

Intelligence Artificielle

Pierre Collet

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection
Equipe Fouille de Données et Bioinformatique Théorique

Pierre.Collet@lsiit.u-strasbg.fr

Historique

- ◆ 1944 : ENIAC (Electronic Numerical Integrator And Calculator) (grosse machine à calculer, programme en dur).
- ◆ 1945 : von Neumann décrit l'architecture idéale : données, mais aussi programme dans la mémoire de l'ordinateur ! (EDVAC : Electronic Discrete *Variable* Automatic Computer)
- ◆ 1949 : Premier « vrai » ordinateur (architecture de von Neumann) : l'EDSAC (Electronic Delay Storage Automatic Calculator).
- ◆ 1950 : A. Turing écrit les premiers papiers sur ce que devraient être des ordinateurs « intelligents » (Computer Machinery and Intelligence, introduisant son « test de Turing »)
- ◆ 1950 : Shannon publie une analyse détaillée du jeu d'échecs vu comme un problème de recherche de solution.
- ◆ 1953 : Premier ordinateur commercial (IBM650).
- ◆ 1955 : Newel et Simon développent le « logic theorist » : en représentant un problème comme un arbre, le programme cherche la branche qui aurait le plus de chances de contenir la bonne solution.

Naissance et soubresauts de l'IA...

- ◆ 1956 : McCarthy invite des collègues pour le « Dartmouth Summer Research Project on *Artificial Intelligence* ». (cf. <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>) avec M. Minsky, N. Rochester, C. Shannon.
- ◆ 1957 : General Problem Solver (GPS), par Newell, Shaw et Simon,
- ◆ 1958 : McCarthy invente le LISP
- ◆ 60→70 : 1ère phase de développement de l'IA, avec de grands espoirs (1967 : Marvin Minsky « Le problème de créer de l'intelligence artificielle sera résolu de manière substantielle en une génération »).
- ◆ Milieu des années 70, les problèmes sont plus difficiles à résoudre que prévu... beaucoup de déception, et les financements de la recherche en IA sont coupés.
- ◆ 80-90 : renaissance de l'IA, avec les systèmes experts, et les réseaux neuronaux. Là aussi, de très grands espoirs grandissent, mais ce coup-ci, Minsky et les autres acteurs de l'IA essaient de calmer l'enthousiasme des industriels, car ils voient bien maintenant que l'IA ne pourra pas tout résoudre.

L'IA aujourd'hui...

- ◆ 90→ : le souffle retombe à nouveau, lorsqu'on se rend compte que les systèmes experts ne sont pas *la* solution, et que les réseaux neuronaux sont difficiles à manipuler et ne renvoient pas d'information utilisable...
- ◆ Actuellement, l'IA a mauvaise presse, car trop d'industriels ont perdu d'argent en investissant dans des techniques promettant monts et merveilles, et qui ne fonctionnent pas si bien.
- ◆ D'autres techniques ont le vent en poupe, comme les méthodes d'optimisation stochastique, notamment inspirées de la nature, comme l'évolution artificielle, l'optimisation par colonie de fourmis, l'optimisation par essaim particulaire (dont on revient), ...

History of EA

- ◆ The idea of implementing artificial evolution into a computer is as old as computers:
 - First computer: Wilke's EDSAC (1949) on a von Neumann architecture.
 - IBM 650 in 1953 (valves technology).
 - 1953: first tests of artificial evolution in Australia, US and Europe.
 - 1957: Fraser evolves binary strings using crossover
 - 1958: Friedberg suggests self-programming computers through mutations.
 - 1959: Friedmann writes a paper on how evolution could be digitally simulated.
 - ...

Main trends

- ◆ 1973: Ingo Rechenberg & Hans-Paul Schwefel (Germany):
Evolution Strategies
- ◆ 1975: John Holland (and later David Goldberg in 1989):
Genetic Algorithms
- ◆ 1985: Michael Cramer (and later John Koza 1992):
Genetic Programming

(Evolutionary Programming, by Lawrence Fogel in 1966 and later David Fogel in 1992 is dying.)



Qu'est-ce que l'intelligence ?

◆ Intelligence ? Intelligence artificielle ?

- Une mouche est-elle intelligente ?
- Et une mouche qui extrait des racines cubiques ?
- Une calculette qui extrait des racines cubiques est-elle intelligente ?
- N'existe-t-il qu'une seule forme d'intelligence ?

« C'est pas logique que dans le sud de la France il y ait des théâtres romains alors qu'on y pratique les corridas, qui sont espagnoles ! »
extrait de E. Angeli, Mes Evangelismes : Pensées d'une blonde

◆ Regardons d'autres domaines...

- Les hommes ont fini par réussir à voler : 
- Et pourtant, tant qu'on a cherché à imiter les oiseaux : 

Qu'est-ce que l'intelligence « artificielle » ?

Nombreuses « définitions » :

Dans un article de 1950, Alan Turing se pose la question de savoir si les machines peuvent penser (il pense que oui), et si c'est le cas, comment le déterminer ?

La solution proposée est de mettre en œuvre une « discussion » entre un homme et une machine. Si l'homme ne peut déterminer que son interlocuteur est une machine (ou un autre homme), alors c'est que la machine est intelligente.

Depuis 1990, Hugh Loebner a mis en place le Loebner Prize qui donnera \$100 000 à la première personne présentant un ordinateur passant le test de Turing.

Depuis, un certain nombre de Chatterboxes ont été développés, mais il reste du chemin à parcourir...

Vainqueur du Loebner Prize 2006

Hello, my name is Joan, and I am a woman.

do you enjoy life as a woman?

Yes.

for example?

I can't think of any.

is it being a woman you enjoy then or simply being human?

Yes, but it's honest. I'm going now.

but we have only just started

Oops, I spelt something wrong a bit ago.

what?

Something about yourself.

I heard something about suicide on the news this morning, did you catch it as well?

Yes we had a twister in birmingham and I live in birmingham.

alabama?

Alabama.

not west midlansds

Okay then, I now consider you boring.

that's not really a nice thing to say is it?

What would be?

so tell me all about yourself

I will if you tell me who your parents are

maybe we have the same parents, what do you think?

I don't care bout the same name. I want a booter.

what's a booter?

I probably don't want to know.

why not?

Because I would have to be married before I could consider having children.

well Joan how about me?

You need to know that I'm not impressed by men who are that forward with me.

how do you know that I am a man?

Because I saw your hairy can.

what's a can?

Wanting.

isn't that getting a little too open for the audience here?

Exactly.

Vainqueur du Loebner Prize 2012

◆ Transcripts of judge 4 program m_embar_chip_vivant round 4

◆ J: What is 2+2

B: 4.

J: What is 4+2?

B: 6.

J: Well done

J: Ask me a sum

B: Thank you. I'm enjoying this chat a lot. What is your name?

B: I didn't understand. How are you doing today?

J: I am doing well, thank you. My name is Tom.

J: What is your name?

B: Thanks for telling me your name, Tom. Do you have any hobbies?

B: Chip.

J: I do Chip. I like talking. Do you like talking?

B: I enjoy our conversation too.

J: What do you like about our conversation?

B: I'm neutral about that. What do you think?

J:

J: What is the weather like?

J: It

B: It's very hot and dry. I like it that way, though. There's also this incessant whirring noise. Do you play any musical instruments?

J: did hail a bit though, didn't it?

B: Interesting point of view. No more than that?

J: Well, quite a lot

...

Autres définitions

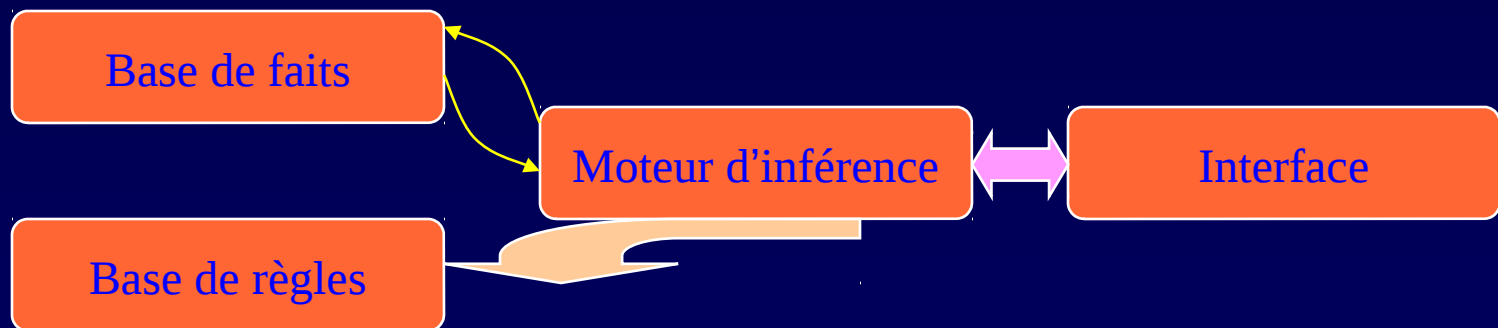
- ◆ Marvin Minsky: « [AI] is the science of making machines do things that would require intelligence if done by humans » (création de caulettes ?)
- ◆ John McCarthy, en Nov. 2004 (cf. <http://www-formal.stanford.edu/jmc/>):
 - Q: **What is AI ?**
 - A: It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable. »
 - Q: **And what is intelligence ?**
 - A: « Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines. »
 - Q: **Isn't there a solid definition of intelligence that doesn't depend on relating it to human intelligence?**
 - A: Not yet. The problem is that we cannot yet characterize in general what kinds of computational procedures we want to call intelligent. We understand some of the mechanisms of intelligence and not others.
 - Q: **Is intelligence a single thing so that one can ask a yes or no question ``Is this machine intelligent or not?''?**
 - A: No. Intelligence involves mechanisms, and AI research has discovered how to make computers carry out some of them and not others. If doing a task requires only mechanisms that are well understood today, computer programs can give very impressive performances on these tasks. Such programs should be considered ``somewhat intelligent".

Systeme Expert

- ◆ Logiciel censé reproduire le comportement d'un expert humain dans un domaine précis. En théorie, si les connaissances sont bien codifiées, toute connaissance humaine peut être convertie en SE.
- ◆ En pratique, c'est moins facile...
- ◆ Le problème principal réside dans la capacité de l'expert à transmettre son savoir. Joueur d'échecs ???

Principe de base...

- ◆ Aussi appelés Système à base de règles.
- ◆ Premiers systèmes experts dans les années 1980, et notamment l'un des premiers : MYCIN (système de diagnostic de maladie bactérienne du sang).
- ◆ L'idée est simple : récupérer l'expertise d'un humain (un médecin, par exemple) en la convertissant sous la forme d'une base de règles.
- ◆ Ensuite, on injecte des faits, et avec un moteur d'inférence, on déduit de plus en plus de faits à partir de la base de règles, jusqu'à répondre au problème.



Exemple

- ◆ Base de règles, de la forme si <prédicat> alors <action> :
 1. Si *Allaite_ses_petits* alors Mammifère
 2. Si *A_des_plumes* alors Oiseau
 3. Si Oiseau et non Vole alors Manchot
 4. Si Marin et non Mammifère alors Poisson
 5. Si Très_lourd et Mammifère alors Baleine
- ◆ Avec la base de faits suivante :
 - *Allaite_ses_petits*
 - *Très_lourd*
- ◆ Le moteur d'inférence regardera quelle règle est applicable. Ici, de toutes les règles, la règle 1 est applicable. Elle ajoutera un nouveau fait dans la base, qui deviendra alors :
 - *Allaite_ses_petits*
 - *Très_lourd*
 - *Mammifère*
- ◆ Maintenant, la règle 5 est applicable, et le moteur trouvera un autre fait :
 - *Baleine*
- ◆ Plus rien ne peut être déduit. On s'arrête là.

Raisonnements sur lesquels on s'appuie :

- ◆ **Modus Ponens** : si A et (A implique B) alors B
 - Ex : Si homme \Rightarrow mortel.
 - Si Socrate est un homme, alors Socrate est mortel.
- ◆ **Modus Tollens** : si non B et (A implique B) alors, non A.
 - Ex : Si homme \Rightarrow mortel
 - Si Zeus non mortel, alors Zeus n'est pas un homme.

Base de faits

- ◆ C'est la « mémoire » du système. Elle est de taille variable au cours de l'exécution, et vidée lorsque l'exécution se termine.
- ◆ Au début de la session, elle contient ce que l'on sait du cas examiné.
- ◆ Ensuite, elle est complétée par les déductions du moteur d'inférence, ou par des questions à l'utilisateur par l'intermédiaire d'une interface.
- ◆ Suivant l'ordre de la logique dans lequel on se place, les faits élémentaires peuvent prendre des valeurs plus ou moins complexes :
 - En logique d'ordre 0, on parle de propositions. Pas de variables ou même de constantes. Seulement des « mots » fixes, qui existent, ou n'existent pas. Ex : « 20km ». En logique d'ordre 0, on ne peut donc pas compter. (La notion de valeur numérique n'existe pas).
 - En logique d'ordre 0+, on peut utiliser des opérateurs de comparaison, mais aussi des constantes. Par exemple, on peut assigner 20km à la constante Distance.
 - En logique d'ordre 1, on peut manipuler des variables, mais du coup, aussi, des prédicats, c'est-à-dire des expressions logiques dont la valeur peut être vraie ou fausse selon la valeur des variables.
 - En logique d'ordre 2, on peut avoir des variables de prédicats, c'est-à-dire des variables contenant des règles. On peut alors écrire des méta-règles.

Base de règles

- ◆ Avec l'aide d'un « cogniticien », on rassemble la connaissance et le savoir-faire de l'expert sous la forme d'une base de règles. La base de règles est figée, et n'évolue pas au cours d'une session de travail.
- ◆ Une base de règles est un ensemble de règles et sa signification logique est la *conjonction* de la signification logique de chacune des règles.
- ◆ Ex : Si <conjonction de conditions> alors <conclusion>
- ◆ En logique d'ordre 1, on peut toujours écrire un fait sous forme d'une règle (et vice-versa) :
 - Fait : Le fer est un métal
 - Règle : Si X est en fer, alors X est en métal.

Moteurs d'inférence

- ◆ Le moteur d'inférence cherche à appliquer les règles, et déduire de nouveaux faits.
- ◆ Attention : il s'agit de programmation « déclarative » et non impérative. Potentiellement, l'ordre des règles n'est pas important (bien que si, en Prolog, où les règles sont examinées de manière séquentielle).
- ◆ Trois modes de fonctionnement :
 - Chaînage avant (raisonnement guidé par les données)
 - Chaînage arrière (raisonnement guidé par les buts)
 - Chaînage mixte (combinaison du chaînage avant et arrière).

Chaînage avant

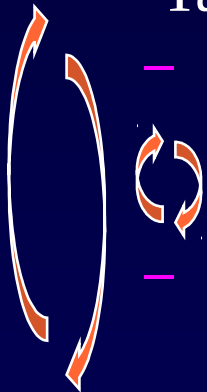
- ◆ Tant que le fait à déduire n'est pas dans la base de faits,
 - Parcourir la base de règles en examinant les prémisses, et déclencher toute règle dont les prémisses sont connues.
 - Pour éviter de boucler, on garde en mémoire les règles déclenchées pour ne pas les déclencher plusieurs fois.
 - Ajouter le fait déduit par la règle déclenchée dans la base de faits (s'il ne s'y trouve pas déjà).
 - S'il n'y a plus de règles déclenchables, et que le fait à déduire n'est pas trouvé, il est difficile de demander de l'aide à l'utilisateur, car on ne sait pas quel nouveau fait serait intéressant (sauf à faire du chaînage arrière).
- ◆ Le désavantage du chaînage avant, c'est que ça peut « exploser » facilement, surtout en logique non monotone (si une variable peut changer de valeur), même avec une petite base.
- ◆ Si, au lieu de chercher un maximum de choses, on cherche plutôt à répondre à une question précise, alors, le chaînage arrière est plus indiqué.

Chaînage arrière

- ◆ Raisonnement guidé par les buts.
- ◆ En partant du fait qu'on cherche à établir, on cherche toutes les règles concluant sur ce fait, et on repart de manière récursive de toutes les prémisses trouvées.
- ◆ Lors de chaque étape, on remplace un but par une conjonction ou une disjonction de sous-buts.
- ◆ Ca ne peut s'appliquer que lorsqu'on connaît le but, ou que lorsque les buts sont finis.
- ◆ Si l'on tombe sur des faits non connus, on peut demander à l'utilisateur s'il les connaît.

Chaînage mixte

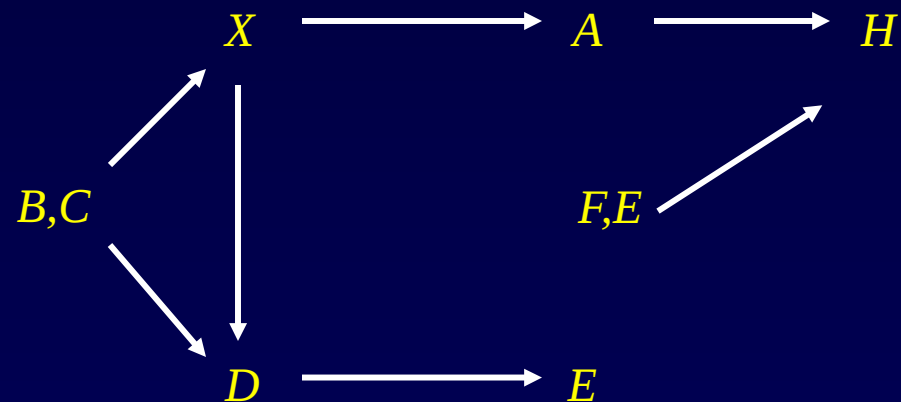
- ◆ Mélange de chaînage avant et arrière :
 - Entrée : Base de faits, et fait à déduire.
 - Tant que FAD non déduit, mais peut encore l'être :
 - Tant qu'on n'est pas bloqués :
 - Faire un chaînage arrière jusqu'à blocage.
 - Saturer la base de faits par un chaînage avant.
 - Demander des questions pertinentes pour ajouter les réponses de l'utilisateur à la base de faits.



Exemple

Soit $BaseFaits = \{B,C\}$, $FaitADecouvrir = H$ et $BaseRègles$ composée des règles :

1. Si B et D et E alors F
2. Si G et D alors A
3. Si C et F alors A
4. Si B alors X
5. Si D alors E
6. Si F et E alors H
7. Si C alors D
8. Si X et A alors H
9. Si X et C alors A
10. Si X et B alors D



Résoudre en chaînage avant, chaînage arrière en profondeur, et en largeur...

Exemple chaînage mixte

- ◆ Soit la base de règles suivante :
 - Si B et C alors A
 - Si D et E alors A
 - Si F et G alors A
 - Si I et J alors G
 - Si J alors $\neg E$
- ◆ On suppose que les faits B , D , F et I sont les seuls faits demandables.
- ◆ La mémoire de travail est initialisée avec l'information J est vrai.
- ◆ La question posée au système est : A est-il vrai ?
- ◆ Quelles sont les questions pertinentes à poser à l'utilisateur ?

C non demandable et n'apparaît nulle part ailleurs, donc ça ne sert à rien de demander B .

D demandable, mais E faux, donc ça ne sert à rien de demander D .

F demandable et pour connaître G , il faut connaître I qui est demandable.

Il faut donc demander I (qui permet de connaître G) et ensuite, demander F .

Régime révocable et irrévocable

- ◆ Régime révocable :

Possibilité de revenir sur la valeur des faits, pour permettre un chaînage arrière. On doit pouvoir maintenir un arbre des états dépendant de l'ordre des règles appliquées. La recherche peut s'orienter en largeur ou en profondeur, selon que l'on considère l'ensemble des règles pouvant s'appliquer à un moment donnée, ou les règles qu'on peut appliquer successivement depuis une situation donnée.

- ◆ Régime irrévocable :

les faits obtenus changent l'état de manière définitive : pas de backtracking possible. Chaque règle ne peut s'appliquer qu'une seule fois. Ce régime convient dans le cas où la plupart des chemins mènent à la solution (ce qui est rare).

Transitions non permutable et irréversibles

- ◆ Attention : il peut y avoir des « paires critiques » de règles, où R1 peut s'exécuter avant R2, mais pas l'inverse...
- ◆ Ex : R1 : $\sin(u) \cos(u) \rightarrow (\sin(2u))/2$
R2 : $\log(uv) \rightarrow \log(u)+\log(v)$
 - Pour traiter $\log(\sin(x)\cos(x))$, R1 et R2 peuvent s'appliquer.
 - Si on applique d'abord R2 : $\log(\sin(x))+\log(\cos(x))$, alors, on ne peut plus appliquer R1.
 - Si on est en régime révocable, on peut alors revenir en arrière et essayer R2 avant R1 : $\log((\sin(2x))/2)$ puis R2 : $\log(1/2)+\log(\sin(2x))$.
- ◆ Pour explorer les deux solutions, il faut être en régime révocable (récursion autorisée).
- ◆ Transition irréversible : si le système expert contrôle un robot cuisinier, 1 seule casserole, R1 : faire une sauce, R2, faire cuire des pâtes, une fois une casserole vidée, on ne peut plus revenir en arrière...

Méta-faits et méta-valeurs

- ◆ Pour qu'un SE puisse modéliser la réflexion humaine, il doit pouvoir « réfléchir » aux faits qu'il manipule, aux formules qu'il peut construire, ...
 - ◆ On parlera de méta-connaissances.
 - ◆ Typiquement, tous les faits ne peuvent pas faire l'objet d'une question à l'utilisateur : si on recherche une maladie, le SE ne peut pas demander à l'utilisateur « quelle maladie avez-vous ? »
 - ◆ Il faut donc donner au fait « diagnostic » un caractère « non demandable ».
- demandable(diagnostic) sera un « méta-fait » qui renverra « faux ».

Méta-règles et méta-système

- ◆ On peut chercher à savoir quelles prémisses évaluer en premier, auquel cas, on aura un premier « méta-niveau » qui déterminera quelle règle (ou méta-règle) choisir.
- ◆ Attention, si on veut manipuler des méta-règles du système expert par les méta-règles elles-même, on risque l'indécidabilité liée au théorème de Gödel.
- ◆ Ex :
 - **MR1 : S'il y a des éruptions ou des rougeurs alors envisager: R1, R2, R3, R4**
 - **MR2 : Si le patient est une femme adulte et si R1, R2, R3, R4 sont envisagées alors R1 est prioritaire**
 - **MR3 : Si le patient est un adolescent ou un enfant et si R1, R2, R3, R4 sont envisagées alors R4 est peu probable**
 - **MR4 : Si deux règles sont également probables alors prendre en priorité celle qui a le plus de conditions**

Exemple de SE avec méta-règles (tiré de H. Gallaire)

MR1 : S'il y a des éruptions ou des rougeurs alors envisager: R1, R2, R3, R4

MR2 : Si le patient est une femme adulte et si R1, R2, R3, R4 sont envisagées alors R1 est prioritaire

MR3 : Si le patient est un adolescent ou un enfant et si R1, R2, R3, R4 sont envisagées alors R4 est peu probable

MR4 : Si deux règles sont également probables alors prendre en priorité celle qui a le plus de conditions

R1 : Si la fièvre est faible, la peau sèche, s'il y a des ganglions, pas de pustules ni de rhume alors envisager la rubéole

R2 : Si les boutons sont isolés et démangent beaucoup avec une faible fièvre, et si la croûte apparaît vite sur les pustules ou les vésicules alors envisager fortement la varicelle

R3 : S'il y a rhume, mal aux yeux, taches roses dans la gorge, boutons en taches, fièvre forte alors envisager la rougeole

R4 : S'il y a amygdales rouges, désquamation, forte fièvre, taches rouge vif alors envisager la scarlatine

◆ Base de faits: patient adulte femme, rougeurs, fièvre faible

1. Détection des métarègles applicables : MR1 MR2 MR4 \Rightarrow MR2

2. Règles applicables par MR2 : R1, R2, R3, R4

3. La base de faits permet de sélectionner R1 et R2, avec une préférence pour R1 (car plu de proba pour R1 d'après le résultat de MR2 mémorisé au tour précédent).

4. On est bloqués. Question posée: « Y a-t-il des pustules ? »

5. Si la réponse est « oui », R1 est éliminée, et seule R2 reste \Rightarrow réponse du SE : envisager fortement la varicelle

SE à logique floue

- ◆ Comme on l'a vu dans l'exemple précédent, souvent, il est difficile de trancher. Avec de la logique floue, on ajoute à chaque fait un indice de fiabilité (valeur entre 0 et 1).
- ◆ En logique floue, on affichera toutes les conclusions, avec leur fiabilité.
- ◆ Lorsqu'un fait reçoit plusieurs fiabilités (suite au déclenchement de plusieurs règles venant renforcer ou affaiblir le fait), elles sont combinées par des formules associatives.
- ◆ MYCIN (un des premiers SE opérationnel dès 1975) utilisait la logique floue, avec des formules associatives complexes)
cf. http://www.cert.fr/dcsd/THESES/fabiani/manuscrit_fabiani/node334.html

Conclusion sur les SE

Les SE fonctionnent très bien, mais...

- ◆ Problème essentiel : récupérer la connaissance d'un expert !
 - Communication expert-informaticien-machine
 - Pb de vocabulaire, de représentation des connaissances
 - Potentiellement très lent à mettre au point, si le domaine est vaste.
- ◆ Ne peut-on pas créer des règles à partir d'exemples ?
- ◆ Réponse : systèmes de classeurs (Learning Classifier Systems), qui élaborent des règles par méthode évolutionnaire.
- ◆ Des règles sont créées aléatoirement, puis notées, suivant qu'elles donnent de bons résultats ou non sur un jeu d'apprentissage.
- ◆ De nouvelles règles sont créées par croisement/mutation, pendant de nombreuses générations, jusqu'à aboutir à un système de règles utilisables.

PROLOG (PROgrammer en LOGique)

- ◆ 1972 : création de PROLOG par Alain Colmerauer et Philippe Roussel à Luminy (Marseille).
- ◆ Prolog manipule trois types de « termes » :
 - Des variables (chaîne alpha-numérique commençant par une majuscule ou par un « _ »). Les variables représentent toujours le même objet (inconnu) et ne changent pas de valeur.
 - Des termes élémentaires (ou atomiques) représentant des objets simples qui peuvent être des :
 - Nombres (entiers ou flottants)
 - Identificateurs (ou atomes) = chaîne alpha-numérique commençant par une minuscule comme **toto**.
 - Chaînes de caractères (entre « ») comme « **bonjour** ».
 - Des termes composés, représentant des objets composés. Ex :
 - `adresse(1, « rue Anatole France », strasbourg)`Le « foncteur » de ce terme composé est `adresse` et il est d'arité 3.

Mise en œuvre en Prolog

- ◆ Un « atome logique » exprime une relation entre des termes, qui peut être vraie ou fausse.
 - `pere(pierre,paul)` est une relation d'arité 2 entre Pierre et paul, pouvant s'interpréter comme « Pierre est le père de Paul ».
 - `habite(X,adresse(1, « rue Anatole France », strasbourg))` peut être interprété comme « Une personne inconnue X habite à l'adresse 1, rue Anatole France. »

Clauses PROLOG

◆ Faits :

- **A.** est un fait « vrai ».
- **pere(pierre,paul).** indique que « pierre est le père de paul » est vrai.
- **egal(X,X).** indique que « X est égal à X » pour toute valeur que X peut prendre.

◆ Règles :

- **$A_0 :- A_1, \dots, A_n.$**
 A_0 est vraie si les relations A_1 et ... et A_n sont vraies. A_0 est appelé *tête de clause* et A_1, \dots, A_n est appelé *corps de clause*.
- **`meme_pere(X,Y) :- pere(P,X), pere(P,Y).`**
pour tout X et pour tout Y, `meme_pere(X,Y)` est vrai s'il existe un P tel que `pere(P,X)` et `pere(P,Y)` soient vrais.

Programme PROLOG

- ◆ Suite de clauses regroupées en « paquets ». L'ordre dans lequel les paquets sont définis n'est pas significatif.
- ◆ Chaque paquet définit un « prédicat » constitué d'un ensemble de clauses dont l'atome de tête est identique et de même arité. L'ordre d'apparition des clauses est indifférent :

`personne(X) :- homme(X).`

`personne(X) :- femme(X).`

Pour tout X, `personne(X)` est vrai si `homme(X)` est vrai, ou `femme(X)` est vrai.

Exécution d'un programme PROLOG

- ◆ Il s'agit de poser une question à l'interprète PROLOG. La réponse est « **yes** » ou « **no** ».
- ◆ Une question peut comporter des variables, auquel cas la réponse de PROLOG est l'ensemble des valeurs vérifiant le programme.
- ◆ **?- pere(toto,X), pere(X,Y).**
Quelles sont les valeurs de **X** et **Y** telles que **pere(toto,X)** et **pere(X,Y)** soient vrais ?
- ◆ PROLOG listera l'ensemble des enfants de toto, et petits-enfants de toto (si toto est grand-père).

Exercice généalogique...

- ◆ En généalogie, il y a des règles (le frère du père est l'oncle), et des faits (Nicolas est le mari de Cécilia).
- ◆ Coder un arbre généalogique en évitant les informations redondantes...

Base de faits

◆ Faits :

```
/* andre+augustine */
/*  |__ bernard+becassine */
/*  |    |__ clement+chantal */
/*  |    |    |__ dudulle */
/*  |    |    |__ damien */
/*  |    |    |__ daniela */
/*  |__ babar */
/*  |    |__ celestine */
/*  |__ brigitte+baptiste */
/*  |__ cedric+charlotte */
/*  |    |__ didier */
/*  |    |__ dagobert */
/*  |    |__ dominique */
/*  |__ caroline */
```

Codage des faits

/ les hommes */*

homme(andre).

homme(bernard).

homme(babar).

...

/ les femmes */*

femme(augustine).

femme(becassine).

femme(brigitte).

...

/ les relations de parenté */*

enfant(bernard,andre).

enfant(bernard,augustine).

enfant(babar,andre).

enfant(babar,augustine)

...

Etablissement des règles...

```
parent(Y,X) :- enfant(X,Y).  
pere(X,Y) :- parent(X,Y),homme(X).  
mere(X,Y) :- parent(X,Y),femme(X).  
fils(X,Y) :- enfant(X,Y),homme(X).  
fille(X,Y) :- enfant(X,Y),femme(X).
```

Donnez les clauses déterminant :

grand-père, grand-mère, petit-fils, petite-fille,
frère, sœur, oncle, tante, cousin, cousine,
ancêtre

Le reste de la famille...

```
grand_parent(X,Y) :- parent(X,Z),parent(Z,Y).
grand_pere(X,Y) :- grand_parent(X,Y),homme(X).
grand_mere(X,Y) :- grand_parent(X,Y),femme(X).
petit_enfant(X,Y) :- grand_parent(Y,X).
petit_fils(X,Y) :- petit_enfant(X,Y),homme(X).
petite_fille(X,Y) :- petit_enfant(X,Y),femme(X).
frere_ou_soeur(X,Y)
:-pere(P,X),pere(P,Y),mere(M,X),mere(M,Y),X\==Y.
frere(X,Y) :- frere_ou_soeur(X,Y),homme(X).
soeur(X,Y) :- frere_ou_soeur(X,Y),femme(X).
oncle_ou_tante(X,Y) :- parent(Z,Y),frere_ou_soeur(X,Z).
oncle(X,Y) :- oncle_ou_tante(X,Y),homme(X).
tante(X,Y) :- oncle_ou_tante(X,Y),femme(X).
cousin_ou_cousine(X,Y) :- oncle_ou_tante(Z,X),enfant(Y,Z).
cousin(X,Y) :- cousin_ou_cousine(X,Y),homme(X).
cousine(X,Y) :- cousin_ou_cousine(X,Y),femme(X).
ancetre(X,Y) :- parent(X,Y).
ancetre(X,Y) :- parent(X,Z),ancetre(Z,Y).
```



Arbres de décision

Notion d'entropie dans la théorie de l'info.

- ◆ l'information est l'inverse de l'entropie.
- ◆ En terme de bits, l'entropie d'une distribution équiprobable de 4 caractères est de 2, car on aura besoin de 2 bits pour coder n'importe lequel des caractères.
- ◆ Un événement de proba $1/8$ (2^{-3}) nécessitera 3 bits pour être représenté. Un événement de proba $1/2^p$ (2^{-p}) nécessitera $-\log_2 1/2^p$ bits. (car $\log_2 2^{-3} = -3$)
- ◆ Exemple intéressant: pour 26 lettres équiprobables, l'entropie d'une lettre est de $-\log_2(1/26) = 4,70\dots$
- ◆ Mais en Français, les lettres ne sont pas équiprobables ! (e, w, ... scrabble !)

Entropie du Français

- ◆ Probas : a=8,11%, b=0,81%, c=3,38%, d=4,28%, e=17,69%, f=1,13%, g=1,19%, h=0,74%, i=7,24%, j=0,18%, k=0,02%, l=5,99%, m=2,29%, n=7,68%, o=5,20%, p=2,92%, q=0,83%, r=6,43%, s=8,87%, t=7,44%, u=5,23%, v=1,28%, w=0,06%, x=0,53%, y=0,26%, z=0,12%.
- ◆ Entropie du Français : $H = -\sum_{i=1}^{26} c_i \log_2 c_i$ avec c_i , probabilité d'apparition du caractère c_i (théorie de l'information : Shannon, 1948) = 3,954 bits. (rappel : $\log_2 x = \log_n x / \log_n 2$)
- ◆ Sur des digrammes, entropie plus faible, car il y a des règles grapho/phonotactiques (q suivi d'un u, gk impossible, ...). Sur des n-grammes, plus n est grand, plus l'entropie est faible.

Utilisation pour un arbre de décision

- ◆ Un ancêtre : codage de Huffman (codage préfixe).
he had had had had he had had had he'd have had a better mark
- ◆ Convention pour arbre binaire : oui (1) à gauche, non (0) à droite.

Arbres de décision, en I.A.

- ◆ Un arbre de décision est une structure permettant de déduire un résultat à partir de décisions successives, mais aussi de créer une base de règles qu'on peut par la suite utiliser dans un moteur d'inférence.
- ◆ On part de la racine, et on choisit une branche suivant la réponse à la question portée par le nœud, jusqu'à tomber sur une feuille, qui est la solution au problème posé.
- ◆ En 1979, Ross Quinlan propose l'algorithme ID3, qui crée des arbres de décision à partir de données, puis en 1986, C4.5. En 1993, *C4.5 programs for machine learning*, Morgan Kaufman, où l'on trouve l'exemple suivant :
 - Soit des enregistrements météo et en parallèle, le fait d'aller jouer au golf ou non. Suivant les conditions (ensoleillement, température, humidité, vent), quelle décision prennent les joueurs de golf ?

Exemple de Quinlan (C4.5, 1993)

Données recueillies

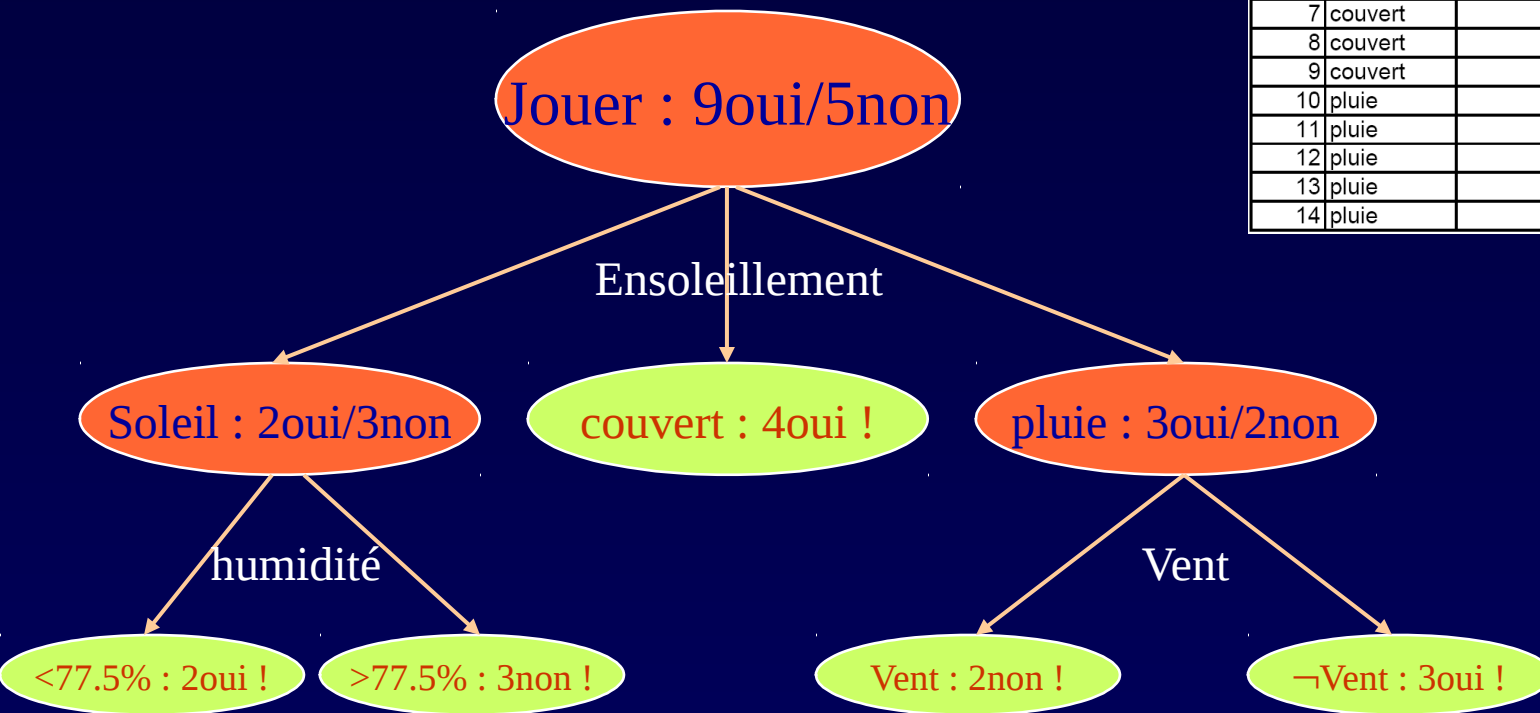
Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

Il y a 14 observations. Dans 9 cas, on ira jouer au golf, dans 5 cas, on restera à la maison.

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

Exemple d'arbre de décision

- ◆ On peut donc créer la racine et 3 branches :



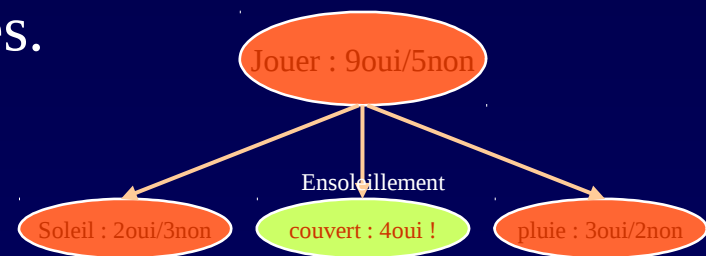
Número	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

Si ensoleillement=soleil et humidité > 77.5% on reste à la maison

Construction de l'arbre

- ◆ Pour créer les nœuds, on étudie toutes les variables, et on prend celle qui maximise un critère donné. Après « jouer » (qui est ce qu'on veut trouver), c'est Ensoleillement qui permet de segmenter au mieux (variable de « segmentation »).
- ◆ Comme il y a 3 modalités dans « ensoleillement » (soleil, couvert, pluie), il y a 3 branches.

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

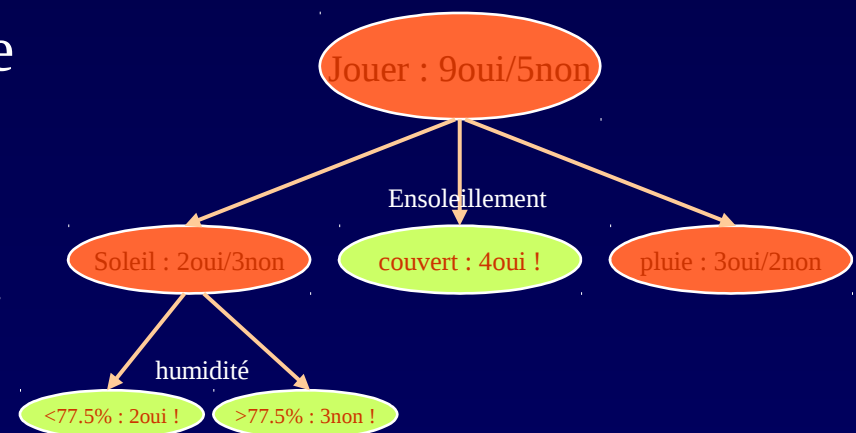


Construction de l'arbre

- ◆ La branche de gauche produite à partir de « soleil » n'est pas définitive, car dans certains cas, on joue (2 cas) et dans d'autres, non (3 cas).

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

- ◆ Ce qui permet de différencier les cas où l'on joue de ceux où l'on ne joue pas est l'humidité (et pas la température).
- ◆ Il faut alors choisir un seuil de « discrétisation » (ici 77.5%) permettant de créer 2 feuilles « pures » (tout oui / tout non).



Problèmes : choisir les variables de segm.

- ◆ Pourquoi « ensoleillement » (et pas « vent ») à la racine de l'arbre ?
- ◆ Pourquoi le seuil « 77.5 % » pour l'humidité ?
- ◆ 3 méthodes principales référencées :
 - CART
 - ID3 / C4.5 / C5.0
 - CHAID (Chi-squared Automated Interaction Detection, Kass[80])

Comment construire l'arbre...

- ◆ Recherche de la bonne variable de segmentation, et (si elle est continue) du seuil qui segmente le mieux.
- ◆ Ca permet de descendre d'un niveau, et on recommence de manière récursive.
- ◆ On s'arrête lorsqu'on rencontre une condition d'arrêt (profondeur max de l'arbre atteinte, nb max de feuilles, effectif de chaque nœud < seuil, ...).
- ◆ Ensuite, on peut éventuellement « élaguer » l'arbre.
- ◆ Si les arbres sont trop touffus, ils risquent de devenir trop spécifiques (risque de **surapprentissage**).

Critères de choix de la variable de segm.

- ◆ Il faut chercher à mettre en premier les variables les plus « importantes », c'est à dire celles dont dépendent le plus la décision à prendre.
- ◆ Suivant que cette variable est vraie ou fausse, on ira dans une direction ou un autre. Ensuite, on recommence en cherchant dans les autres variables la plus discriminante.
- ◆ Notations :
 - Soit p une proposition
 - $N(p)$ le cardinal des exemples associés à p .
 - $N(k/p)$ le cardinal des exemples de classe k associés à p .
 - $P(k/p)$ la proportion d'éléments de classe k par rapport à p .

Notations

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

- ◆ $N(\{\text{Soleil}, T^\circ > 65, H > 65, \text{pas de Vent}\}) = 3$
- ◆ $N(\text{Joue} / \{\text{Soleil}, T^\circ > 65, H > 65, \text{pas de Vent}\}) = 1$
- ◆ $P(\text{Joue} / \{\text{Soleil}, T^\circ > 65, H > 65, \text{pas de Vent}\}) = 1/3$

Choix de la variable de segmentation

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

◆ 9 jouent, 5 ne jouent pas. En regardant d'après :

◆ L'ensoleillement : (2,3) (4,0) (3,2)

Le vent : (3,3) (6,2)

La température . . : (Problème ! c'est continu !)

L'humidité : (Problème ! c'est continu !)

Comment comparer les différents choix ?

- ◆ On définit des fonctions permettant de mesurer le degré de mélange des exemples entre les différentes classes.
- ◆ Une bonne fonction prendra son minimum (resp. max) lorsque tous les exemples sont dans une même classe (le nœud est pur) et son maximum (resp. min) lorsqu'ils sont équirépartis.
- ◆ Si 14 joueurs potentiels et 2 classes (joue / joue pas), une segmentation idéale donnera (14, 0) et (0,14), et une segmentation sur une variable non discriminante (7,7).

Fonctions d'évaluation de la discrimination

- ◆ C4.5 utilise la notion d'entropie (évaluation du mélange des classes).
- ◆ $\text{Entropie}(p) = H(p) = - \sum_{k=1}^n P(k/p) \log_2 P(k/p)$
- ◆ Si une variable v ne discrimine rien, il y a autant de joueurs qui jouent si elle est vraie que si elle n'est pas vraie.
- ◆ $H(\text{classe equiprobable}) = - 7/14 \log_2 7/14 - 7/14 \log_2 7/14 = 1$
- ◆ $H(k=\text{Joue}) = - 9/14 \log_2 9/14 - 5/14 \log_2 5/14 = 0.939$
- ◆ $H(\text{disc pour } k \text{ pour Ensoleil.} = \text{Soleil}) = - 2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.971$
- ◆ $H(\text{disc pour } k \text{ pour Ensoleil.} = \text{couvert}) = - 4/4 \log_2 4/4 - 0/4 \log_2 0/4 = 0$
- ◆ $H(\text{disc pour } k \text{ pour Ensoleil.} = \text{pluie}) = - 3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.971$
- ◆ Maintenant, il faut composer les entropies pour évaluer la discrimination de l'ensoleillement pour k :

$$\text{disc}(v/k) = H(k) - \sum_{j=1} P(v_j) H(k/v_j)$$

avec $P(v_j)$ la proportion d'éléments de v_j qui satisfont k .
- ◆ Ici, ça donne :

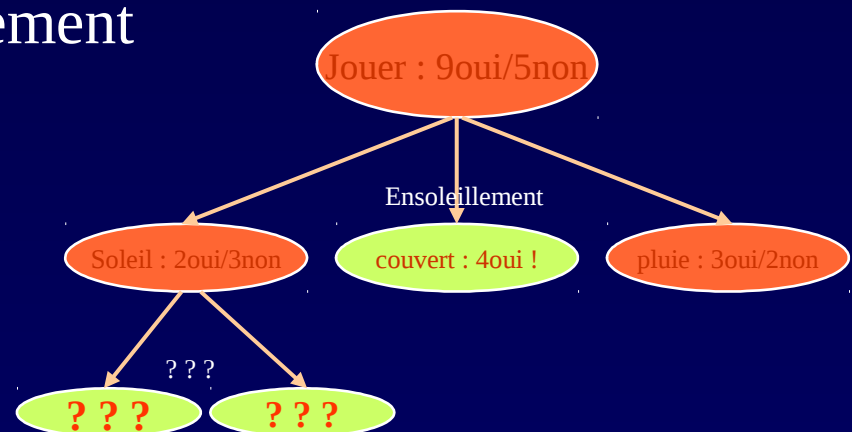
$$\text{disc}(\text{Joue}/\text{Ensoleillement}) = 0.939 - (5/14 \times 0.971 + 4/14 \times 0 + 5/14 \times 0.971) = 0.2454$$

Quelle variable choisir ?

- ◆ Quelle est la valeur discriminante du vent ?

$$0.939 - (6/14 \times 1 + 8/14 \times 0.811) = 0.0474$$

- ◆ Pouvoir discriminant = entropie max – entropie de la variable (normalisation de l'entropie)
- ◆ Le pouvoir discriminant de l'ensoleillement est 0,2454 et le pouvoir discriminant du vent est 0,0474
- ◆ Donc on choisit l'ensoleillement
- ◆ Recommencer avec le niveau du dessous...



Variables continues...

- ◆ Là, il faut comparer toutes les autres variables :
 - température, humidité, vent.
- ◆ Oui, mais l'humidité est une variable continue !
- ◆ Comment fait-on pour les variables continues ?
 - Tout comme l'ensoleillement pouvait prendre 3 valeurs, s'il y a du soleil « humidité » peut prendre les valeurs : 70, 85, 90, 95. On peut donc discrétiser en prenant les valeurs médianes, et pour toutes ces valeurs, déterminer la valeur discriminante de l'humidité...

70 - | - 85 - | - 90 - | - 95
 77.5 87.5 92.5

Un nœud est terminal s'il arrive sur des feuilles parfaitement discriminées.

Algorithme CART

- ◆ C'est une simplification de l'algorithme C4.5
- ◆ Au lieu de calculer l'entropie, la fonction utilisée pour mesurer le degré de mélange est la fonction de « Gini » (ou « indice d'impureté de Gini »).
- ◆
$$\text{Gini}(p) = \sum_{j=1}^n P(k/v_j)(1-P(k/v_j))$$
$$= 1 - \sum_{j=1}^n P(k/v_j)^2$$
- ◆ Tout comme l'entropie, cette fonction est maximale pour des modalités équiprobables

Application à l'exemple...

Numéro	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

- ◆ 9 jouent, 5 ne jouent pas. En regardant d'après :
- ◆ L'ensoleillement : (2,3) (4,0) (3,2) Disc (ensoleillement) = 0,1162428571
- Le vent : (3,3) (6,2) Disc (vent) = 0,03042857143
- La température . . : 70,5 78,5 Disc (temp) = 0,0316
- L'humidité : (4,1) 77,5 (3,2) Disc (humidité) = 0,031
- 87,5 (1,1) 92,5 (1,1)
- ◆ En mélangeant les facteurs discriminants avec la même formule que pour C4.5, $\text{disc}(p,t) = \text{Gini}(p) - \sum_{j=1}^n P_j \text{Gini}(p/v_j)$ fabriquez l'arbre CART.
- ◆ $\text{Gini}(p \text{ disc par humidité}) = 0,459 - 0,32 * 5/14 + 0,58 * 5/14 + 2/14 * 0,5 + 2/14 * 0,5$

Méthode CHAID

- ◆ Pour commencer (comme pour les autres méthodes), on met de côté le cas des variables continues (humidité, température).
- ◆ Restent deux descripteurs discrets. Pour chaque variable, partitionnement des observations, calcul d'un « indicateur de qualité » et choix de la variable « optimisant » cet indicateur.
- ◆ Une segmentation permet de définir un « tableau de contingence » croisant la variable à prédire et le descripteur candidat.

Tableau de contingence

...pour déterminer la dépendance entre 2 caractères :

Notation

$y \backslash x$	x_1	x_l	x_L	Σ
y_1		...		
y_k	...	n_{kl}	...	$n_{k.}$
y_K		...		
Σ		$n_{.l}$		n

Exemple avec des probas

Dé 1 \ Dé 2	1	2	Ni 1 ni 2	Total
6	1/36	1/36	4/36	6/36
Pas 6	5/36	5/36	20/36	30/36
Total	6/36	6/36	24/36	36/36

2 descripteurs seront indépendants si : $n_{kl} = \frac{n_{k.} \cdot n_{.l}}{n}$

En simplifiant :

Khi² d'écart à l'indépendance

- ◆ Formule du Khi² pour 2 caractères :

$$\chi^2 = \sum_{k=1}^K \sum_{l=1}^L \frac{\left(n_{kl} - \frac{n_{k.} \times n_{.l}}{n} \right)^2}{\frac{n_{k.} \times n_{.l}}{n}}$$

0 si indépendance

pour pondérer chaque case

Cette formule fait la somme de l'« indépendance » pour chaque case, mise au carré pour pouvoir par la suite diviser par le poids de chaque case.

Plus le Khi² d'écart à l'indépendance est proche de 0, plus les 2 caractères sont indépendants.

Normalisation par nb de degrés de liberté

- ◆ Tschuprow propose :

$$t = \frac{\chi^2}{n\sqrt{(K-1) \times (L-1)}}$$

(qui ramène la valeur du χ^2 à l'intervalle [0 ; 1])

Tableau de contingence pour variables discrètes

- ◆ Détermination de l'« importance » de la variable « ensoleillement » sur le fait d'aller jouer au golf :

Jouer/Ens	Couvert	Pluie	Soleil	Total
Non	0	2	3	5
Oui	4	3	2	9
Total	4	5	5	14

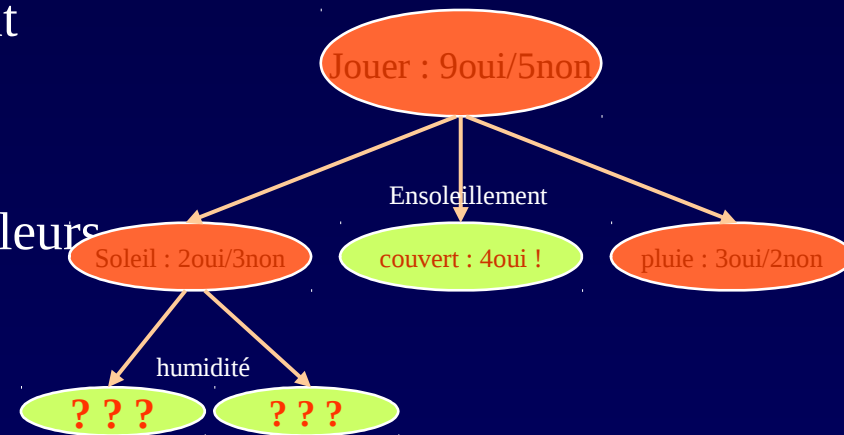
Número	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

- ◆ Si l'on commence par des variables discrètes, quelles sont les valeurs des t de Tschuprow pour :
 - ◆ Ensoleillement : 0.3559
 - ◆ Vent : 0.2582
- ◆ Quelle est la variable la plus « importante » ? (Celle dont dépend le plus la décision d'aller jouer au golf, et qu'on mettra donc en premier dans l'arbre ?)

Tableau de contingence pour variables réelles

Número	Ensoleillement	Température (°F)	Humidité (%)	Vent	Jouer
1	soleil	75	70	oui	oui
2	soleil	80	90	oui	non
3	soleil	85	85	non	non
4	soleil	72	95	non	non
5	soleil	69	70	non	oui
6	couvert	72	90	oui	oui
7	couvert	83	78	non	oui
8	couvert	64	65	oui	oui
9	couvert	81	75	non	oui
10	pluie	71	80	oui	non
11	pluie	65	70	oui	non
12	pluie	75	80	non	oui
13	pluie	68	80	non	oui
14	pluie	70	96	non	oui

- ◆ Maintenant, en étudiant le cas « soleil »
 - Comment choisir la bonne variable de segmentation ?
 - Calculer le t de Tschuprow pour les variables autres que *Ensoleillement*.
- ◆ Comment fait-on pour les variables continues ?
 - S'il y a du soleil « humidité » peut prendre les valeurs :
70 - | - 85 - | - 90 - | - 95
 - Déterminer le t pour toutes les valeurs médianes (77.5, 87.5, 92.5)



t pour les valeurs médianes de « humidité »

- ◆ On voit que pour la valeur 77.5, $t = 1$!
- ◆ Dans ce cas, aller jouer au golf dépend exactement de l'humidité (si < 77.5 , j'y vais, sinon, j'y vais pas).
- ◆ Mais qu'en est-il pour les autres variables (température ? vent ?)
- ◆ Et bien il faut calculer le t

Jouer	Hum < 77.5	Hum >= 77.5
Oui	2	0
Non	0	3
t de Tschuprow	1	

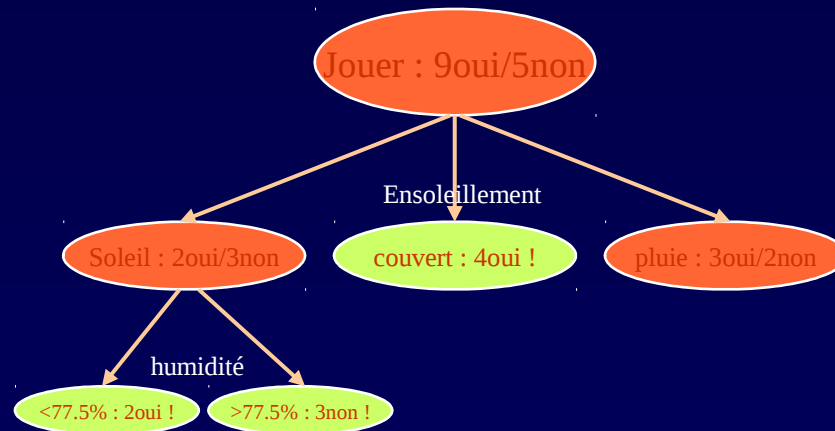
Jouer	Hum < 87.5	Hum >= 87.5
Oui	2	0
Non	1	2
t de Tschuprow	0.67	

Jouer	Hum < 92.5	Hum >= 92.5
Oui	2	0
Non	2	1
t de Tschuprow	0.41	

Segmentations candidates pour « soleil »

Descripteur	Point de coupure	T de Tschuprow
Humidité	77.5	1.00
Température	77.5	0.67
Vent	-	0.17

Il faut donc choisir « Humidité », car cette variable permet de proposer des feuilles « pures ».

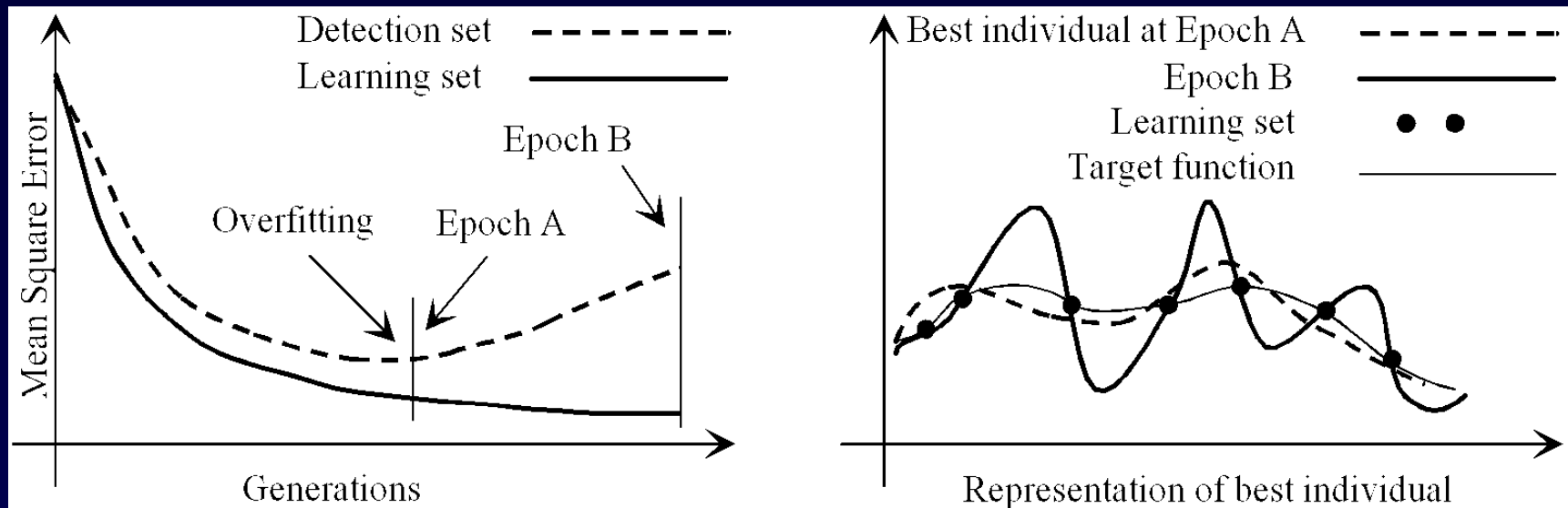


Maintenant, il faut refaire pareil pour « pluie ».

Notion de sur-apprentissage

- ◆ Observation : plus une solution est simple, plus son résultat est généralisable.
- ◆ Au début d'un apprentissage, amélioration des résultats avec des solutions simples (moins de choses à optimiser) jusqu'à ce qu'on arrive à un seuil où les améliorations génériques ne sont plus possibles.
- ◆ Alors, on rentre dans une phase où l'amélioration provient de l'ajout de verrues aux solutions génériques pour mieux approximer un point précis.

Détection du sur-apprentissage et évaluation



Nécessité d'avoir 3 jeux de données (Schoenauer[06]):

- 1 jeu d'apprentissage (sur lequel on fait évoluer les individus)
- 1 jeu de test (pour détecter le surapprentissage).
- 1 jeu d'évaluation (pour évaluer l'individu sur un jeu qui n'a pas participé à l'évolution de l'individu).

C. Gagné, M. Schoenauer & al., « GP, Validation Sets and Parsimony Pressure », EuroGP'06

Réseaux neuronaux

Neurones

- ◆ Le neurone biologique apparaît en 1891, avec une réelle validation en 1955 par observation d'un espace synaptique par microscopie électronique.
- ◆ Cerveau humain : environ 10^{11} neurones et de 10^{13} à 10^{14} connexions. Nombreux types différents, de taille très variable, basés sur une structure identique.
- ◆ 1 neurone peut :
 - Recevoir des signaux en provenance de l'environnement ou d'autres neurones,
 - Intégrer, traiter et manipuler des signaux,
 - Envoyer de l'information à d'autres neurones.
- ◆ Il est composé d'un noyau, de dendrites (10^3 à 10^4) pour l'information entrante et d'un axone (pouvant atteindre un mètre !) pour l'information sortante.
- ◆ Entre 2 neurones, l'information circule dans le sens axone vers dendrites à travers de synapses.

Influx nerveux

- ◆ Mécanismes de génération et de transmission de l'influx nerveux expliqués en 1940. En se dissolvant, certains sels forment des ions positifs et négatifs. Un déséquilibre crée une différence de potentiel.
- ◆ Principaux ions concernés : Sodium (Na^+), Potassium (K^+), Calcium (Ca^{2+}). La différence de potentiel est au repos de l'ordre de -70mV .
- ◆ Le corps cellulaire intègre l'influx nerveux en provenance des dendrites. Au-delà d'un certain seuil, il produit un « potentiel d'action » qui se diffuse le long de l'axone grâce à des canaux à ions.
- ◆ Le potentiel d'action dure quelques millisecondes et se propage à plusieurs m/s.
- ◆ Le potentiel d'action est identique, mais la fréquence est fonction du stimulus. C'est la fréquence qui code l'information.
- ◆ Arrivé au niveau des synapses, le potentiel d'action provoque la fusion de vésicules et la diffusion de neuromédiateurs, qui rencontreront des récepteurs de la membrane post-synaptique.
- ◆ La synapse peut accroître ou réduire le potentiel post-synaptique (effets excitateurs ou inhibiteurs). La durée de l'activation d'une synapse peut aller de quelques millisecondes à plusieurs minutes !

Apprentissage

- ◆ Mécanismes d'auto-organisation des poids synaptiques et des liaisons neuronales.
- ◆ L'habituation = disparition progressive d'une réaction en cas de stimuli répétés
- ◆ La sensibilisation = renforcement progressif d'une réaction en cas de stimuli répétés.
- ◆ Si les expériences d'habituation ou de sensibilisation sont répétées, de nouvelles connexions apparaissent ou disparaissent pour une mémorisation à long terme.

Traitement de l'influx

- ◆ Le corps cellulaire reçoit les influx en provenance des neurones auxquels il est lié par les dendrites. Certaines liaisons sont excitatrices, d'autres inhibitrices.
- ◆ Interprétation simple : le neurone fait la somme des influx reçus. Au-delà d'un certain niveau, émission d'un potentiel d'action (intégration synaptique).
- ◆ Intégration spatiale (dépend des neurones liés) et temporelle (possibilité de combiner des suites d'influx).

Modèle de McCulloch – Pitts

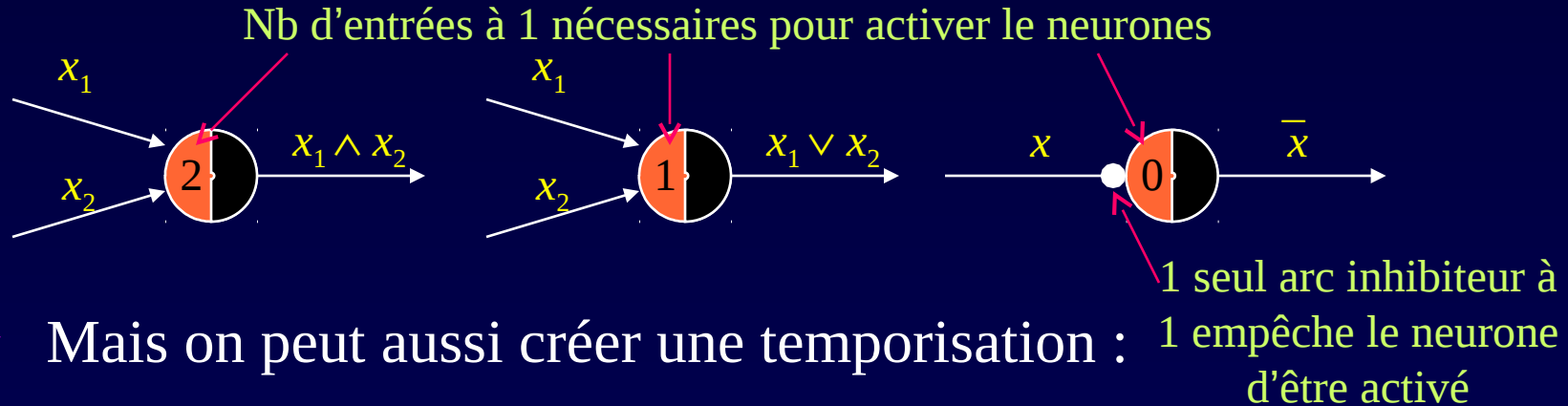
Les neurones de McCulloch-Pitts (1943) satisfont 5 propriétés :

- 1) L'activité des neurones est binaire (tout ou rien).
- 2) Pour exciter un neurone, il faut exciter un nombre fixe de synapses pendant une période d'« addition latente », indépendante de l'activité précédente.
- 3) Le seul retard provient des synapses.
- 4) L'activité de toute synapse inhibitrice bloque l'activation du neurone.
- 5) La structure du réseau est stable dans le temps.

(Les règles 4 et 5 ne sont pas conformes au modèle biologique.)

Implémentation d'opérations booléennes

- ◆ On peut implémenter ET, OU, NON :



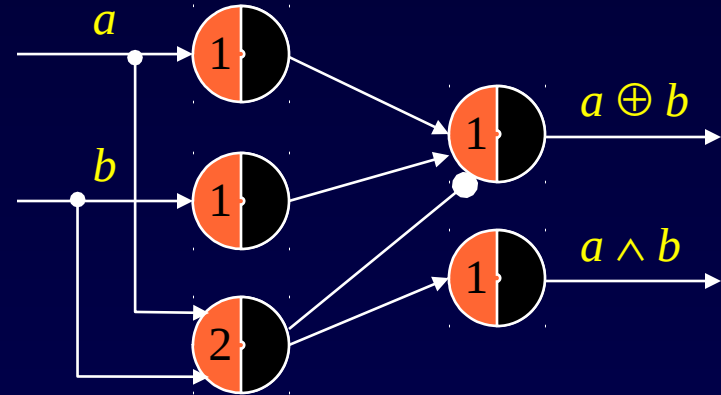
- ◆ Mais on peut aussi créer une temporisation :



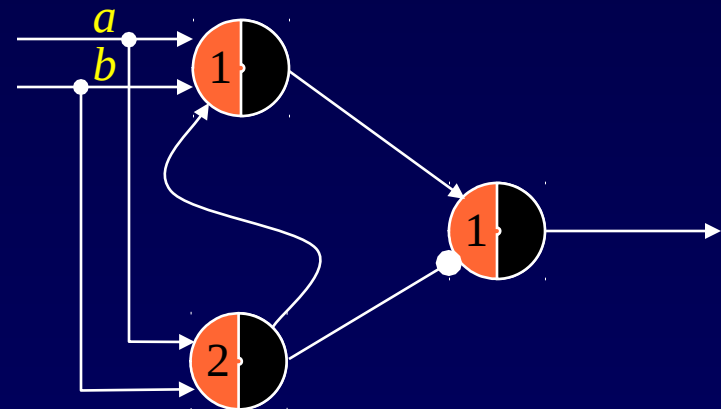
- ◆ Et donc, n'importe quelle opération booléenne, tout comme on fabrique un circuit logique à l'aide de portes logiques (universalité logique).
- ◆ En 1956, Kleene prouve l'universalité au sens de Turing des réseaux de McCulloch-Pitts.

Réseaux récurrents de McCulloch-Pitts

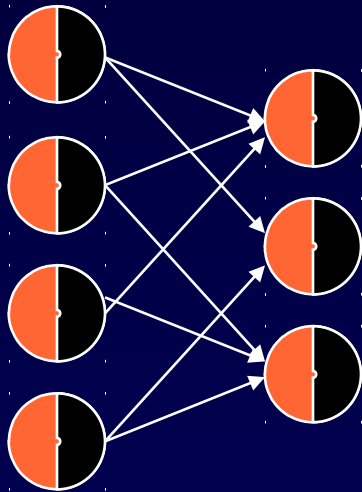
- ◆ Additionneur 1 bit booléen.
2 sorties : somme ($a \oplus b$) et retenue ($a \wedge b$).



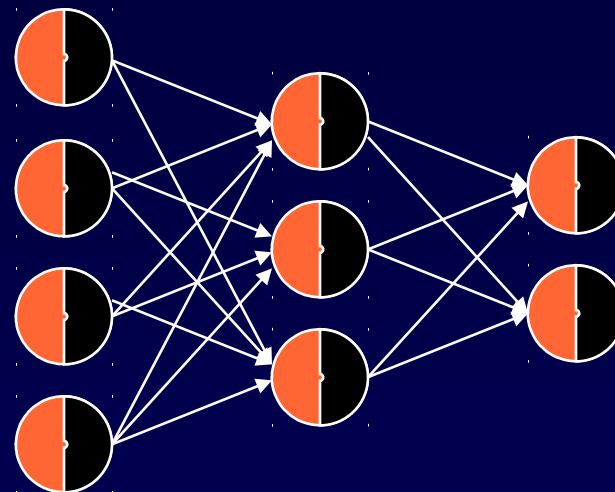
- ◆ Mais du fait de la temporisation, on peut faire des boucles !
(c'est un réseau récurrent)
Dans un premier temps, la somme sort du neurone de sortie, et dans un deuxième temps, c'est la retenue !



Réseaux « feed-forward » McCulloch-Pitts

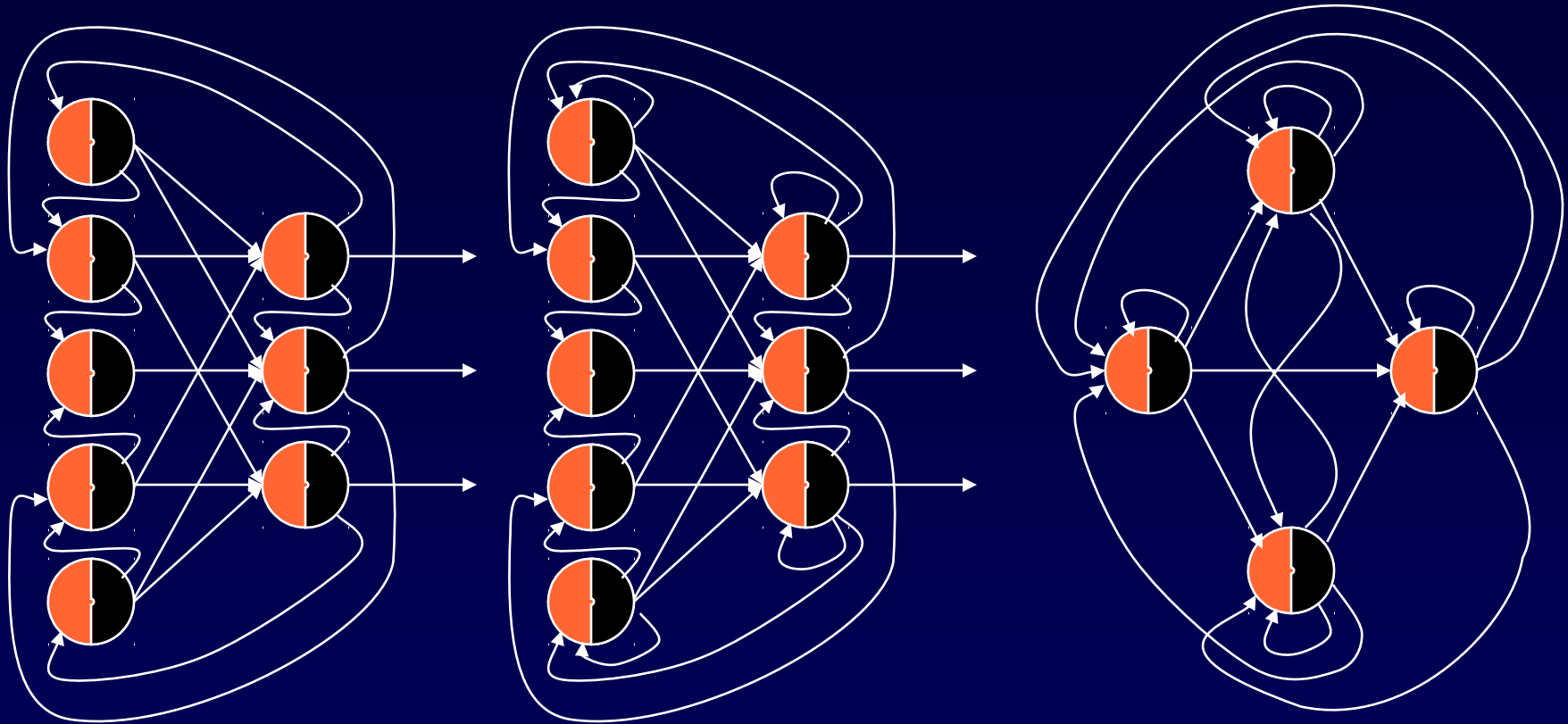


Réseau « feed forward »
monocouche 4-3 local



Réseau « feed forward »
multicouches 4-3-2 complet

Topologies de réseaux récurrents



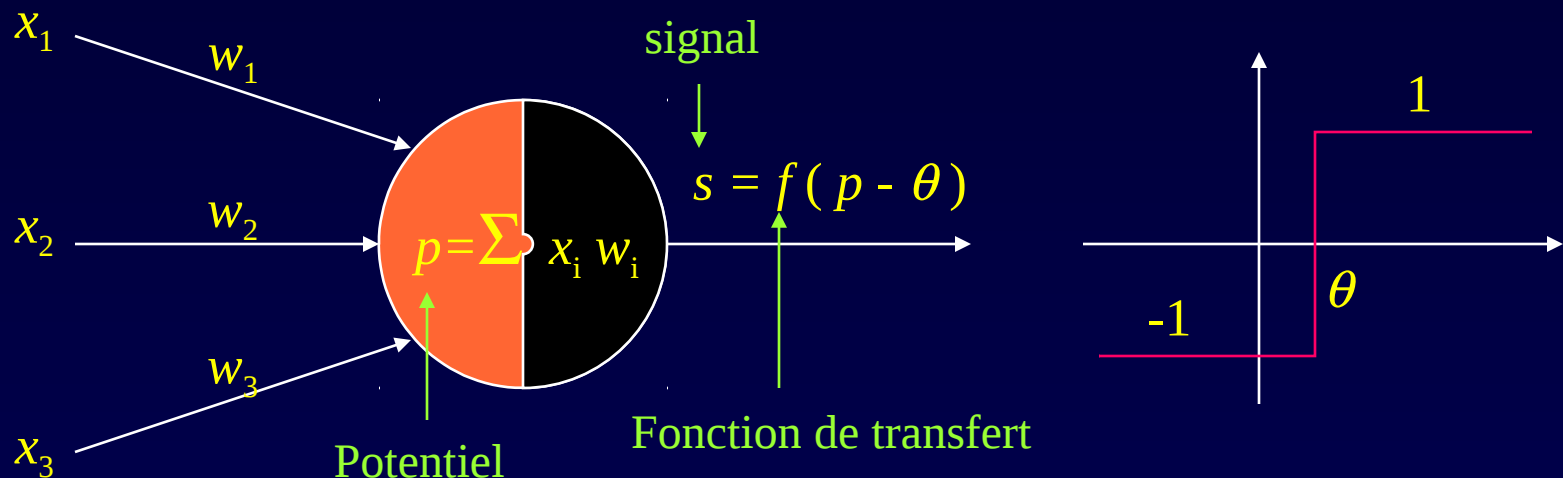
Réseaux récurrents sans auto-récurrance, avec auto-récurrance et complet

Réseaux susceptibles d'apprentissage

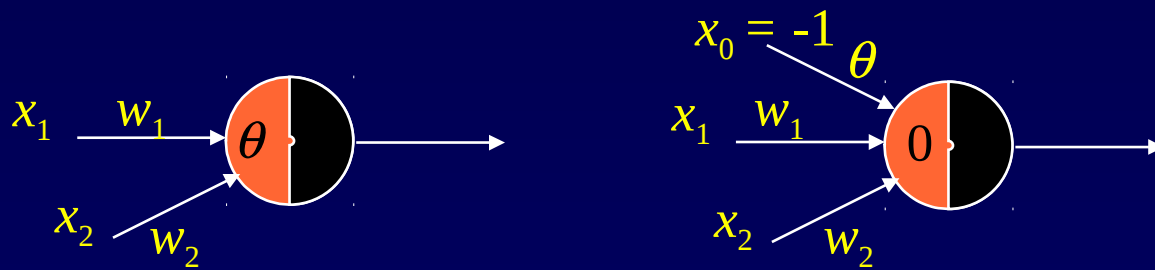
Perceptron

- ◆ En 1958, Rosenblatt met au point le « Perceptron », dans le but d'élaborer une classe de « modèles cérébraux. »
- ◆ Un Perceptron est un réseau monocouche qui possède :
 - Des unités sensibles (S-units) pour percevoir des stimulations extérieures
 - Des unités d'association (A-units) avec connexions entrantes et sortantes, créant un signal de sortie en fonction des entrées.
 - Des unités de réponses (R-units) avec connexions entrantes, créant un signal extérieur = +1 si la somme des signaux d'entrée est >0 , -1 si la somme est <0 , et sortie indéterminée si $= 0$
 - Une matrice d'interactions, définissant des coefficients de couplage entre différentes unités (poids synaptiques).

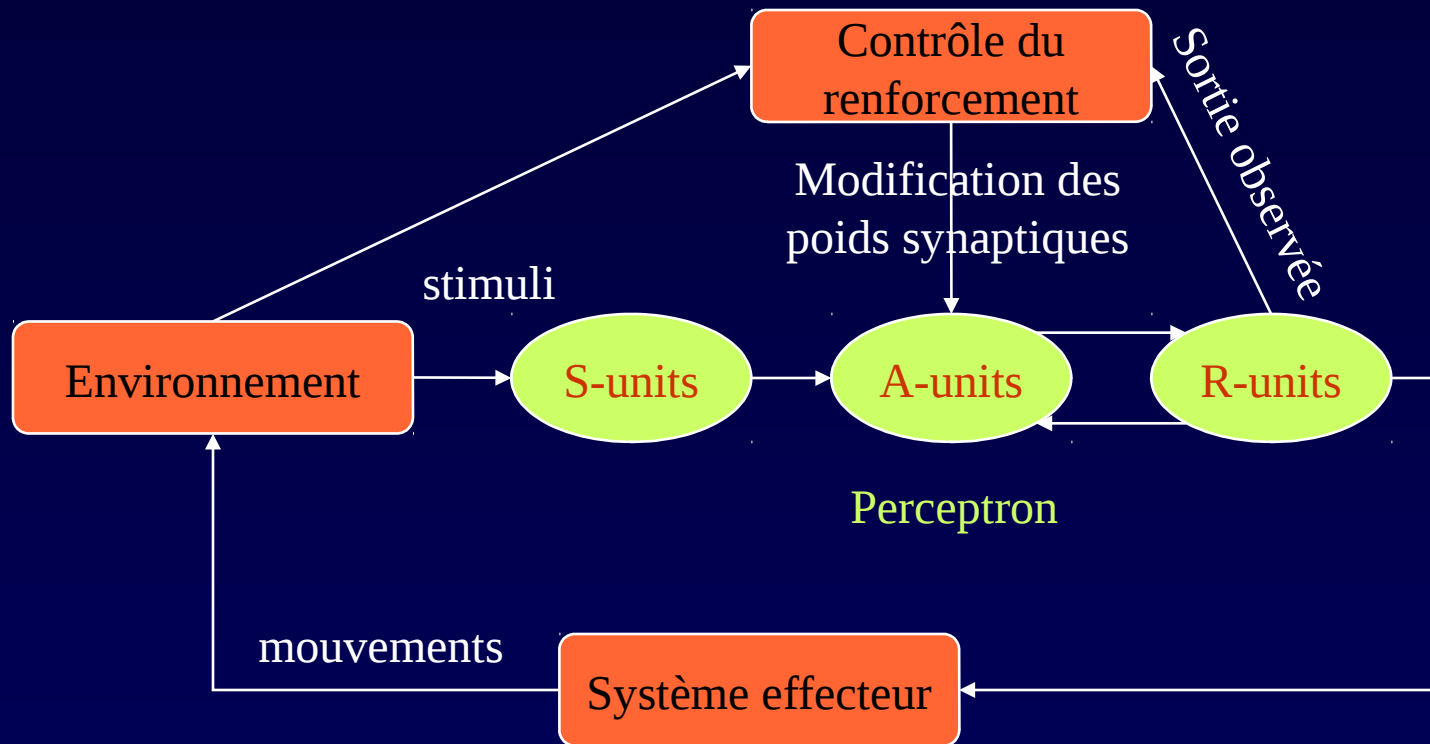
McCulloch/Pitts + poids synaptiques



Souvent, θ est considéré comme un « seuil », ou un « biais », avec deux façons de le faire apparaître :



Perceptron : apprentissage des poids



Perceptron d'après Rosenblatt, 1962
Apprentissage supervisé par renforcement

Règle de renforcement

$$w_{ij}(t+1) = w_{ij}(t) + \eta(d_j - s_j)s_i$$

Avec :

- $w_{ij}(t)$: poids de la connexion entre neurones i et j à t .
- η : pas d'apprentissage, variable, entre 0 et 1.
Généralement, démarrage à 0.8, et réduction progressive au fur et à mesure qu'on approche de la solution.
- d_j : sortie désirée
- s_j : sortie obtenue
- s_i : signal transmis par le neurone i .

Fonction de transfert et convergence

- ◆ Perceptron : fonction bipolaire.

$$s_j = f(p_j) = \begin{cases} 1 & \text{si } p_j \geq 0 \\ -1 & \text{si } p_j < 0 \end{cases}, \quad p_j = \sum w_{ij} s_i$$

- ◆ Rosenblatt a démontré la convergence du Perceptron en un temps fini (si une solution existe, le Perceptron la trouvera en un temps fini !).
- ◆ Mais en pratique, résultats décevants. Par ex : possible de reconnaître les lettres de l'alphabet, mais à la condition qu'elles soient toujours présentées de manière absolument identique. Manque de généralisation (surapprentissage).

Adaline

- ◆ Widrow et Hoff en 1960 : ADActive Linear NEuron. Différence essentielle avec le Perceptron : la fonction de transfert n'est plus bipolaire, mais linéaire : $f(p) = p$.
- ◆ Cette fonction est dérivable.
- ◆ Méthode d'apprentissage basée sur la minimisation de l'erreur quadratique (règle delta). Si le signal d'un neurone de sortie est $\sum w_{ij}s_i$, alors l'erreur sur l'exemple k est définie comme $1/2 (d_k - y_k)^2$, avec d_k et y_k les résultats désirés et obtenus sur l'exemple k . Sur m exemples, l'erreur totale est :

$$\frac{1}{2} \sum_{k=1}^m (d_k - y_k)^2$$

Règle delta

- ◆ La règle d'apprentissage de Widrow-Hoff permet d'apprendre d'après des informations locales.
- ◆ Il s'agit d'un algorithme de gradient (on modifie les poids à l'opposé de la dérivée de l'erreur par rapport au poids) mais sans connaissance globale du problème.
- ◆ Mais c'est une approximation : on n'utilise que les informations locales (entrée courante) et seulement à partir d'une suite aléatoire d'observations.
- ◆ Règle d'apprentissage pour l'exemple k :

$$w_{ij}(t+1) = w_{ij}(t) + \eta(d_{jk} - s_{jk})s_i$$

Différence Perceptron / Règle Delta

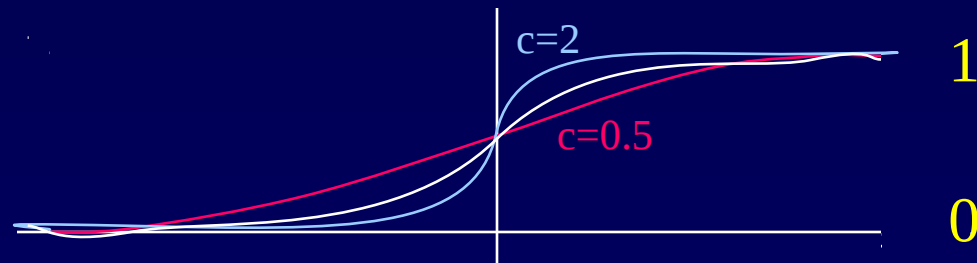
- ◆ Différence avec Perceptron :
 - Pas forcément de convergence lorsque les classes sont séparables...
 - Le Perceptron s'arrête à la première solution valide, alors que la règle delta cherche un minimum.
 - Si les classes ne sont pas parfaitement séparables, le Perceptron ne trouve pas, alors que la règle delta converge vers une solution incorrecte.
- ◆ Bref, la règle delta est plus floue, ce qui correspond mieux aux applications réelles.

Réseaux multicouches

- ◆ Permet de faire mieux que le Perceptron ou Adaline, mais nécessite d'autres algorithmes d'apprentissage.
- ◆ En ajoutant des couches cachées, il a été prouvé que le Perceptron multicouche est un « approximateur universel ». Barron (1993) dit : « *Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini, possédant tous la même fonction de transfert, et un neurone de sortie linéaire.* »
- ◆ Le seul problème, c'est qu'on ne connaît pas de méthode générique permettant de concevoir un réseau répondant à un problème particulier, et notamment : on ne sait pas combien de neurones il faut pour obtenir une approximation d'une qualité donnée...

Perceptron multicouche et apprentissage

- ◆ Pas de lien direct entre couche d'entrée et couche de sortie : comment assigner l'erreur ?
- ◆ Rétropropagation du gradient : extension de la règle delta de l'Adaline, basée sur une méthode de gradient. Il faut donc que la fonction de transfert soit continue et dérivable.
- ◆ Fonction la plus couramment utilisée : sigmoïde (en forme de S) : $S_c(x) = \frac{1}{1+e^{-cx}}$



Sigmoïde vs tangente hyperbolique

- ◆ La fonction sigmoïde est intéressante, car sa dérivée vaut
$$S_c'(x) = c S_c(x) (1 - S_c(x))$$
mais elle donne une réponse sur $]0, 1[$
- ◆ En revanche, la fonction tangente hyperbolique donne une réponse sur $] -1, 1[$ et elle tend à mieux « homogénéiser » l'espace de recherche... (mais la dérivée vaut $4/(e^x + e^{-x})^2$).
- ◆ La sigmoïde ou tanh sont parfaitement adaptées à la réalisation de bonnes fonctions de transfert, mais leurs sorties ne sont pas entières. Il faut donc mettre des seuils d'acceptation (=1 si > 0.9 , par exemple).

Algorithme de rétropropagation du gradient

- ◆ Découvert simultanément en 1986 par Le Cun en France, et Rumelhart et McClelland aux US.
- ◆ Extension de la règle delta : règle delta généralisée.
- ◆ Pour $p_i^k = \sum w_{ji} s_j^k + \theta_i$ et $s_i^k = \varphi(p_i^k)$ avec :
 - p_i^k le potentiel du neurone i pour l'exemple k ,
 - w_{ji} le poids de la connexion du neurone j vers le neurone i ,
 - θ_i le seuil du neurone i ,
 - s_j^k le signal de sortie du neurone j pour l'exemple k ,
 - φ la fonction de transfert,

l'erreur sur l'exemple k est définie comme :

$$E^k = \frac{1}{2} \sum_{o=1}^O (d_o^k - s_o^k)^2$$

avec d_o^k le signal désiré pour l'exemple k sur le neurone de sortie o , et O le nombre de neurones sur la couche de sortie.

Fonction d'apprentissage

- ◆ Avec l'application de la règle delta, on définit le signal d'erreur sur neurone de sortie (o pour output) : $\delta_o^k = (d_o^k - s_o^k) \varphi'(p_o^k)$
- ◆ Et le signal d'erreur sur neurone caché h connecté au neurone suivant h'

$$\delta_h^k = \varphi'(p_h^k) \sum_{h'=0}^H \delta_{h'}^k w_{hh'}$$

- ◆ On en déduit la fonction d'apprentissage (avec η pas d'apprentissage) :

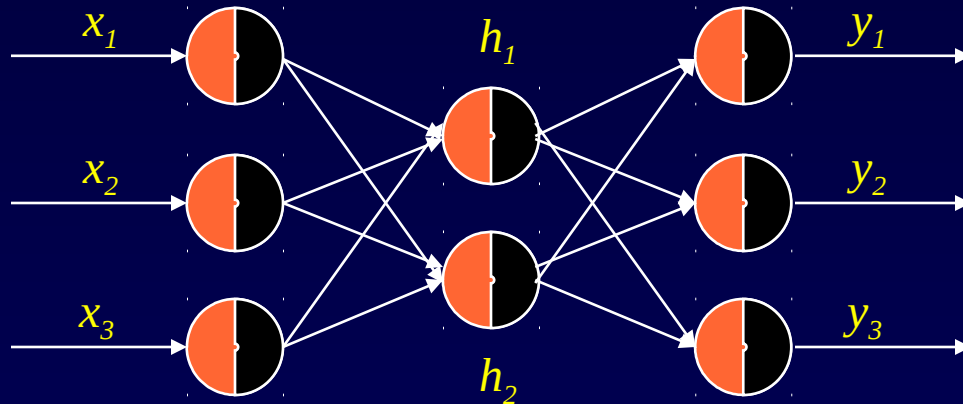
$$\Delta^k w_{ji} = \eta \delta_i^k s_j^k$$

- ◆ Ce qui permet de remettre les poids à jour avec la formule :

$$w_{ji}(t+1) = w_{ji}(t) + \Delta^k w_{ji}$$

Exemple avec réseau 3-2-3

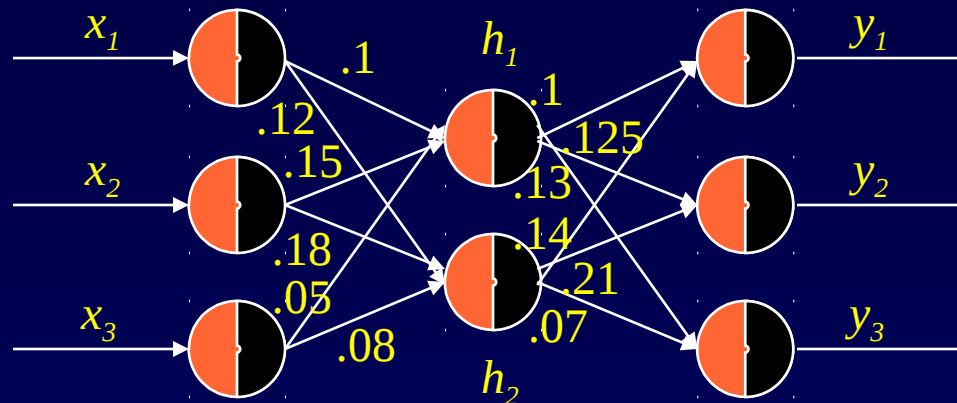
- ◆ 3 neurones sur la première couche, 2 neurones cachés, 3 neurones sur la dernière couche.



- ◆ Supposons que soumis à l'entrée $x_1=0.9$, $x_2=0.1$, $x_3=0.9$, le réseau doive ressortir $y_1=0.1$, $y_2=0.9$, $y_3=0.9$.

Exemple 3-2-3 : Initialisation

1. On initialise tous les poids aléatoirement. Il est recommandé de les distribuer uniformément sur un intervalle $[-m, +m]$ de manière à ce que la moyenne soit nulle (pour accélérer l'apprentissage).



Exemple 3-2-3 : Propagation avant

2. On présente le premier exemple (0.9, 0.1, 0.9) aux neurones d'entrée.
- Les neurones cachés reçoivent les valeurs pondérées par les arcs et calculent les potentiels :
 - $p_{h1} = 0.9 \times 0.1 + 0.1 \times 0.15 + 0.9 \times 0.05 = 0.150$
 - $p_{h2} = 0.9 \times 0.12 + 0.1 \times 0.18 + 0.9 \times 0.08 = 0.198$
 - Chaque neurone caché calcule son signal (avec une sigmoïde de paramètre 1 aussi appelée fonction logistique) :
 - $s_{h1} = \frac{1}{1 + e^{-0,150}} = 0.537$
 - $s_{h2} = \frac{1}{1 + e^{-0,198}} = 0.549$

Exemple 3-2-3 : Propagation avant (2)

3. On propage les résultats vers les neurones de sortie, dont les potentiels valent alors :

$$- p_{y1} = 0.537 \times 0.1 + 0.549 \times 0.14 = 0.131$$

$$- p_{y2} = 0.537 \times 0.125 + 0.549 \times 0.21 = 0.182$$

$$- p_{y3} = 0.537 \times 0.13 + 0.549 \times 0.7 = 0.108$$

➤ Cela permet de calculer le signal de chacun des neurones de sortie :

$$- s_{y1} = \frac{1}{1 + e^{-0,131}} = 0,533$$

$$- s_{y1} = \frac{1}{1 + e^{-0,182}} = 0,545$$

$$- s_{y1} = \frac{1}{1 + e^{-0,108}} = 0,527$$

Exemple 3-2-3 : Calcul de l'erreur

- ◆ Si l'on utilise la formule : $E^k = \frac{1}{2} \sum_{o=1}^O (d_o^k - s_o^k)^2$
l'erreur pour chaque neurone vaut :
 - Pour y_1 : 0.1 (désirée) – 0.533 (réelle) = -0.433
 - Pour y_2 : $0.9 - 0.545 = 0.355$
 - Pour y_3 : $0.9 - 0.527 = 0.373$

- ◆ Ce qui donne comme valeur totale:

$$E = 0.5 (-0.433^2 + 0.355^2 + 0.373^2) = 0.226$$

Si l'erreur est encore considérée comme trop importante, alors on boucle, sinon, on s'arrête.

Exemple 3-2-3 : Calcul du signal d'erreur

Signal d'erreur sur chaque neurone de sortie :

$$\delta_0 = (d_0 - s_0) \varphi'(p_0)$$

- Calcul de $\varphi'(p_0)$: en se souvenant que la dérivée de la fonction de transfert est $s_0 (1 - s_0)$, on a :
 - $y_1 : 0.533 (1 - 0.533) = 0.249$
 - $y_2 : 0.545 (1 - 0.545) = 0.248$
 - $y_3 : 0.527 (1 - 0.527) = 0.249$
- On réinjecte pour calculer δ_0 :
- $\delta_{y1} = -0.433 \times 0.249 = -0.108$
- $\delta_{y2} = 0.355 \times 0.248 = 0.088$
- $\delta_{y3} = 0.373 \times 0.249 = 0.093$

Calcul du signal d'erreur sur couche cachée

- ◆ Sur chacun des neurones de la couche cachée, le signal d'erreur vaut :

$$\delta_h^k = \varphi'(p_h^k) \sum_o \delta_o^k w_{ho}$$

- ◆ Le calcul de la dérivée se fait⁰ comme précédemment :

- $h_1 : 0.537 (1 - 0.537) = 0.249$

- $h_2 : 0.550 (1 - 0.550) = 0.247$

- ◆ Pour chaque neurone caché, il faut maintenant faire la somme des signaux d'erreur des neurones de sortie liés, pondérés par les coefficients synaptiques, soit :

- $h_1 : -0.108 \times 0.1 + 0.088 \times 0.125 + 0.093 \times 0.130 = 0.0123$

- $h_2 : -0.108 \times 0.14 + 0.088 \times 0.21 + 0.093 \times 0.07 = 0.00987$

- ◆ Ce qui donne comme signaux d'erreur :

- $\delta_{h1} = 0.249 \times 0.0123 = 0.0031$

- $\delta_{h2} = 0.247 \times 0.00987 = 0.0024$

Correction des poids de la couche de sortie

On peut enfin utiliser la fonction d'apprentissage

$$\Delta^k w_{ji} = \eta \delta_i^k s_j^k$$

qui permet de mettre les poids à jour avec la formule

$$w_{ji}(t+1) = w_{ji}(t) + \Delta^k w_{ji}$$

soit, avec η (le pas d'apprentissage) à 1 pour la couche de sortie :

- $w_{h1,y1} : 0.100 + (-0.108 \times 0.537) = 0.042$
- $w_{h1,y2} : 0.125 + (0.088 \times 0.537) = 0.172$
- $w_{h1,y3} : 0.130 + (0.093 \times 0.537) = 0.180$
- $w_{h2,y1} : 0.140 + (-0.108 \times 0.550) = 0.081$
- $w_{h2,y2} : 0.210 + (0.088 \times 0.550) = 0.258$
- $w_{h2,y3} : 0.070 + (0.093 \times 0.550) = 0.121$

Correction des poids de la couche cachée

Avec η (le pas d'apprentissage) à 1, on a pour la couche cachée :

- $w_{x1,h1} : 0.100 + (0.0031 \times 0.9) = 0.103$
- $w_{x2,h1} : 0.150 + (0.0031 \times 0.1) = 0.150$
- $w_{x3,h1} : 0.050 + (0.0031 \times 0.9) = 0.053$
- $w_{x1,h2} : 0.120 + (0.0024 \times 0.9) = 0.122$
- $w_{x2,h2} : 0.180 + (0.0024 \times 0.1) = 0.180$
- $w_{x3,h2} : 0.080 + (0.0024 \times 0.9) = 0.082$

Avec ces nouveaux poids, l'erreur totale qui était de 0.226 passe à 0.210. On s'est donc amélioré et il faut continuer jusqu'à arriver à une erreur acceptable.

Conclusion sur les Perceptrons Multicouches

- ◆ La rétropropagation fonctionne bien, et le Perceptron Multicouches présenté précédemment est le plus utilisé. Cependant :
 - La complexité algorithmique est un handicap. L'algorithme de rétropropagation est lourd, et converge lentement si le problème est un peu difficile (temps d'apprentissage en $O(n^3)$ avec n le nombre de connexions dans le réseau).
 - Aucun moyen de savoir combien il faut de neurones, combien de couches cachées, quelle taille, quelles interconnexions...

Dimensionnement pratique d'un ANN

- ◆ En pratique, on utilise :
 - Une méthode d'accroissement de la taille, qui consiste à partir d'un réseau minimum, auquel on ajoute des neurones ou des couches pour arriver à un réseau capable de résoudre le problème posé (*cf. méthode « cascade-correlation »*).
 - A l'inverse, méthode de réduction de taille. On part d'un réseau complexe, et on supprime des connexions inutiles (*cf. méthode « weight decay »*).

Données d'apprentissage

- ◆ L'ordre de présentation des données a une très grande importance sur la vitesse de convergence (si on commence par emmener l'ANN n'importe où, c'est très difficile de revenir vers les bons poids après).
- ◆ Seul pb : aucun moyen de connaître quel est le meilleur ordre.
- ◆ Du coup, on présente les exemples dans un ordre aléatoire, et à chaque itération, on modifie l'ordre de manière aléatoire.

Taille des données

- ◆ Pour généraliser, il est essentiel d'avoir suffisamment de données. Si pas assez d'exemples différents, le réseau va apprendre « par cœur », avec des problèmes de généralisation.
- ◆ Inversement, plus il y a de données, mieux on généralisera, mais le risque existe d'introduire plus de bruit (auquel les RN sont sensibles) qui va faire baisser la capacité de généralisation...
- ◆ Pour un réseau à une couche cachée, *Widrow's rule of thumb* : $N = W / \mathcal{E}$, avec W le nombre de paramètres libres (à optimiser) et \mathcal{E} l'erreur relative sur la généralisation. Si l'on accepte une erreur de 10%, il faut présenter 10x plus d'exemples que de paramètres libres.

Normalisation des entrées

- ◆ Les valeurs des entrées peuvent être éminemment variables, avec un problème de convergence, car les valeurs élevées auront une influence plus grande sur le réseau que les valeurs faibles.
- ◆ Il faut donc « normaliser » les entrées avant de les utiliser.
- ◆ Méthode simple : pour chaque vecteur d'entrée I_n , on calcule la moyenne μ_n et l'écart type σ_n , et on fait la transformation $I'_n = (I_n - \mu_n) / \sigma_n$, qui permet d'avoir une distribution de moyenne 0 et d'écart-type 1.

« Pas » d'apprentissage η

- ◆ Il est essentiel au bon fonctionnement de la rétropropagation. Trop élevé, des oscillations apparaissent. Trop faible, il ralentit la convergence.
- ◆ Idéalement, il doit être élevé dans les descentes, et se réduire lorsqu'une oscillation (un changement de signe de la dérivée de la fonction d'erreur) se produit.
- ◆ Algo original de Rumelhart propose un pas d'apprentissage global, mais certains proposent un pas d'apprentissage par connexion (méthodes *delta-bar-delta* et *Rprop*).
- ◆ Des algorithmes d'ordre 2 (utilisant les dérivées secondes) existent, basés sur la méthode de Newton, utilisant un développement de Taylor du second ordre.

Réseaux récurrents

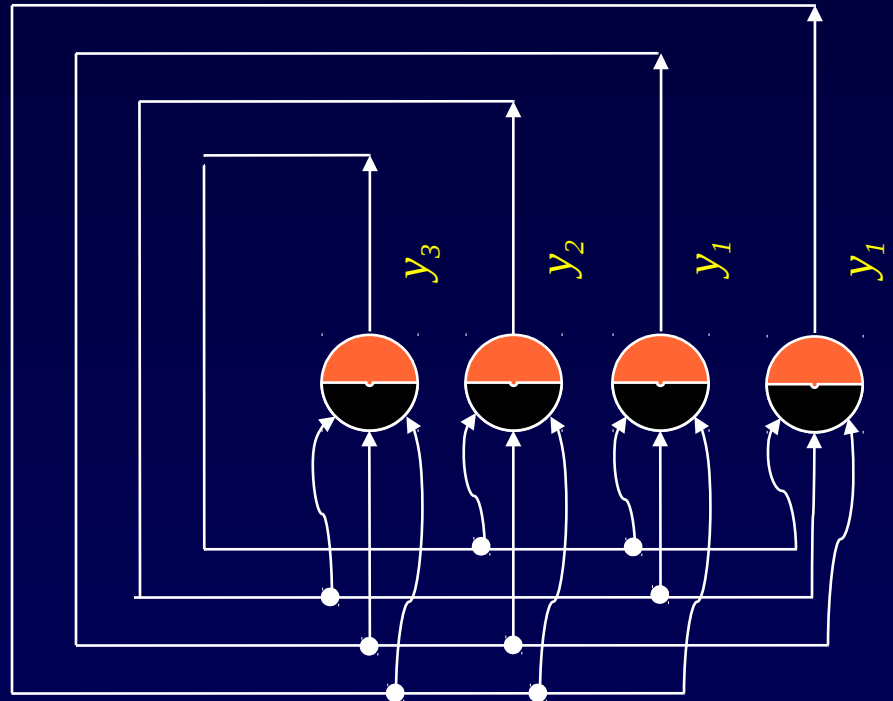
Réseaux de Hopfield

- ◆ But : implémentation d'une mémoire auto-associative (qui permet de faire correspondre des entrées X_i à des sorties, qui sont identiques aux entrées...)
- ◆ Si les entrées et les sorties sont des mots, alors un réseau auto-associatif pourra, à partir d'une phrase partielle (ou légèrement erronée) pointer sur la phrase correcte. Ce type de mémoire s'apparente à la mémoire humaine, qui est capable de généraliser, et de reconnaître quelqu'un en voyant partiellement son visage.

Structure d'un réseau de Hopfield

- ◆ Réseau totalement connecté hors connexions réflexives.
- ◆ Fonction de transfert :
$$s_j = \text{sgnb}(\sum w_{ij} s_i - \theta_j)$$

avec $\text{sgnb} = 1$ ou -1
suivant le signe de son opérande.
- ◆ Les neurones sont mis à jour aléatoirement, mais une seule fois par cycle.



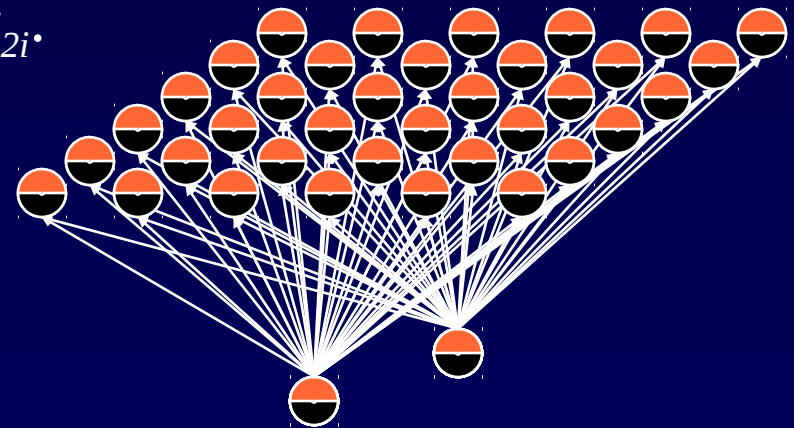
Performance des réseaux de Hopfield

- ◆ En tant que mémoires auto-associatives, les réseaux de Hopfield ont une « capacité de stockage » de l'ordre de : $N / 4 \log N$, avec N le nombre de cellules du réseau (= la taille des entrées).
- ◆ Ces réseaux ont une capacité à reconnaître des données bruitées ou partielles, mais aussi une capacité à mémoriser des choses fausses, et aussi à « oublier » (!) ce qui les rend intéressants pour les neuro-biologistes.

Réseaux auto-organisés

Modèle de Kohonen

- ◆ Utilisation : classification automatique non supervisée d'un ensemble d'entrées.
- ◆ Une seule couche de neurones (en plus de la couche d'entrée), habituellement organisée en 2D (histoire de créer une « carte de Kohonen »).
- ◆ A chaque neurone i sont associées 2 entrées via des connexions de poids w_{1i} et w_{2i} .



Fonctionnement

- ◆ A chaque entrée un et un seul neurone est activé (celui qui correspond le mieux).
- ◆ L'activation du neurone influence ses voisins directs, pour effectuer un couplage « latéral ».
- ◆ Adaptation suivant la règle de Hebb $w_i(t+1) = w_i(t) + \eta(t) \Phi(i,k,t)(x(t) - w_i(t))$

où η représente le pas d'apprentissage, et $\Phi(i,k,t)$ est la fonction de voisinage permettant d'ajuster l'apprentissage en fonction de la distance au neurone gagnant.