'inférence.

Apprendre c'est raisonner par induction

- Une induction est un processus d'inférence d'une loi générale ou d'un principe, à partir de l'observation d'instances particulières, d'exemples.
- Un raisonnement inductif, c'est de manière plus générale un apprentissage par l'expérience.

Le problème de l'inférence inductive peut se poser dans les termes suivants:

On dispose d'un espace H d'hypothèses : $H = \{ H_1, \dots, H_n \}$ mutuellement exclusives et exhaustives. A chaque hypothèse H_i , on assigne une probabilité $p(H_i)$ dite probabilité a priori de l'hypothèse H_i .

A partir d'une donnée D, on cherche à sélectionner l'hypothèse H_i la plus compatible avec cette donnée.

On note :

$p(D / H_i)$ la probabilité d'observer la donnée D sachant que l'hypothèse H_i est correcte.

$p(H_i / D)$ la probabilité a posteriori

On a les formules de BAYES et des probabilités totales :

$$p(H_i / D) = \frac{p(D / H_i) \cdot p(H_i)}{p(D)} = \frac{p(D / H_i) \cdot p(H_i)}{\sum_i p(D / H_i) \cdot p(H_i)}$$

Toutes ces probabilités sont supposées calculables à partir de D et H.

Dans le contexte de l'apprentissage, $p(H_i)$ est la croyance initiale de l'apprenti dans l'hypothèse H_i .

Comme nous l'avons fait régulièrement, particularisons le problème général dans le domaine des chaînes de caractères pris d'un alphabet $\Omega = \{ 0, 1 \}^*$.

Définissons une semi mesure μ sur Ω par $\mu(x) = \sum_{y \in \Omega} p(xy)$, perçue comme une "chance"

d'observer une chaîne ayant le préfixe x, c'est à dire qui commence par la sous chaîne x.

Dans ce contexte, le problème de l'inférence est de prédire le symbole suivant de la chaîne, à l'aide de la semi mesure μ qu'il nous faudra définir.

Ainsi H_i est l'hypothèse que la chaîne admet comme préfixe S_i où $i \in \{0, 1\}$ (c'est à dire ici soit S_0 soit S_1) et la donnée D est le fait que la chaîne admette le préfixe S.

on a donc : $p(S_i / S) = \frac{p(S / S_i) \cdot \mu(S_i)}{\mu(S)}$ mais si la chaîne S est générée par un processus

déterministe nous avons $p(S / S_i) = 1$ pour tout i, si bien que : $p(S_i / S) = \frac{\mu(S_i)}{\mu(S)}$

Il s'agit alors de déterminer pour un problème donné une distribution de probabilités a priori. Certaines particularités du problème peuvent nous amener à décider d'accepter telle ou telle distribution, mais bien souvent nous ne saurons quoi choisir.

Le résultat suivant est donc fondamental :

SOLOMONOFF propose une mesure μ , distribution universelle de probabilité a priori

Distribution de SOLOMONOFF LEVIN :

Notons $A(x)$ l'ensemble de tous les programmes p auto délimités qui sortent la chaîne x sur une machine de TURING universelle U et $P_U(x) = \sum_{p \in A(x)} 2^{-|p|}$

Après avoir rappelé quelques définitions, nous allons voir que la distribution P_U possède quelques propriétés remarquables :

- D1 Toute fonction $\mu : \mathbb{N} \rightarrow [0,1]$ satisfaisant aux propriétés des distributions de probabilités excepté que $\sum_x \mu(x) \leq 1$ est appelée **mesure**.
- D2 Toute fonction f est appelée **semi calculable inférieurement** si et seulement s'il existe une fonction récursive $(x,k) \rightarrow g(x,k)$ non décroissante en k telle que $f(x) = \lim_{k \rightarrow +\infty} g(x,k)$
- D3 la mesure μ **domine** la mesure μ' s'il existe une constante c telle que pour tout x entier on ait : $\mu'(x) \leq c \cdot \mu(x)$
- D4 Dans une classe de fonctions, une mesure qui domine toutes les autres est appelée **mesure universelle**. Nous noterons m une telle mesure.

Il est connu qu'aucune mesure μ ne domine toutes les autres dans la classe des mais :

Théorème regroupant les résultats de **LEVIN 1970**,

- La distribution P_U de SOLOMONOFF LEVIN,
- la distribution $2^{-K(x)}$ (où $K(x)$ est la complexité auto délimitée de KOLMOGOROV de la donnée x),
- toute mesure m universelle de LEVIN de la classe des mesures qui sont *semi calculables inférieurement* coïncident à une constante multiplicative près.

En effet, il existe une constante c telle que pour tout x , à cette constante c additive près on ait $-\log m(x) = -\log P_U(x) = K(x)$ ce résultat étant basé sur l'inégalité de Kraft :

Inégalité de Kraft : soit l_1, l_2, \dots est une suite d'entiers

dire $\sum_n 2^{-l_n} \leq 1$

équivalent à dire il existe un prefix code $c : \mathbb{N} \rightarrow \{0; 1\}^*$ tel que $l_n = |c(n)|$

Ainsi toute mesure $m(x)$ universelle semi calculable inférieurement, la probabilité a priori $P_U(x)$ de SOLOMONOFF et l'expression toute simple $2^{-K(x)}$ coïncident à une constante multiplicative près.

Ces résultats fondent quelques remarques de bon sens :

- P_U équivaut à $2^{-K(x)}$ où $K(x)$ est la cause la plus courte ; c'est le principe d'OCCAM.

principe du rasoir d'OCCAM : choisir en priorité l'explication la plus simple.

Règle de KEMENY : choisir l'hypothèse la plus simple compatible avec les valeurs observées.

- K et P_U n'étant pas calculables, toute méthode d'apprentissage passera par des heuristiques qui implicitement utilisent des approximations.

BIBLIOGRAPHIE

apprentissage :

- [BOUCH 92] BOUCHERON S.
" Théorie de l'apprentissage : de l'approche formelle aux enjeux cognitifs "
HERMES1992.
- [TIBER 87] TIBERGIEN G., ROULIN J.L., POLLIER A.
" Apprendre : quoi quand comment et pourquoi? "
Actes des 2èmes journées françaises de l'apprentissage. IMAG LIFIA Grenoble
1987.
- [VALIA 84] VALIANT L.G.
A theory of the learnable
Communications of the ACM 27:1134-1142, 1984

connexionnisme :

- [ACKEY 85] ACKLEY D.H. - HINTON G.E. - SEJNOWSKI T.J.
A learning algorithm for BOLTZMANN machines
Cognitive Science 9 : 147 - 169
- [FUKUS 83] FUKUSHIMA K. - MIYAKE S. - ITO T.
**Neocognitron : a neural network model for a mechanism of visual pattern
recognition**
IEEE Transactions on Systems, Man and Cybernetics SMC-13 : 826 -834
- [HEBB 49] HEBB D.O.
How we know universals : the perception of auditory and visual forms
New York Wiley | "the first stage in perception : growth of the assembly " : xi-xix,
60-78.
- [HOPFI 82] HOPFIELD J.J.
Neural Networks and physical systems with emergent computational abilities
Proceedings of the National Academy of Sciences 79 : 2554 - 2558
- [KIRPA 83] KIRPATRICK S. - GELATT C.D. - VECCHI M.P.
Optimization by simulated annealing
Science 220 : 671 - 680
- [KOHON 72] KOHONEN T.
Correlation matrix memories
IEEE Transactions on Computers C-21 : 353-359
- [KOHON 82] KOHONEN T.
Self-organized formation of topologically correct feature maps
Biological Cybernetics 43 : 59 - 69
- [KOHON 84] KOHONEN T.
Self organisation and associative memory
Berlin Singer - Verlag
- [LECUN 87] LE CUN Y.
Modèles connexionnistes de l'apprentissage

Thèse de doctorat Université de PARIS VI

[MINSK 69]..... MINSKY M. - PAPERT S.

Perceptrons

Cambridge MA: MIT Press

[PERET 84]..... PERETTO P. - NIEZ J.J.

Stochastic dynamics of Neural Networks

IEEE Transactions on systems, Man and Cybernetics SMC-16 : no 1

[PERSO 86] PERSONNAZ L. - GUYON I. - DREYFUS G.

Collective computational properties of neural networks

New Learning Mechanisms. Physical Review A, vol 34 n°3

[PITTS 43]..... PITTS W. - Mc CULLOCH W.

A logical calculus of the ideas immanent in nervous activity

Bulletin of Mathematical Biophysics 5: 115-133

[ROSEN 58] ROSENBLATT F.

The perceptron : a probabilistic model for information storage and organization in the brain.

Psychological Review 65: 386-408

[RUMEL 6]..... RUMELHART D.E. - HINTON G.E. - WILLIAMS R.J.

Learning representations by back-propagating errors

Nature 323 : 533 - 536

[SEJNO 86] SEJNOWSKI T.J. - ROSENBERG C.R.

NETalk : a parallel network that learns to read aloud

The Johns Hopkins University Electrical Engineering & Computer Science Techn. Report

[WIDRO 60]..... WIDROW B. - HOFF M.E.

Adaptative switching circuits

Ire Wescon Convention Record New York : 96 -104

Partie 2 - Chapitre 5.

Modélisation des systèmes complexes

où l'on introduit l'approche systémique

Partie 2 - Chapitre 5 :

L'objet de ce chapitre est d'étudier la notion de modélisation.

Un phénomène compliqué sera rendu compréhensible par sa simplification, sa décomposition en éléments simples.

Un phénomène complexe sera rendu intelligible par une composition d'actions pour un projet concevable de connaissances.

01. La modélisation : rendre intelligible un système

Nous cherchons à modéliser un phénomène, c'est à dire à réaliser une construction intentionnelle par composition de symboles ou de sous modèles susceptibles :

- de rendre intelligible le phénomène perçu comme complexe ou compliqué,
- d'amplifier les raisonnements relatifs au phénomène,
- d'anticiper les conséquences des projets d'actions possibles dans le cadre du phénomène.

Deux sortes de phénomènes se distinguent :

- les phénomènes compliqués,
- les phénomènes complexes.

La modélisation de chacune de ces sortes de phénomènes, obéit à des principes différents de conception.

Pour donner du sens, pour comprendre :

un **Système compliqué** : on peut le simplifier pour découvrir son explication, par décomposition en unités simples.

un **Système complexe** : on doit le modéliser pour construire sa compréhension, son intelligibilité.

En simplifiant, c'est à dire en mutilant un système complexe, on détruit à priori son intelligibilité.

L'intelligibilité du compliqué se fait par simplification et donc par mutilation

Le simple est toujours le simplifié [BACHELARD] :

La méthode est : **Commençons par simplifier le compliqué.**

L'intelligibilité du complexe se fait par modélisation

Nous ne raisonnons que sur des modèles [P. VALERY] :

La question est : **Quelles méthodes pour modéliser la complexité ?**

Comment élaborons-nous les modèles sur lesquels nous raisonnons ?



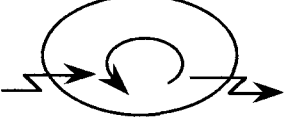

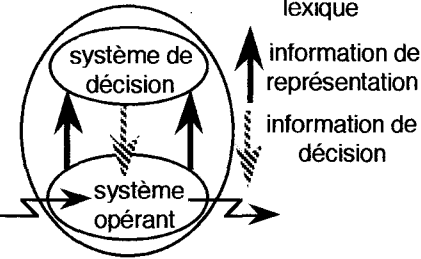
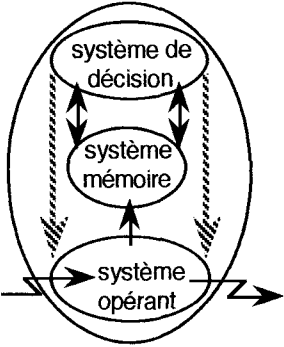
Le modèle est une représentation artificielle que l'on construit dans sa tête et que l'on "dessine" sur un support physique, construit par l'homme, il agence des symboles.

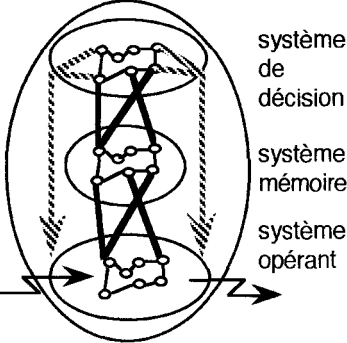
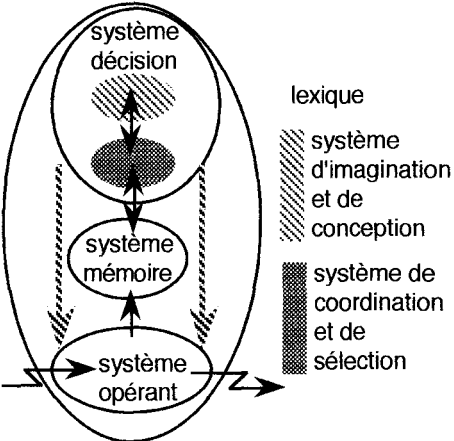
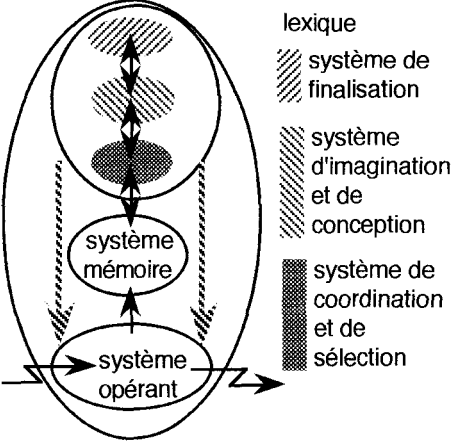
Modéliser, c'est à la fois identifier et formuler un problème et chercher à le résoudre en raisonnant sur des simulations. *"Modélisation et simulation, réflexion et raisonnement sont les deux faces inséparables de toute délibération"* [LEMOI 90]

comparaison modélisation analytique – modélisation systémique

Modélisation analytique du compliqué	Modélisation systémique du complexe
<p>Concept de base : l'objet élémentaire</p> <p>objet comme élément d'une structure</p> <p>objet découpable</p>	<p>Concept de base : l'action implexe ou processus ou projet</p> <p>action comme élément d'une composition</p> <p>projet concevable de connaissance</p>
<p>Question initiale :</p> <p>De quoi c'est fait ?</p>	<p>Question initiale :</p> <p>Qu'est ce que ça fait ?</p>
<p>Logique disjonctive :</p> <p>C'est la logique Aristotélicienne pour une méthode hypothético-déductive. L'effort est un effort de discernement qui sépare des objets d'expérience [disjoindre]</p>	<p>Logique conjonctive :</p> <p>C'est une logique du "ou" pour une méthode axiomatique inférentielle. L'effort est un effort de généralisation conjoint qui unifie des complexes d'expérience [joindre]</p> <p>L'acte fondamental est de joindre son attention au système à considérer.</p> <p>La modélisation d'un système complexe va s'organiser en une série d'itérations entre les projets et les représentations symboliques que s'en construit le modélisateur.</p> <p>"Rien n'est donné, tout est construit" BACHELARD</p> <p>Un système complexe est un système construit par l'observateur qui s'y intéresse, c'est un modèle d'un phénomène perçu complexe, que l'on construit par modélisation systémique.</p> <p>Le système est vu comme un enchevêtrement intelligible et finalisé d'actions indépendantes, d'interactions en inter relations</p>
<p>pour du ... compliqué</p> <p>décomposable</p> <p>en agissant par décomposition</p> <p>de simplexe : unité simple</p>	<p>pour du ... complexe</p> <p>indécomposable</p> <p>en agissant par composition</p> <p>d'implexe : unité d'action indécomposable</p>
<p>exemples de modèles de problèmes compliqués :</p> <p>théories mathématiques, statistiques, de la décision, du contrôle optimal, des catégories, des catastrophes, des bifurcations, ...</p>	
<p>méthodes à la recherche d'un problème qui leur convienne</p>	<p>problèmes à la recherche de méthodes qui leur conviennent</p>

La caractérisation d'un phénomène dans une perspective de modélisation se hiérarchise en 9 niveaux de description :

1	il est	le phénomène est identifiable	
2	il fait	le phénomène est actif	
3	il est régulé	le phénomène possède une certaine stabilité	
4	il s'informe	le phénomène se renseigne sur son comportement	
5	il décide	le phénomène prend le statut de système, il est composé de sous systèmes en interaction : le système de décision et le système opérant.	
6	il mémorise	le phénomène est un complexe de systèmes en interaction : systèmes de décision, de mémorisation et opérant.	

7	il coordonne	le phénomène est un complexe de systèmes en interaction : systèmes de décision, de mémorisation et opérant ; on identifie une coordination.	 <p>système de décision système mémoire système opérant</p>
8	il imagine	le phénomène est un complexe de systèmes en interaction, systèmes de décision, de mémorisation et opérant ; on identifie une coordination et une imagination.	 <p>système décision système mémoire système opérant</p> <p>lexique système d'imagination et de conception système de coordination et de sélection</p>
9	il se finalise	le phénomène est un complexe de systèmes en interaction, systèmes de décision, de mémorisation et opérant ; on identifie une coordination, une imagination et une finalisation. Buts et objectifs déterminent l'évolution du phénomène dans le temps.	 <p>système décision système mémoire système opérant</p> <p>lexique système de finalisation système d'imagination et de conception système de coordination et de sélection</p>

Nous allons illustrer notre propos en construisant deux modèles : les modèles de la coopération et de l'organisation.

02. Le modèle de la coopération

Le modèle de la coopération se caractérise par une connaissance répartie entre différents acteurs spécialistes ayant chacun des méthodes de résolutions spécifiques, des points de vue différents, qui coopèrent entre eux pour résoudre un problème soumis à des contraintes.

Ainsi chacun des experts résout un sous problème qui peut soit s'inscrire dans un effort commun entrepris par un groupe, soit entrer en concurrence avec d'autres.

Ainsi chacun des experts dispose de connaissances limitées sur son environnement, sur les actions et les intentions des autres.

Ainsi se produisent des échanges (aspect communication).

Ainsi s'exploitent des ressources à partager (aspect cohabitation).

Ainsi se définissent des tâches à réaliser (aspect coopération).

Ainsi doit émerger un comportement global cohérent de la société d'experts à partir de décisions prises localement par chaque spécialiste.

Phénomène	Optimisation du comportement du phénomène.	Les actions	Paramètre sur lequel agit le phénomène.	Action
Reconnaître	la reconnaissance	filtrage unification	forme	filtre (forme)
Se situer	le placement	mouvement	distance	mouv (dist)
Communiquer	la communication	coopération sociale	signal	coop (signal)
Planifier	la meilleure stratégie de résolution de problème	création de plan stratégique	plan	strat (plan)

Il s'agit de faire coopérer des méthodes préexistantes pour résoudre des problèmes dont la difficulté excède la capacité de chaque méthode considérée séparément.

Certains problèmes émergent dans le cadre de la coopération de méthodes distribuées :

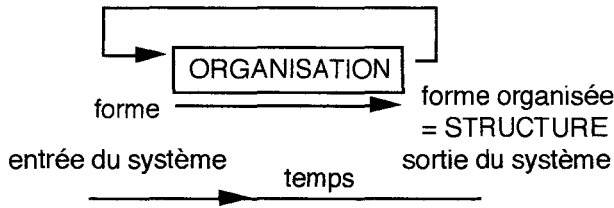
- utiliser des méthodes hétérogènes,
- autoriser l'emploi de plusieurs formalismes,
- faire coopérer les méthodes en l'absence d'algorithme connu de recherche.

Chaque module possède trois caractéristiques :

- regroupement de connaissances homogènes reposant sur un même formalisme et les mêmes modèles de compétence,
- le contrôle est bien connu et maîtrisé,
- les modules sont indépendants entre eux.

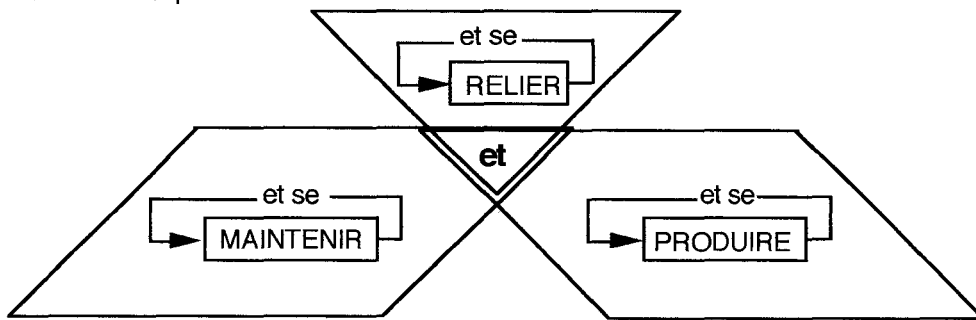
03. Le modèle de l'organisation

Un système d'organisation est un système qui admet en entrée une forme et qui fournit en sortie une forme organisée, c'est à dire une structure.



Qu'est-ce que de l'organisation ?

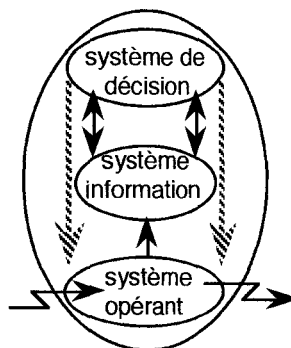
L'organisation rend compte du comportement de chacun des niveaux ci-dessous, à la fois vis à vis de leur projet et de leur articulation les uns par rapport aux autres, sans qu'il soit possible de les séparer.



C'est à dire :
 FONCTIONNER
 et
 SE TRANSFORMER
 dans
 UN ENVIRONNEMENT

c'est à dire :
 Maintenir, relier et produire
 et
 se maintenir, se relier et se produire
 pour
 quelques finalités

Le modèle de l'organisation est un complexe de trois sous systèmes : un système de décision, un système d'information et un système d'opération.



Le projet du système est d'augmenter son organisation, sa compétence.
 A chacun des niveaux de ces trois systèmes, on distingue :

Système de décision	COMPRENDRE	CONCEVOIR	FINALISER
Système d'information	COMPUTER	COMMUNIQUER	MEMORISER
Système d'opération	PRODUIRE	RELIER	MAINTENIR
	dans un même instant	et dans une évolution	pour une autonomie

04. Le symbole

L'information qui in-forme, l'organisation qui la forme : la symbolisation.

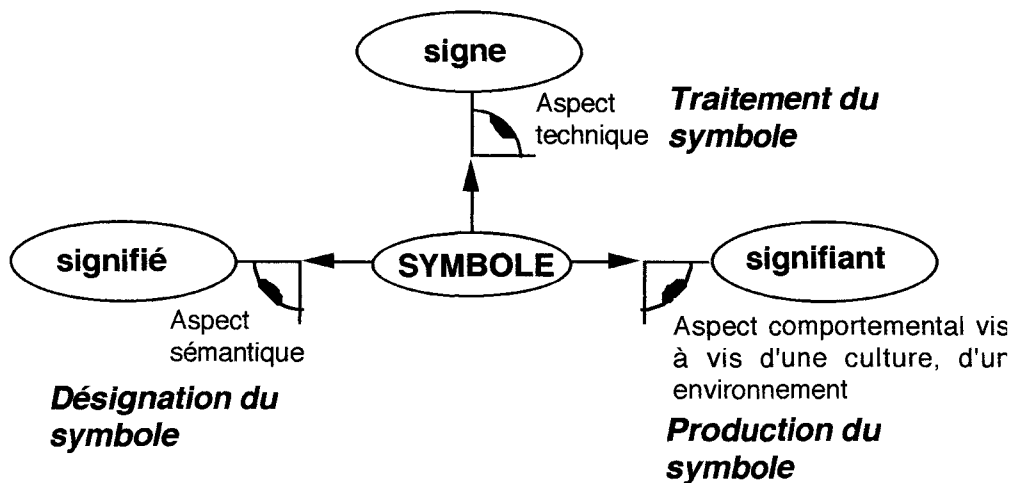
Question : Y a t il des unités constitutives du signe ?

Réponse d'Umberto ECO :

Une unité ne peut être cataloguée une fois pour toute : ainsi un point pourra tantôt représenter un œil, tantôt un pépin de raisin : il a à chaque fois un sens contextuel.

Si les codes iconiques existent alors ce sont des codes faibles, mais on peut distinguer trois articulations dans le signe visuel :

- la figure au sens géométrique
- le signe : unité de reconnaissance difficile à isoler (nez, œil, nuage, ...)
- l'énoncé plus ou moins complexe : "ceci est un cheval debout de profil".



Le symbole est un symbole qui relie un symbole à un symbole qui relie un symbole ...

Ainsi un symbole apparaît comme un opérateur op à point fixe unique, qui fournit un sens K perçu soit comme l'opérateur lui-même ou son point fixe.

Désignation symbolique	c'est l'identification des symboles dans et par l'action vis à vis d'un projet.
Computation symbolique	c'est la production et la transformation de configurations de symboles.

On peut chercher à donner du signifié une description formelle, espérant définir un procédé permettant d'associer à la structure "profonde" de la sémantique, la structure de surface syntaxique? Un tel "foncteur" ne sera bien sur pas univoque, car on peut admettre que toutes les possibilités d'expressions synonymes d'un texte donné sont relativement limitées en nombre.

Pourtant, au niveau des structures syntaxiques "de surface", l'adéquation des systèmes formels à la description d'un corpus est des plus limitée. La notion de formalisation d'une structure syntaxique n'est pas encore clairement définie. Elle pourrait être le siège d'un processus dynamique permanent dont la structure formelle statique ne sera qu'une morphologie observable, au sein d'une théorie qui rétablirait le lien jusqu'ici manquant entre dynamique globale (le signifié) et morphologie locale en laquelle elle se manifeste (le signifiant).

BIBLIOGRAPHIE

ouvrages généraux :

[LEMOI 90] LEMOIGNE J.L.

" La modélisation des systèmes complexes "

Afcet Systèmes Dunod, Paris 1990

Partie 2 .

Guide : Survol du paysage étudié

Nous donnons dans ce chapitre deux guides de lectures, à deux vitesses, des textes des chapitres précédents : une première lecture fort résumée, et une seconde résumée au septième.

Résumé partie 2

Partie 2 - Chapitre 1 - La machine PASTIS

Le projet PASTIS se propose d'étudier une machine qui explore, sur un jeu de données graphiques, un itinéraire de compréhension de dessin, en adaptant et en optimisant pour chaque niveau : l'architecture à la complexité, le temps à l'espace, la redondance à l' à peu près.

L'objectif général de ce projet est de concevoir et fonder des modules architecturaux spécialisés pour l'apprentissage et la reconnaissance de "dessins au trait".

Ces modules seront rapides parce qu'approximatifs, et utilisés aussi bien pour des données de bas niveau que pour des données symboliques.

PASTIS sera une machine qui évolue dans le temps.

Partie 2 - Chapitre 2 - Modèles neurobiologiques de la vision

Les travaux des neurobiologistes permettent d'entrevoir des explications des mécanismes cérébraux.

Ces travaux décrivent le cortex en termes de régions, d'aires spécialisées dans une fonction précise : détection de formes, de lignes, de couleurs, de mouvement, ... stimulées en parallèle, socialisant leurs activités pour faire émerger une vision du monde par une synchronisation de leur réaction à un même stimulus, restructurant leurs circuits dans une sorte de plasticité.

Nous identifions ainsi quatre singularités :

- une organisation modulaire du système visuel : voir est une tâche complexe qui nécessite de nombreuses opérations se décomposant en opérations plus élémentaires.
- un traitement massivement parallèle : parallélisme à la fois dans le temps (les opérations sont effectuées en même temps sur une même zone de la scène visuelle) et dans l'espace (les opérations sont effectuées en différents endroits du système visuel)
- un couplage plus ou moins précis dans le temps de caractéristiques traitées en parallèle dans des zones parfois fort éloignées l'une de l'autre, comme code d'une reconnaissance d'objets de la scène visuelle, dans une sorte de hiérarchie de niveaux d'abstraction croissants.
- une plasticité cérébrale comme moteur d'apprentissage.

Partie 2 - Chapitre 3 - Mécanismes artificiels de la vision

Nous identifions des modes de représentation de l'image, du dessin au trait, dont nous montrons qu'ils caractérisent des classes de méthodes :

1 • Méthodes morphologiques : on code l'image comme un ensemble de pixels d'un plan (euclidien \mathbb{R}^2 ou discret \mathbb{Z}^2) sur lequel on va faire agir des transformations.

Le présumé est : un acte perceptif visuel peut se modéliser par des transformations ensemblistes.

2 • Méthodes statistiques : on code l'image comme un vecteur de \mathbb{R}^p dont les composantes mesurent p caractéristiques définissant l'image.

Le présumé est : il y a beaucoup de chances pour que ...

3 • Méthodes structurelles : on code l'image comme une structure informatique.

Le présumé est : il y a une forme, une structure dans l'image.

4 • Méthodes analytiques : on code l'image par une équation sur les coordonnées de ses points dans un espace.

Le présumé est : il y a vérification d'une relation numérique.

5 • Méthodes connexionnistes : on code l'image comme une configuration d'état d'un ensemble d'unités interconnectées.

Le présumé est : il y a des liens.

Nous détaillons plus particulièrement les modèles connexionnistes, en proposant une classification de ceux-ci, en montrant leurs forces, leurs faiblesses et en précisant les modèles physiques les ayant inspirés.

Nous observons ici que le mode de représentation de l'image enferme dans une classe de méthodes. Chacune de ces classes est étudiée, les algorithmes précisés.

Partie 2 - Chapitre 4 - Classification de modèles de l'apprentissage.

Nous classifions des modèles de l'apprentissage selon cinq points de vue qui focalisent cinq définitions :

Apprentissage (au sens béhavioriste) : Modification stable du **comportement** imputable à l'**expérience**.

Apprentissage (au sens gestaltiste) : Modification stable de l'**organisation**, imputable à la **capacité à percevoir globalement des objets**.

Apprentissage (au sens cognitiviste) : Modification stable des **structures mentales internes**, imputable à une **assimilation - accommodation**.

Apprentissage (au sens connexionniste) : Modification stable des **poids des connexions entre les unités**, imputable à une adaptation à une réponse imposée.

Apprentissage (au sens formel) : Modification stable d'un **modèle mathématique sophistiqué**, imputable à l'élaboration d'un **calcul**.

Dans cette perspective :

apprendre c'est converger	(cadre des probabilités)
apprendre c'est décoder	(cadre de la cryptographie)
apprendre c'est comprimer	(cadre de la réduction de la quantité d'information pour une augmentation de sa qualité)

Nous passons en revue ces différentes sortes d'apprentissage.

Partie 2 - Chapitre 5 - Modélisation de systèmes complexes.

Nous ne cherchons pas ici, à présenter de façon exhaustive les techniques de modélisation, mais tout modestement, à illustrer, dans un cas particulier, une démarche qui porte un regard sur des systèmes complexes.

Nous identifions neuf niveaux de description d'un phénomène dans une perspective de modélisation : être, faire, se réguler, s'informer, décider, mémoriser, coordonner, imaginer, se finaliser.

Nous présentons un modèle de la coopération et un modèle de l'organisation.

Plan

Les travaux de cette thèse s'inscrivent dans le cadre du projet PASTIS, d'étude d'une machine qui explore sur un jeu de données graphiques, un itinéraire de compréhension de dessin au trait, en adaptant et en optimisant pour chaque niveau, l'architecture à la complexité, le temps à l'espace, la redondance à l' à peu près.

Ce mémoire étudie plus particulièrement :

- une approche par le biais de la complexité de Kolmogorov, du passage du numérique au symbolique et de l'explosion combinatoire dans la création et le parcours des bases de connaissances.
- des procédés de traitement de bas niveau sur des figures d'une rétine permettant de faire émerger des niveaux de structure des dessins, c'est à dire d'en faire apparaître une compréhension symbolique.
- des procédés généraux de vision dont nous montrons qu'ils découlent du mode de codage du dessin.
- des procédés généraux d'apprentissage, dont nous montrons qu'ils découlent de manières de voir le monde.
- des procédés de modélisation de systèmes complexes, vus à la lumière de l'approche systémique.

Ces procédés seront mis en œuvre pour la réalisation de la machine PASTIS.

Les chapitres de ce mémoire résumés dans ce guide sont :

Partie 2 - chapitre 1. La machine PASTIS.

où l'on présente plus précisément certaines notions.

Partie 2 - chapitre 2. Mécanismes humains de la vision.

où l'on survole les modèles neurobiologiques de la vision.

Partie 2 - chapitre 3. Vision et dessins.

où l'on présente et discute des mécanismes artificiels.

Partie 2 - chapitre 4. Apprentissage.

où l'on classe les modèles.

Partie 2 - chapitre 5. Modélisation des systèmes complexes.

où l'on introduit l'approche systémique.

Nous résumons maintenant ces différents chapitres :

Partie 2 - Chapitre 1 - La machine PASTIS

où l'on s'adapte aux contraintes technologiques

La recherche que nous présentons, s'inscrit initialement dans un projet global appelé PASTIS.

Nous décrivons dans ce chapitre les objectifs de ce projet.

Le projet PASTIS se propose d'étudier une machine qui explore, sur un jeu de données graphiques, un itinéraire de compréhension de dessin, en adaptant et en optimisant pour chaque niveau : l'architecture à la complexité, le temps à l'espace, la redondance à l' à peu près.

L'objectif général de ce projet est de concevoir et fonder des modules architecturaux spécialisés pour l'apprentissage et la reconnaissance de "dessins au trait". Ces modules seront rapides parce qu'approximatifs, et utilisés aussi bien pour des données de bas niveau que pour des données symboliques. PASTIS sera une machine qui évolue dans le temps.

Son comportement quotidien alterne des périodes de veille et de rêve :

- en état de veille, des dessins sont présentés à PASTIS, ils sont analysés par l'action en parallèle des différents modules architecturaux, et PASTIS se forge une croyance sur le dessin. Cette croyance est infirmée ou confirmée par un dialogue basé sur les mots que PASTIS a déjà appris. Ceci renforce la partie "comprise" de sa connaissance et fait évoluer la partie "incomprise".
- en état de rêve, PASTIS explore la partie non structurée de sa connaissance pour détecter des régularités et ainsi faire émerger des concepts qui seront nommés par dialogue dès le réveil.

Deux aspects se distinguent lorsqu'il s'agit de mettre en œuvre effectivement un procédé mécanique sur une machine parallèle non encore construite :

- sa simulation sur une machine classique séquentielle,
- sa simulation effective partielle sur un composant modulaire de la machine complète,

Un module architectural de rotation rapide d'une figure F d'une rétine a été développé, dans le cadre d'une thèse présentée par Jérôme DELEU.

Les autres développements sont pour l'instant des simulations sur des ordinateurs classiques séquentiels.

Partie 2 - Chapitre 2 - les mécanismes humains de la vision

où l'on survole les modèles neurobiologiques de la vision

Rappelons que notre but n'est pas d'étudier la cognition (nous ne sommes pas spécialistes), mais d'en utiliser certaines connaissances pour éclairer notre démarche. Il faut savoir que bon nombre de points de vue évoqués de façon lapidaire sont en fait discutables et discutés par les spécialistes.

Le cerveau, organe de la pensée, est souvent associé au tout dernier progrès technique. Ce modèle "mécanique" est toujours faux.

Contrairement à un ordinateur, le cerveau n'a pas été construit à des fins précises ni conformément à des principes bien établis, mais s'est constitué par une sélection naturelle dans une longue évolution de perfectionnements successifs.

Les travaux des neurobiologistes permettent d'entrevoir des explications des mécanismes cérébraux.

Ceux-ci décrivent l'organisation du cortex en termes de régions, d'aires spécialisées dans une fonction précise : détection de formes, de lignes ; détection de couleurs ; détection de mouvement ; ... Cette notion de spécialisation fonctionnelle conforte l'idée d'une organisation modulaire du système visuel : voir est une tâche complexe qui nécessite de nombreuses opérations se décomposant en opérations plus élémentaires.

Chacune de ces aires socialise son activité avec celle des autres pour faire émerger une vision du monde.

L'information visuelle pré-traitée dans la rétine, atteint le cortex au niveau de l'aire primaire notée V1, elle se sépare alors en au moins quatre voies perceptives spécialisées, traitant ainsi en parallèle différentes caractéristiques du stimulus.

Ce parallélisme est à la fois un parallélisme dans le temps (les opérations sont effectuées en même temps sur une même zone de la scène visuelle) et un parallélisme dans l'espace (les opérations sont effectuées en différents endroits du système visuel).

Le modèle admis est donc celui d'une dissociation fonctionnelle en aires spécialisées dans le traitement en parallèle d'une caractéristique particulière de la perception visuelle.

Un autre aspect est celui de la synchronisation des émissions des différentes aires spécialisées, réagissant à un même stimulus.

Ces couplages dans le temps, plus ou moins précis, de caractéristiques traitées en parallèle dans des aires parfois fort éloignées l'une de l'autre, serait le code de la reconnaissance d'objets de la scène visuelle, dans une sorte de hiérarchie de niveaux d'abstraction croissants.

Un dernier aspect est celui de la plasticité cérébrale restructurant les circuits cérébraux pour un apprentissage.

Ces résultats neurobiologiques mettent ainsi en évidence quatre singularités de la "machine cérébrale" :

- une organisation modulaire,
- un traitement massivement parallèle,
- un couplage plus ou moins précis dans le temps comme code d'une reconnaissance,
- une plasticité des circuits comme moteur de l'apprentissage.

Partie 2 - Chapitre 3 - Vision et dessins

où l'on présente et discute des mécanismes artificiels

Pour présenter le domaine de la vision des mécanismes artificiels, nous proposons un bref état des recherches en systèmes de vision, qui nous permet d'identifier :

d'une part un objet d'étude : l'image,

d'autre part deux tâches essentielles : la segmentation et l'interprétation.

Ces tâches mettent en œuvre des outils et des connaissances organisés en systèmes.

Nous allons nous intéresser alors à l'image pour identifier des modes de représentation, dont nous montrons qu'ils caractérisent des classes de méthodes.

Les représentations d'une image se répartissent en **cinq** catégories caractérisées par des classes de méthodes :

1 • Méthodes morphologiques : on code l'image comme un ensemble de pixels d'un plan (euclidien R^2 ou discret Z^2) sur lequel on va faire agir des transformations.

Le présumé est : un acte perceptif visuel peut se modéliser par des transformations ensemblistes.

2 • Méthodes statistiques : on code l'image comme un vecteur de R^p dont les composantes mesurent p caractéristiques définissant l'image.

Le présumé est : il y a beaucoup de chances pour que ...

3 • Méthodes structurelles : on code l'image comme une structure informatique.

Le présumé est : il y a une forme, une structure dans l'image.

4 • Méthodes analytiques : on code l'image par une équation sur les coordonnées de ses points dans un espace.

Le présumé est : il y a vérification d'une relation numérique.

5 • Méthodes connexionnistes : on code l'image comme une configuration d'état d'un ensemble d'unités interconnectées.

Le présumé est : il y a des liens.

Nous observons ici que le mode de représentation de l'image enferme dans une classe de méthodes.

Nous étudions alors chacune de ces classes de méthodes :

1 • Méthodes morphologiques : on code l'image comme un ensemble de pixels d'un plan (euclidien R^2 ou discret Z^2) sur lequel on va faire agir des transformations.

Le présumé est : un acte perceptif visuel peut se modéliser par des transformations ensemblistes.

Le problème de la reconnaissance d'une figure dans une image est traité de la manière suivante

Comme les signaux visuels interagissent de manière ensembliste :

- Une forme F d'une image E , ensemble de pixels, sera modélisée par un sous ensemble de E ,

- Un acte perceptif sera modélisé par une transformation de E vers E , manière d'agir sur les pixels selon un point de vue identifié par un élément structurant.

Cet élément structurant associé à un pixel P de l'image est un ensemble de pixels souvent voisins de P . Il modélise une manière de modeler l'image.

Ces transformations vont permettre de faire apparaître des structures dans l'image, éléments de leur description.

2 • Méthodes statistiques : on code l'image comme un vecteur de \mathbb{R}^p dont les composantes mesurent p caractéristiques définissant l'image.

Le pré-supposé est : il y a beaucoup de chances pour que ...

Le problème de la reconnaissance d'une figure dans une image est traité de la manière suivante :

Comme reconnaître une figure, c'est identifier un cas particulier de cette figure parmi de nombreux exemples de cette figure :

- Une forme F d'une image E , ensemble de pixels, sera modélisée par un objet d'un espace de représentation (point, droite,...) à classer de manière automatique dans un espace de classification (partition, recouvrement, hiérarchie,...).
- Un acte perceptif sera modélisé par une action de classification automatique de cet objet, action matérialisée par un algorithme.

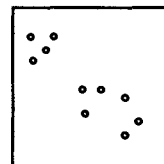
Pour réaliser ceci, nous disposons :

- de différents espaces de classification et modes de représentation.
Nous définissons ainsi les partitions, les recouvrements, les hiérarchies et les pyramides, éléments de base d'espaces de classification.
Nous évoquons un centre de gravité, une droite de régression, un point d'une classe, une distance, un axe factoriel ou discriminant, une variable, une loi de probabilité, une courbe, ... comme éléments de base d'espaces de représentation de classes d'objets.
Pour rendre compte du caractère correct de ces représentations, nous définissons une mesure D de dissemblance entre une classe et une représentation de classe et une mesure d'adéquation W entre une partition et une représentation de partition.
- d'un algorithme de classification automatique appelé "nuées dynamiques". Ce procédé mécanique de calcul cherche à optimiser un critère W qui exprime l'adéquation entre une classification P d'objets et un mode de représentation L des classes de cette classification. Le problème d'optimisation se pose en terme de recherche simultanée de la classification et de représentations possibles, qui optimisent ce critère.

Par exemple :

Prenons une rétine R isomorphe à $[1, n]^2$ et le problème de la classification automatique de groupes de pixels de cette rétine en k classes appelées figures F .

Dans l'exemple de la rétine ci-contre en 2 classes.



Nous pouvons choisir comme espace de représentation R_F d'une figure F , la rétine elle-même : $R_F=R$. c'est à dire que nous allons représenter une figure F de la rétine perçue comme une classe de pixels par un point G de la rétine. Ce point peut être le centre de gravité discret de la figure F , ou un point représentatif de la figure F .

Nous pouvons choisir comme espace de représentation R_P d'une partition en k classes, l'ensemble $R_P=R \times R \dots \times R = R^k$. Une classe de la partition est ainsi représentée par un pixel représentatif de la rétine, le k uplet des k classes de la partition, par un k uplet de pixels de la rétine.

Nous pouvons choisir pour rendre compte des rapports entre objets et représentations de ces objets :

Une fonction g dite de représentation qui permet d'associer à toute partition $P_k=(F_1, \dots, F_k)$ de la rétine, sa représentation $L_k = g(P)$ dans R_P par $L_k=(G_1, \dots, G_k)$ soit les k centres de gravité discrets des k classes de la partition P_k .

Une fonction f dite d'affectation permet d'associer à toute représentation L_k une partition P_k lui correspondant. $f(L_k)=P_k$ où les figures F_k de P_k sont définies comme les pixels M de la rétine les plus proches de G_k que de tous les autres G_i

Une fonction D de dissemblance entre la figure F et sa représentation G comme l'inertie de F par rapport à G : $D(F, G) = \text{inertie de F par rapport à G}$.

Une fonction W d'adéquation entre une partition P_k formée de figures F_k et une représentation L_k de partition formée de points G_k comme somme des dissemblances

entre les F_k et les G_k :
$$W(P_k, L_k) = \sum_{n=1}^k D(F_n, G_n)$$

L'algorithme permet de trouver une partition P^n et une représentation L^n de la façon suivante :

à partir d'une solution initiale $V_0 = (P^0, L^0)$ estimée ou tirée au hasard itérer jusqu'à stabilisation de la suite V par : $V_{n+1} = (P^{n+1}, L^{n+1})$ avec $L^{n+1} = g(P^n)$ et $P^{n+1} = f(L^n)$

On prouve que la suite u définie par $u_n = W(P^n, L^n)$ est décroissante convergente et V stationnaire à partir d'un rang fini.

Cet algorithme souffre de certaines insuffisances : la solution obtenue dépend de la représentation et de la partition choisies initialement et nécessite le choix a priori d'un nombre k de classes. Le problème du choix du nombre k de classes n'est pas encore résolu et fait l'objet d'une recherche dans notre université.

3 • Méthodes structurelles : on code l'image comme une structure informatique.

Le présupposé est : il y a une forme, une structure dans l'image.

Le problème de la reconnaissance d'une figure dans une image est traité de la manière suivante :

Comme l'image pour être stockée dans une mémoire d'ordinateur doit être codée par une structure de données informatique

- Une forme F d'une image E, ensemble de pixels, sera modélisée par une structure de données
- Un acte perceptif sera modélisé par une transformation, une action, une fonction sur cette structure de données.

•1• La structure de données est une chaîne, un mot, une liste de composants élémentaires pris dans un alphabet.

Dans l'ensemble de toutes les chaînes codant une forme, un concept est une classe de motifs F qui se "ressemblent".

• un premier aspect est de modéliser la relation de "ressemblance" entre motifs.

Nous disposons de différents algorithmes :

- CC recherche d'un motif M dans une forme chaîne F. [MORRIS PRATT KNUTH]
On manipule un indicateur P qui, comme son nom l'indique, indique la longueur du plus long préfixe du motif M reconnu comme sous chaîne de F au cours de l'exploration. Une fonction Lien qui ne dépend que du motif M permet de faire évoluer P.
- DE calcul d'une distance entre deux motifs M_1 et M_2 à une certaine confusion près [WAGNER FISCHER]

Trois opérations élémentaires sont définies qui modélisent la confusion autorisée :
la substitution d'un élément a du motif par un autre b de coût : $C(a, b)$;
l'insertion d'un élément novateur a dans le motif de coût $C(\cdot, a)$;
la destruction d'un élément a du motif de coût $C(a, \cdot)$.

La distance $d(M_1, M_2)$ entre deux motifs M_1 et M_2 est définie comme le coût de la suite de transformations la moins coûteuse pour transformer M_1 en M_2

Le processus de calcul de cette distance est un processus de programmation dynamique.

• **un second aspect est de modéliser le concept, sous ensemble de mots, par un langage.**

Nous disposons de différents algorithmes bien connus sur les langages réguliers ou algébriques :

- AS analyse syntaxique

Reconnaissance de l'appartenance d'un mot à un langage.

- IR inférence régulière

Étant donné un ensemble fini de mots EA (échantillon d'apprentissage), trouver une grammaire régulière G d'un langage L acceptant les mots de EA.

•2• **La structure de données est un terme, une formule, cas particulier d'arbre.**

Un terme est représenté par un arbre étiqueté par des symboles soit d'un ensemble F, soit d'un ensemble V. Les symboles de $F = \{ f, g, a, \dots \}$ sont interprétés comme un nom de fonction à n arguments ($n \geq 0$) ; les symboles de $V = \{ x, y, \dots \}$ sont interprétés comme des noms de variables. Exemple : $f(g(x), a, h(g(y), z))$

Nous disposons de différents algorithmes sur ces termes :

- S substituer

On appelle substitution **sigma** toute fonction de V dans l'ensemble des termes, qui à chaque variable de V associe le terme qui doit lui être substitué.

Substituer (t, **sigma**) c'est générer un terme dans lequel toute occurrence des variables est remplacée par son terme remplaçant selon les indications de la substitution **sigma**.

- AU antiunifier : c'est l'algorithme de base de l'apprentissage symbolique

Antiunifier (t_1, t_2) c'est générer un terme "généralisant" au mieux deux termes donnés t_1 et t_2

si les deux termes t_1 et t_2 sont des variables ou si $\text{racine}(t_1) \neq \text{racine}(t_2)$ alors on généralise t_1 et t_2 par une variable t de nom arbitraire, sinon on a $\text{racine}(t_1) = \text{racine}(t_2)$ et le généralisé aura pour racine cette racine commune.

- F filtrer : c'est l'algorithme de base de la reconnaissance de motif

Filtrer (t, u) c'est générer une substitution **sigma** telle que

$$\text{Substituer}(t, \text{sigma}) = u$$

En fait Filtrer est l'opération inverse de substituer. Si le filtrage n'est pas possible **Filtrer** (t, u) fournit la substitution non définie nil. On génère la substitution **sigma** par ajouts successifs de couples (variable x, terme v). Si l'on obtient à la fois les couples (x, v) et (x, v') alors il faut vérifier $v = v'$ sinon on a un échec.

- U unifier : généralisation du filtrage le rendant symétrique

unifier (t_1, t_2) c'est générer une substitution **sigma** telle que

$$\text{sigma}(t_1) = \text{sigma}(t_2) = t$$

Le terme t est dit unifié de t_1 et t_2

Si les deux termes t_1 et t_2 sont différents

alors on détermine une disparité (x, v) entre t_1 et t_2 ,

si x est une variable alors on cherche à unifier les termes s_1 et s_2 obtenus par :

s_1 est la substitution dans t_1 de la variable x par le terme v et

s_2 est la substitution dans t_2 de la variable x par le terme v

sinon l'unification est impossible.

•3• **La structure de données est un arbre**

Dans l'ensemble de tous les arbres codant une forme, un concept est une classe de motifs arborescents F qui se "ressemblent".

• **un premier aspect est de modéliser la relation de "ressemblance" entre motifs.**

- DA calcul d'une distance entre deux arbres motifs M_1 et M_2 à une certaine confusion près.

Comme pour les chaînes on met en place un processus de programmation dynamique pour évaluer un coût minimal d'une suite de transformations faisant évoluer M_1 en M_2 .

• **un second aspect est de modéliser le concept, sous ensemble d'arbres par un langage.**

- AS analyse syntaxique

Reconnaissance par automate d'arbres de l'appartenance d'un arbre à un langage d'arbre.

- IR inférence régulière

Étant donné un ensemble fini d'arbres EA (échantillon d'apprentissage), trouver un automate d'arbres reconnaissant les arbres de EA.

•4• **La structure de données est un graphe**

- DG calcul d'une distance entre deux graphes motifs M_1 et M_2 à une certaine confusion près.

La complexité d'un tel algorithme est vraisemblablement exponentielle.

- MG "matching" entre deux graphes.

Nous présentons quatre problèmes et quatre classes de méthodes : méthodes métriques de mise en correspondance ; méthodes de recherche par retour arrière ; méthodes utilisant un graphe de mise en correspondance ("association graph") ; méthodes spécifiques aux graphes planaires.

4 • Méthodes analytiques : on code l'image par une équation sur les coordonnées de ses points dans un espace.

Le présumé est : il y a vérification d'une relation numérique.

Le problème de la reconnaissance d'une figure dans une image est traité de la manière suivante :

- Une forme F d'une image E, ensemble de pixels, sera modélisée par une équation sur les coordonnées de ses points.
- Un acte perceptif sera modélisé par une mesure d'erreur, une sorte de "distance" entre deux courbes..

5 • Méthodes connexionnistes : on code l'image comme une configuration d'état d'un ensemble d'unités interconnectées.

Le présumé est : il y a des liens.

Le problème de la reconnaissance d'une figure dans une image est traité de la manière suivante :

- Une forme F d'une image E, ensemble de pixels, sera modélisée par une configuration d'états d'un système d'unités interconnectées.
- Un acte perceptif sera modélisé par une réponse du système à une sollicitation.

Nous proposons une grille de description d'un système connexionniste, en le définissant par **cinq** attributs caractéristiques :

- 1 • un **nom** qui identifie le système
- 2 • une **architecture** :

Le système est constitué d'un nombre fini N d'**unités connectées** entre elles.

Une **unité** :

- est identifiée par un numéro i de 1 à N.
- est connectée en entrée et en sortie à d'autres unités
- possède à l'instant t, un état interne e_i . Cet état est soit discret (0 ou 1) soit continu dans un intervalle.
- possède une fonction de transition f_i . Cette fonction réalise un bilan des entrées pour fournir un résultat aux unités connectées en sortie.

Les **connexions** :

- sont identifiées par un couple (i, j) de numéros d'unité. La connexion (i, j) relie l'unité i à l'unité j.
- sont pondérées par un réel $p_{i, j}$. Ces pondérations sont regroupées dans une matrice N x N des influences.

- 3 • une **dynamique** :

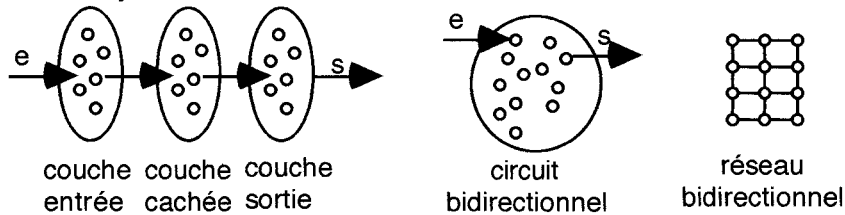
- en exploitation : il s'agit alors de fournir une réponse à une sollicitation
- en apprentissage : il s'agit alors d'adapter les influences pour que la réponse calculée corresponde à la réponse voulue à la sollicitation.

- 4 • une **inspiration** :

C'est à dire une référence à un phénomène extérieur qui guide la compréhension du fonctionnement du système.

- 5 • un ou des **indicateurs** associés à un état du système, permettant de juger de son évolution dans le temps.

Deux types de systèmes connexionnistes apparaissent : ceux agencés en couches successives, l'influence y est unidirectionnelle et ceux agencés de façon homogène en circuits, l'influence y est bidirectionnelle.



Les systèmes connexionnistes unidirectionnels :

Nom : Perceptron

Architecture : en couches

Couche :	Entrée	Cachée	Sortie
Nom :	Rétine	Association	Décision
unités	nb	N^2	N
	e_i	{0, 1}	{0, 1}
	f_i	identique	spéciale

Matrice P des influences : c'est un vecteur [1..N] des influences entre la couche d'association et la couche de décision.

Dynamique :

en exploitation : application à l'instant t des fonctions de transition selon la sollicitation e en entrée.

en apprentissage : dynamique de ROSENBLATT : modification des influences par apprentissage supervisé par correction d'erreur optimisant le critère de gradient J(P) :

$$J(P) = \sum_{e \in M} - P \cdot e \quad \text{où } M \text{ est l'ensemble des exemples "e" mal classés par le perceptron}$$

de matrice d'influence P.

Cette modification pour la composante i du vecteur P est : $\Delta P_i = \mu(t) \cdot [S^d - S^c] \cdot e_i$

où $\mu(t)$ est un coefficient d'apprentissage fonction du temps t ; S^d et S^c les sorties désirées et calculées à la sollicitation e.

Inspiration : simuler une assemblée de neurones humains.

Indicateur : J(P)

Nom : Adaline

Architecture : en couches

Couche :		Entrée	Sortie
Nom :		Rétine	Adaptation
unités	nb	N	une
	e _j	continu	continu
	f _j	identique	

Matrice P des influences : c'est un vecteur [1..N] des influences entre la couche rétine et la couche adaptation.

Dynamique :

en exploitation : application à l'instant t des fonctions de transition selon la sollicitation e en entrée.

en apprentissage : dynamique de WIDROW HOFF : modification des influences minimalisant un critère de type énergie pour une entrée "e" : J(P, e) :

$J(P, e) = [P \cdot e - S^d]^2$ où S^d est la sortie désirée pour l'entrée e

Cette modification pour la composante i du vecteur P est :

$\Delta P_i = 2 \cdot J(P, e) [S^d - \sum_k P_k \cdot e_k] \cdot e_i$ où S^d est la sortie désirée

Inspiration : bâtir des adaptateurs en jouant sur les influences.

Indicateur : J(P, e)

Nom : Filtre linéaire

Architecture : en couches

Couche :		Entrée	Sortie
Nom :		Rétine	
unités	nb	Ne	Ns
	e _j	continu	continu
	f _j	identique	identique

Matrice P des influences : c'est un vecteur P [1..Ne, 1..Ns] des influences entre la couche rétine et la couche de sortie.

Dynamique :

en exploitation : application à l'instant t de la formule $S = P \cdot E$ où E est le vecteur [1..Ne] des Ne états des unités d'entrée.

en apprentissage : Imposition d'une capacité innée : On dispose de k exemples prototypes que l'on désire "apprendre" au filtre linéaire. Soit AE le vecteur [1..k, 1..Ne] des k exemples de configurations d'entrée placées en colonnes et AS le vecteur [1..k, 1..Ns] des k exemples de configurations de sortie correspondantes placées dans le même ordre en colonne. Toute solution X de l'équation matricielle $AS = X \cdot AE$ est une matrice [1..Ne, 1..Ns] qui peut être prise comme matrice P des influences.

Cette équation peut être résolue exactement par la matrice pseudo inverse de PENROSE, soit de manière approchée : approximation de KOHONEN ou approximation de WIDROW HOFF.

Inspiration : Utiliser les résultats mathématiques de l'algèbre linéaire.

Indicateur : aucun.

Les systèmes connexionnistes bidirectionnels :

Nom : Réseau de HOPFIELD

Architecture : circuit bidirectionnel complètement connecté : **connexions symétriques**

unités	nb	N
	e_i	{-1, 1}
	f_i	semi linéaire à seuil μ_i

Matrice P des influences : c'est un vecteur [1..N, 1..N] des influences entre chaque unité

On notera U le vecteur [1..N] des seuils de transition μ_i des unités et E le vecteur [1..N] des états des unités du réseau

Dynamique :

en exploitation : application d'une procédure de récurrence jusqu'à stabilisation du réseau :

Tant que pas stable

faire

- choisir avec la probabilité uniforme le numéro i de l'unité à actualiser
- calculer l'état suivant de l'unité numéro i

fait

Pour calculer l'état suivant, on évalue, selon une certaine densité de probabilité qui peut dépendre d'un facteur de bruit b ou d'une "température" T, la probabilité de changement d'état de l'unité numéro i. Si cette probabilité dépasse un seuil de confiance fixé à l'avance alors l'état de l'unité n° i est changé sinon il reste identique.

Pour une dynamique de recuit simulé :

Répéter

relaxer : laisser évoluer selon une dynamique régie par une température T

refroidir : réduire T selon une loi géométrique de raison 0,9

Jusqu'à ce que le réseau soit gelé : (refroidissement n'entraînant plus aucune modification)

en apprentissage : dynamique de HEBB :

- 1• on commence avec des connexions toutes nulles (hypothèse de la table rase)
- 2• on force le système dans un état particulier E^S
- 3• on examine tous les couples (i, j) d'unités et on modifie $P_{i,j}$ de $\Delta P_{i,j}$ selon la règle de HEBB :

si $e_i^S = +1$ et $e_j^S = +1$ alors $\Delta P_{i,j} = +1$ (unités simultanément actives : on renforce)

si $e_i^S = +1$ et $e_j^S = -1$ alors $\Delta P_{i,j} = -1$ (unités opposées : on affaiblit)

si $e_i^S = -1$ et $e_j^S = +1$ alors $\Delta P_{i,j} = -1$ (unités opposées : on affaiblit)

si $e_i^S = -1$ et $e_j^S = -1$ alors $\Delta P_{i,j} = +1$ (unités simultanément inactives : on renforce)

Inspiration : physique statistique : interaction entre deux spins moins intervention d'un champ extérieur. Les dynamiques d'actualisation du réseau reposent sur des modèles de relaxation isotherme (GLAUBER), de MONTE CARLO, de recuit simulé (KIRPATRICK),...

Indicateur : $J(P, e) = \frac{1}{2} \sum_{i \neq j} P_{i,j} \cdot e_i \cdot e_j - \sum_i U_i \cdot e_i - \frac{1}{2} \sum_{i \neq j} P_{i,j} \cdot e_i \cdot e_j$ représente

l'interaction entre deux spins et $\sum_i U_i \cdot e_i$ l'intervention d'un champ extérieur.

Citons aussi la machine de BOLTZMANN qui introduit une innovation par rapport au réseau de HOPFIELD en distinguant deux catégories d'unités : les unités visibles et les unités cachées.

La dynamique d'apprentissage de BOLTZMANN cherche à équilibrer la probabilité d'évolution du réseau en phase libre avec la probabilité d'évolution en phase contrainte, c'est à dire lorsque l'on impose aux unités visibles de sortir de prendre l'état de sortie désiré en fonction de l'état d'entrée donné.

Ces exemples "historiques" montrent bien d'une part :

- pour les réseaux unidirectionnels : une dynamique d'exploitation ultra simple et des problèmes d'apprentissage.
- pour les réseaux bidirectionnels : des dynamiques d'exploitation sophistiquées et des dynamiques d'apprentissage variées : du plus simple (HEBB) à des modèles probabilistes (BOLTZMANN)

D'une part des avancées ont été réalisées sur les dynamiques d'apprentissage :

- dans le cadre des réseaux en couches, pour la détermination par apprentissage de matrices P d'influences de moindre norme :
 - détermination de P par projection orthogonalisation : algorithme de GRAMM SCHMIDT
 - détermination de P par pseudo inverse : algorithme de GREVILLE et algorithme de KABALA RASAKLOO
 - détermination de P par rétro propagation du gradient : algorithme de Le CUN.
- dans le cadre des réseaux bidirectionnels :
 - détermination de P guidée par deux objectifs : avoir toutes les unités en équilibre et avoir les sorties au niveau désiré, par un indicateur : une distance par rapport à l'objectif, par une dynamique : répartir itérativement les déséquilibres et erreurs par coopération entre les unités : s'aider soi-même et aider les autres.

D'autre part des avancées ont été réalisées sur les architectures ou les modèles:

- réseaux organisés en colonnes et en aires de BURNOD : architecture en colonnes ayant trois niveaux d'activation et trois couches, regroupées en aires. L'apprentissage consiste à réaliser des couplages, découplages entre colonnes selon des règles inspirées de la règle de HEBB.
- réseaux simpliciaux inspirés par les objets "mathématiques" complexes simpliciaux (COSNES)
- réseaux auto adaptatifs de KOHONEN régis par des équations différentielles, l'apprentissage n'est pas supervisé, il y a sélection de l'unité qui réagit le plus à une sollicitation et augmentation de l'activité de cette unité et des unités du groupe qui l'entoure, ce qui amène des corrélations naturelles.

Partie 2 - Chapitre 4 - Apprentissage

où l'on classe les modèles

Nous classons les modèles de l'apprentissage selon cinq points de vue qui focalisent cinq définitions :

Apprentissage (au sens behavioriste) : Modification stable du **comportement** imputable à l'**expérience**.

Apprentissage (au sens gestaltiste) : Modification stable de l'**organisation**, imputable à la **capacité à percevoir globalement des objets**

Apprentissage (au sens cognitiviste) : Modification stable des **structures mentales internes**, imputable à une **assimilation - accommodation**.

Apprentissage (au sens connexionniste) : Modification stable des **poids des connexions entre les unités**, imputable à une adaptation à une réponse imposée

Apprentissage (au sens formel) : Modification stable d'un **modèle mathématique sophistiqué**, imputable à l'élaboration d'un **calcul**

Dans cette perspective :

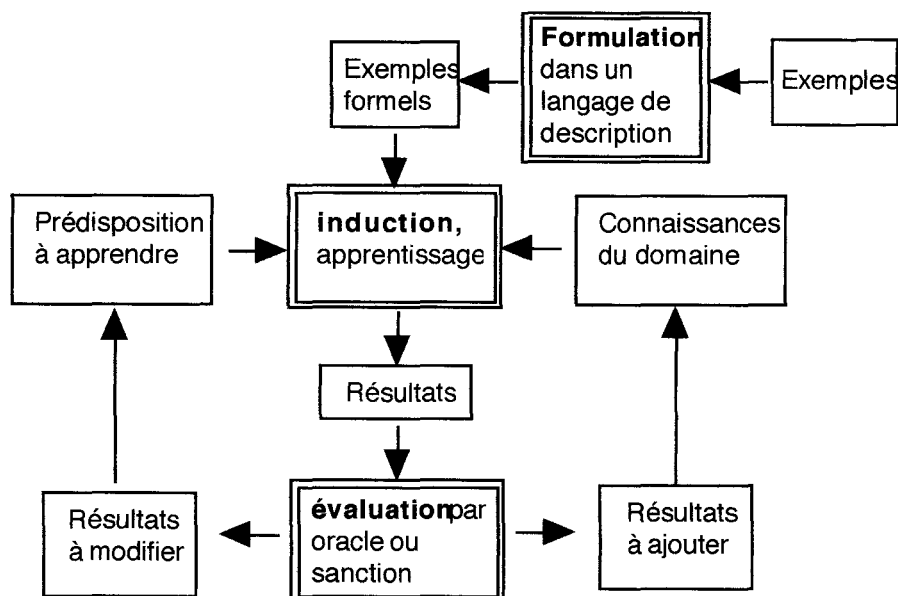
apprendre c'est converger (cadre des probabilités)

apprendre c'est décoder (cadre de la cryptographie)

apprendre c'est comprimer (cadre de la réduction de la quantité d'information pour une augmentation de sa qualité)

• 1 • Le modèle comportemental

Le modèle comportemental peut être résumé par le schéma suivant :



• 2 • *Le modèle gestaltiste*

L'apprentissage est une amélioration de l'**organisation**, apparaissant comme une propriété émergente de perceptions de formes, à partir d'une capacité à distinguer les objets qui résulte d'une structuration accomplie par le système nerveux.

• 3 • *Le modèle cognitiviste*

Les modèles piagetiens identifient quatre concepts : le concept d'assimilation, d'accommodation, d'équilibration et, de structure logico algébrique, au travers de deux principes :

- 1 • Tout schème (structure) tend à **assimiler** (c'est à dire à s'incorporer les éléments extérieurs compatibles avec sa nature).
- 2 • Tout schème (structure) tend à **s'accommoder** aux éléments qu'il assimile (c'est à dire à se modifier pour tenir compte de la nouveauté sans perdre la mémoire du passé).

Un stade est un système général de pensée obéissant au principe d'équilibration. PIAGET distingue ainsi quatre stades : le stade sensori-moteur (4 à 18 mois); le stade relationnel (18 mois à 5 ans); le stade dimensionnel (5 à 10 ans); le stade vectoriel ou abstrait (11 à 18 ans)
L'évolution cognitive apparaît comme étant la complexification des mécanismes de traitement de l'information.

• 4 • *Le modèle connexionniste*

L'apprentissage pour un système est la propriété de modification de son comportement en fonction de son histoire, c'est à dire la capacité d'acquérir de nouveaux comportements ou d'en modifier d'anciens par des expériences répétées.

Le problème de l'apprentissage pour un système connexionniste formé d'unités, est celui de la modification des poids synaptiques des connexions, formalisant les influences réciproques, dans le but de faire réaliser au système la tâche souhaitée.

Le problème étant reformulé en termes de modification, de transformation, d'évolution d'une matrice notée P d'influences entre les unités, il entre dans de nombreux cadres formels.

Classons les selon les types d'apprentissage :

Imposition d'une capacité innée : le système est doté dès le départ de la capacité, elle n'est donc pas enseignée. La matrice P est obtenue par résolution préalable d'une "équation".

Apprentissage supervisé : c'est à dire apprentissage avec un tuteur supposé posséder la connaissance à transmettre. La matrice P est obtenue par itération, comme résultat d'un processus organisé en étapes successives correspondant à la présentation d'exemples associant une entrée à une sortie désirée, le plus souvent fondé sur l'optimisation d'un indicateur interprétable physiquement comme une énergie, une entropie, une inertie, un potentiel,....

Apprentissage non supervisé : c'est à dire un apprentissage compétitif, sans tuteur, qui renforce pour tout exemple présenté au système les liens entre les unités qui reconnaissent le mieux cet exemple. La matrice P rend compte étape après étape du renforcement de ces liens.

• 5 • *Le modèle formel*

L'apprentissage est envisagé comme une modification répétée de l'état d'un dispositif calculatoire appelé "apprenti", modification elle même d'ordre calculatoire.

Le présupposé de l'approche formelle est de déclarer que des niveaux d'explication formels et fonctionnels sont pertinents pour étudier un phénomène comme l'apprentissage, et fonctionnent de manière autonome.

Si apprendre c'est d'une manière ou d'une autre calculer, alors une approche de l'apprentissage doit s'appuyer sur la notion de calcul. Le cadre de la théorie de la complexité est un cadre candidat à une modélisation possible.

Si apprendre c'est d'une manière ou d'une autre converger, alors une approche de l'apprentissage doit s'appuyer sur la notion de convergence. Le cadre de la théorie des probabilités est un cadre candidat à une modélisation possible.

Le modèle de VALIANT [VALIA 84] ou théorie de l'apprentissage *pac* (probably approximately correct) respecte cette double filiation.

Présentons et illustrons dans un tableau, le problème de l'apprentissage formel pour un système artificiel :

Problème P d'apprentissage pour un système artificiel (au sens de HAUSSLER)	Définitions	Exemple
<i>c'est quoi ?</i>	c'est : un quintuplet : $P = (X, Y, F, L, D)$ où	Problème de l'apprentissage RECT des rectangles du plan
1 l'espace des données du problème	X : est le domaine du problème	X est le plan euclidien \mathbb{R}^2
2 l'espace des résultats du problème	Y : est le codomaine du problème	Y est { oui, non } : le point (c, l) de \mathbb{R}^2 est-il dans le rectangle r ?
3 les concepts du problème	F : est une classe de fonctions f de X dans Y	F classe de fonctions notées r : modélisant les rectangles du plan définis par les coordonnées des extrémités de l'une de leurs diagonales $\mathbb{R}^2 \xrightarrow{r} \{\text{oui, non}\}$ $(c, l) \mapsto r(c, l)$ si $c_A \leq c \leq c_B$ et $l_A \leq l \leq l_B$ alors $r(c, l) = \text{oui}$ sinon $r(c, l) = \text{non}$
4 la variabilité du problème <i>remarque</i> : on peut se limiter à D_X densité de probabilité sur $(X, \Omega(X))$ car $(X \times Y, \Omega(X \times Y))$ est alors muni de D $P_D\{A\} = P_D \int_X \chi(x) \mathbb{1}_{A(x)} dx$ χ est la fonction caractéristique du concept à apprendre	D : est une classe de densités de probabilités sur l'espace : $(X \times Y, \Omega(X \times Y))$	D : densités de probabilité, uniformes sur le plan.
5 la tolérance admise du problème on définit une erreur : $\text{err}(f) = E_D(L)$ (espérance) on définit un optimum : $\text{opt}(D, F) = \min_{f \in F} E_D(L)$	L : est une fonction de perte relative à f, bornée par M $X \times Y \xrightarrow{L} [0; M]$ $(x, y) \mapsto L(f(x), y)$ <ul style="list-style-type: none">• perçue comme une variable aléatoire bornée sur $X \times Y$ qui admet ainsi des moments à tous les ordres• chiffre l'erreur de modélisation entre y et f(x).	L fonction de perte standard si $r(c, l) = y$ alors $L((c, l), y) = 1$ sinon $L((c, l), y) = 0$ c'est à dire si la question r(c, l) : le point (c, l) est-il dans le rectangle r donne la réponse y alors la fonction L donne 1 sinon elle donne 0.
<i>comment est-il résolu ?</i> apprendre c'est converger pour une suite S appelée échantillon, d'hypothèses de l'apprenti formulées à partir de présentations d'exemples. Lorsque la taille de l'échantillon croît, la suite des hypothèses de l'apprenti doit probablement converger vers une hypothèse approximativement correcte.	il est résolu par : une fonction de prédiction A appelée apprenti sur l'échantillon S $X \xrightarrow{A[S]} Y \quad \text{ou} \quad X \times S \xrightarrow{A} Y$ $u \mapsto v \quad \text{ou} \quad (u, s) \mapsto v$ v est tel que $(u, v) \in s$ un problème d'apprentissage est donc résolu par un apprenti modélisé par une fonction	L'apprenti est modélisé par une fonction A de prédiction : $\mathbb{R}^2 \xrightarrow{A[S]} \{\text{oui, non}\}$ $(c, l) \mapsto A[S](c, l)$ une solution au problème d'apprentissage du rectangle r avec les données D, α et δ est un rectangle r' tel que $P_D\{r \Delta r'\} \leq 1 - \alpha$

<p>La fonction A modélise l'apprenti</p> <p>L'apprenti connaît :</p> <ul style="list-style-type: none"> • le taux de fiabilité δ • le taux d'erreur à respecter $1-\alpha$ ou <p>La précision de prédiction α</p>	<p>A résout le problème P</p> <p><i>si et seulement si</i></p> <p>pour toute mesure D, la suite Z_n des variables aléatoires définies par :</p> $Z_n(S) = \text{err}_D(A[S n])$ <p>converge uniformément en probabilité vers $\text{opt}(D, F)$</p>	<p>L'apprenti tire aléatoirement des points du plan selon une mesure D de probabilité et détermine grâce à un "oracle" si les points appartiennent ou non au rectangle à apprendre; c'est à dire, il génère un échantillon S. On prouve que la constitution d'un échantillon de taille $\lceil \frac{4}{1-\alpha} \ln \frac{4}{\delta} \rceil$ assure l'apprentissage d'un rectangle r</p>
---	--	---

autrement dit :

pour tout paramètre d'écart de HAUSSLER k strictement positif, pour toute précision de prédiction α comprise entre 0 et 1, pour tout taux de fiabilité δ compris entre 0 et 1, il existe une fonction m de \mathbb{R}^3 dans \mathbb{R} , telle que pour toute mesure D de probabilité, tout échantillon S de $m(k, \alpha, \delta)$ couples tirés aléatoirement selon D, on ait :

$$P_{D^{m(k, \alpha, \delta)}} \{ d_k(\text{optimum}, \text{erreur}(A[S])) \geq \text{précision} \} \leq \text{fiabilité}$$

<p>d_k est la distance de HAUSSLER définie par :</p> $d_k(f, g) = \frac{ E_D(L_f) - E_D(L_g) }{k + E_D(L_f) + E_D(L_g)}$	<p>la complexité en échantillon d'un apprenti A est la fonction m_A définie par</p> $m(k, \alpha, \delta) = \min_n \max_D \{ d_k(\text{opt}(D, F), \text{err}_D(A[S])) \geq \alpha \} \leq \delta$ <p>la complexité en échantillon d'un problème P est le minimum de la complexité en échantillon des apprentis A résolvant le problème</p>
---	--

Quelles sont ses caractéristiques ?	Le problème ...	
<p>1 la nature du terrain d'apprentissage</p> <p>une des qualités de ce protocole sera sa reproductibilité</p>	<p>est soumis à un protocole</p> <p>un protocole est articulé en deux composantes :</p> <ul style="list-style-type: none"> • l'état initial (une configuration stable), c'est à dire les compétences initiales de l'apprenti • une interaction dans une durée de l'apprenti avec son environnement 	<p>par exemple pour les interactions :</p> <ul style="list-style-type: none"> • un dressage • un processus d'équilibration relaxation • une induction • un processus de transmission
<p>2 la nature du succès</p> <p>Le critère d'apprentissage ne sert pas à évaluer l'apprentissage, mais à permettre la définition de problèmes d'apprentissage justiciables de traitements et de démonstrations formels.</p>	<p>dispose d'un critère de succès :</p> <p>un critère d'apprentissage définit les suites d'hypothèses qui peuvent être produites par un apprenti qui a effectivement appris l'objet à apprendre, il peut ne pas être vérifiable expérimentalement mais seulement être utilisé pour une preuve formelle.</p>	<p>par exemple :</p> <ul style="list-style-type: none"> • le critère d'identification finie. • fréquence d'apparition de certains événements • détection de certaines régularités
<p>3 la nature de ce que l'on cherche à apprendre</p>	<p>possède un objet d'apprentissage :</p>	<p>par exemple :</p> <ul style="list-style-type: none"> • des catégories perceptives techniques : triangles, cercles,... • des fonctions • des grammaires...

Partie 2 - Chapitre 5 - Modélisation des systèmes complexes

où l'on introduit l'approche systémique.

Le modèle est une représentation artificielle que l'on construit dans sa tête et que l'on "dessine" sur un support physique, construit par l'homme, il agence des symboles.

Modéliser, c'est à la fois identifier et formuler un problème et chercher à le résoudre en raisonnant sur des simulations. *"Modélisation et simulation, réflexion et raisonnement sont les deux faces inséparables de toute délibération"* [LEMOI 90]

La caractérisation d'un phénomène dans une perspective de modélisation se hiérarchise en 9 niveaux de description :

- **il est** : le phénomène est identifiable
- **il fait** : le phénomène est actif
- **il est régulé** : le phénomène possède une certaine stabilité
- **il s'informe** : le phénomène se renseigne sur son comportement
- **il décide** : le phénomène prend le statut de système, il est composé de sous systèmes en interaction : le système de décision et le système opérant.
- **il mémorise** : le phénomène est un complexe de systèmes en interaction : systèmes de décision, de mémorisation et opérant.
- **il coordonne** : le phénomène est un complexe de systèmes en interaction : systèmes de décision, de mémorisation et opérant ; on identifie une coordination.
- **il imagine** : le phénomène est un complexe de systèmes en interaction, systèmes de décision, de mémorisation et opérant ; on identifie une coordination et une imagination.
- **il se finalise** : le phénomène est un complexe de systèmes en interaction, systèmes de décision, de mémorisation et opérant ; on identifie une coordination, une imagination et une finalisation. Buts et objectifs déterminent l'évolution du phénomène dans le temps.

Le modèle de la coopération

Le modèle de la coopération se caractérise par une connaissance répartie entre différents acteurs spécialistes ayant chacun des méthodes de résolutions spécifiques, des points de vue différents, qui coopèrent entre eux pour résoudre un problème soumis à des contraintes.

Chacun des experts :

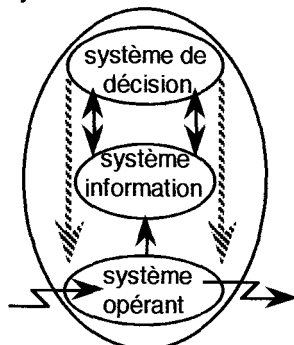
- résout un sous problème qui peut soit s'inscrire dans un effort commun entrepris par un groupe, soit entrer en concurrence avec d'autres.
- dispose de connaissances limitées sur son environnement, sur les actions et les intentions des autres.
- produit des échanges par des signaux (aspect communication)
- exploite des ressources à partager (aspect cohabitation)
- définit des tâches à réaliser (aspect coopération)
- reconnaît des formes par filtrage ou unification (aspect reconnaissance)
- se situe dans un environnement par évaluation de distances (aspect placement)
- planifie son action par création de plans stratégiques (aspect finalisation)
- voit émerger un comportement global cohérent de la société d'experts à partir de décisions prises localement par chaque spécialiste.

Le modèle d'organisation

Un système d'organisation est un système qui admet en entrée une forme et qui fournit en sortie une forme organisée, c'est à dire une structure.

Le modèle de l'organisation est un complexe de trois sous systèmes :

un système de décision, un système d'information et un système d'opération.



Le projet du système est d'augmenter son organisation, sa compétence.

A chacun des niveaux de ces trois systèmes, on distingue :

Système de décision	COMPRENDRE	CONCEVOIR	FINALISER
Système d'information	COMPUTER	COMMUNIQUER	MEMORISER
Système d'opération	PRODUIRE	RELIER	MAINTENIR
	dans un même instant	et dans une évolution	pour une autonomie

SOMMAIRE

Introduction générale	1
Partie 1 - Chapitre 0Introduction	1
Partie 1 - Chapitre 1. Préliminaires relatifs aux machines	1
1. Optimalité :.....	1
2. Minimalité.....	1
Partie 1 - Chapitre 2. Complexité restreinte	1
1. Définition :.....	1
2. Conditions de cohérence.....	2
3. Propriétés restreintes élémentaires :.....	4
Partie 1 - Chapitre 3. Langages booléens de représentation.	1
1. L'idée de base:.....	1
2. Axiomes booléens.....	2
3. Propriétés booléennes élémentaires :.....	2
4. Exemples :.....	6
Exemple 1. : Les mots et la structure de pile.....	6
Exemple 2. : Le filtrage et l'unification dans les arbres.....	8
Exemple 3. : Les ensembles.....	8
Partie 1 - Chapitre 4. Prétraitement numérique et compression. Représentations exactes et approchées.	1
1. Généralités.....	1
Représentation par approximation dans un langage L.....	2
1. Définitions:.....	2
2. Continuité, minimalité et effectivité de la représentation.....	2
2. Étude de cas: dessins au trait sur une rétine.....	3
1. Deux exemples de langages de représentation.....	3
1. Le cas non restreint:.....	3
2. Approximation par segments.....	4
2. Niveaux d'organisation d'un dessin au trait.....	7

Partie 1 - Chapitre 5. Complexité et traitement numérique symbolique. Une première approche.....	1
1. Généralités.....	1
1. Traitement numérique, traitement symbolique et booléen.....	2
2. Exemple.....	3
3. Qualités d'un langage de représentation booléenne relativement à un espace sémantique (E,d).....	6
- La précision.....	6
-La (capacité d') analogie.....	6
-La (capacité de) différenciation.....	6
-L'épaisseur.....	6
2. Le dessin au trait.....	7
1. L'étape numérique : approximation par segments.....	7
2. Un langage booléen sans capacité d'analogie.....	8
3. Un langage booléen sans capacité de différenciation.....	8
4. Un compromis entre capacités d'analogie et de différenciation : représentation pyramidale.....	10
 Partie 1 - Chapitre 6. Construction de bases de connaissances	1
Aspect polynomial de la construction de la base :.....	2
 Partie 1 - Chapitre 7.....Conclusion	1
 Partie 1 Annexes.....	1
Annexe I. Dessin au trait.....	1
Annexe II. L'approximation d'un dessin au trait par un nombre minimum de segments est NP-dure.....	4
Théorème :.....	4
 Partie 1 Bibliographie.....	1
 Partie 2 - Chapitre 0.....Introduction.	1

Partie 2 - Chapitre 1.....La machine PASTIS	1
01. La notion de machine PASTIS.....	2
02. Les étapes historiques de l'évolution du comportement PASTIS.....	2
03. Les états du comportement quotidien de PASTIS.....	3
04. Les choix de base pour PASTIS.....	4
4.1 Les choix des objets.....	4
4.2 Les choix architecturaux.....	5
05. La machine PASTIS illustrée.....	6
BIBLIOGRAPHIE.....	7
Partie 2 - Chapitre 2.Mécanismes humains de la vision	1
01. Les modèles neurobiologiques de la vision.....	2
02. Élaboration des images visuelles par le cerveau.....	3
Notion de spécialisation fonctionnelle d'aires corticales.....	3
Notion de synchronisation.....	5
BIBLIOGRAPHIE.....	7
Partie 2 - Chapitre 3..... Vision et dessins	1
01. Les systèmes de vision : état du domaine.....	2
1.1 Introduction.....	2
1.2 Outils et connaissances.....	2
1.3 Styles de conception et fonctions mises en œuvre.....	3
1.4 Conclusion.....	4
02. L'image objet d'étude : ses représentations.....	5
03. Les méthodes.....	6
3.1 . Méthodes de la morphologie.....	6
+ Exemples :.....	8
+ Hiérarchie des transformations de la morphologie mathématique :.....	11
+ Conclusion :.....	11
3.2 . Méthodes statistiques :.....	12
3.3 . Méthodes structurelles :.....	17
3.3.1. On code la forme comme une chaîne.....	17
3.3.2. On code la forme comme un terme.....	22
3.3.3. On code la forme comme un arbre.....	27

3.3.4. On code la forme comme un graphe.....	30
3.4 . Méthodes analytiques :.....	33
3.4.1. l'approximation par une ligne polygonale ou une courbe.....	33
3.4.2. l'approximation par B spline.....	36
3.5 . Méthodes connexionnistes :.....	38
3.5.1 Description d'un système connexionniste.....	38
3.5.1.1. Les unités constituant un système connexionniste	38
3.5.1.1.1 L'automate.....	38
3.5.1.1.2 L'unité booléenne de VON NEUMANN.	39
3.5.1.1.3 Le neurone formel de MAC CULLOCH et PITTS.	39
3.5.1.1.4 Les fonctions de transition des automates.....	40
3.5.1.2 Description d'un système connexionniste.....	41
3.5.1.3 Choix possibles pour des systèmes connexionnistes	44
3.5.2. Ingénierie des systèmes connexionnistes	45
3.5.3 Panorama des systèmes connexionnistes :.....	46
3.5.3.1 Les racines	46
3.5.3.2 Les descendance de la modélisation de VON NEUMANN.....	47
3.5.3.2.1 Les réseaux cellulaires.....	47
3.5.3.2.2 Les réseaux booléens à une entrée.....	48
3.5.3.2.3 Les réseaux booléens à construction aléatoire.....	48
3.5.3.3 Les descendance du modèle de MAC CULLOCH et PITTS	49
3.5.3.3.1 Le perceptron de ROSENBLATT [1958].....	50
3.5.3.3.2 L'adaptateur linéaire ADALINE de WIDROW [1960].....	56
3.5.3.3.3 Le filtre linéaire.....	58
3.5.3.3.4 Le réseau de HOPFIELD	62
3.5.3.3.5 La machine de BOLTZMANN	68
3.5.3.4 Les perspectives actuelles.....	70
3.5.3.4.1 Projection orthogonalisation (GRAMM SCHMIDT).....	71
3.5.3.4.2 Calcul d'une pseudo inverse (PENROSE)	72
3.5.3.4.3 Rétro propagation du gradient (Le CUN).....	75
3.5.3.4.4 Réseaux récurrents déterministes bidirectionnels (GUIVARCH).....	78
3.5.3.4.5 Réseaux organisés en colonnes et en aires (BURNOD)	82
3.5.3.4.6 Réseaux auto adaptatifs (KOHONEN).....	85
3.5.3.4.7 Réseaux simpliciaux (COSNES)	88
3.5.4. Exemples de machines connexionnistes.....	89
3.5.4.1 DIAGNOSTIQUER, systèmes experts et machines connexionnistes	89
3.5.4.2 RECONNAITRE, lire un texte écrit.....	90
3.5.4.3 ASSOCIER, apparier des images.....	90
3.5.4.4 RESTAURER l'image la plus probable, à partir d'une forme bruitée.	91
3.5.4.5 PREVOIR un moment optimal.....	91
3.5.5. Conclusion sur les modèles connexionnistes	91
BIBLIOGRAPHIE.....	93

Partie 2 - Chapitre 4.....Apprentissage 1

01. De la notion naïve de l'apprentissage à l'approche formelle2

02. Apprentissage comportemental5

03. Apprentissage gestaltiste8

04. Apprentissage cognitif.....10

 4.1 Modèle de SIEGLER10

 4.2 Modèle de PASCUAL LEONE10

 4.3 Modèle de Case.....11

 4.4 Modèle de FISCHER11

 4.5 Modèle de HALFORD11

05. Apprentissage connexionniste12

 5.1 Formes d'apprentissage.....12

 + Imposition d'une capacité (capacité innée).....12

 + Apprentissage supervisé (capacité enseignée).....12

 + Apprentissage non supervisé (capacité acquise).....13

 5.2 Mécanismes d'apprentissage.....13

 + L'auto association13

 + L'hétéro association13

 + La détection de régularité.....13

 5.3 Tactiques d'apprentissage.....13

 + Apprentissage non supervisé compétitif.....13

 + Apprentissage supervisé fondé sur l'optimisation d'un indicateur14

 Indicateurs :14

 LOIS itératives :14

 LOIS stochastiques :15

 Apprentissage par imposition d'une capacité innée.....15

 LOI linéaire:.....15

06. Apprentissage formel16

07. Apprentissage formel inductif.....23

BIBLIOGRAPHIE.....25

Partie 2 - Chapitre 5.....Modélisation des systèmes complexes	1
01. La modélisation : rendre intelligible un système.....	2
comparaison modélisation analytique – modélisation systémique.....	3
02. Le modèle de la coopération.....	6
03. Le modèle de l'organisation.....	7
04. Le symbole.....	8
BIBLIOGRAPHIE.....	9

Partie 2 . Guide :	1
Résumé partie 2.....	2
Plan.....	4
Partie 2 - Chapitre 1 - La machine PASTIS	5
Partie 2 - Chapitre 2 - les mécanismes humains de la vision.....	6
Partie 2 - Chapitre 3 - Vision et dessins.....	7
Partie 2 - Chapitre 4 - Apprentissage.....	16
Partie 2 - Chapitre 5 - Modélisation des systèmes complexes.....	20

