INTELLIGENCE ARTIFICIELLE
ARTIFICIAL INTELLIGENCE
APPLICATION A LA DEMONSTRATION DE LA TABLE DE VERITE EN LOGIQUE MATHEMATIQUE

par Laurent Cancé Francis
10/07/2008
fin: 5/12/2008
ajout: 5/12/2010

The present document is part of the package to understand what can be an artificial intelligence with some application to logic.

Le présent document est une partie de l'ensemble des documents pour comprendre ce que pourrait être une intelligence artificelle avec application à la logique, à la démonstartion de la table de vérité.

The following is in french.

L'algorithme résoud le problème du pattern recognition, vous trouverez ici l'application à la démonstation de la table de vérité.

Algorithme d'intelligence artificielle

1.
considérer une matrice nxn.
symbolisation des concepts.
injection des concepts perçus.
ordonnancement hierarchique des concepts.
principe du jeu de la vie sur la matrice nxn.

2.
principe:

INPUT: considérer une matrice nxn.
symbolisation (stockage hierarchique) des concepts.
opérations sur les symboles => nouveau concepts
=> signification conceptuelle

injection du résultat dans une seconde matrice:

durée de vie de chaque élément de la matrice
évolution de proche en proche
=> émotionnel
(ex: jeu de la vie, chaos, idée de propagation chimique)
+réinjection de concepts par confrontation aux concepts résultants
=>état

OUTPUT: "opération sur les 2 matrices, stockage hiérarchique du résultat"

association a de nouveaux concepts par nouvelles données en INPUT.
+association d'une partie des concepts résultants à un ensemble de concepts

différenciation des concepts purs et résultants sur la valeur des éléments (ex: 0,1 ou [0..1])

hierarchisation des concepts par ordonnacement

exemple:
animal de compagnie*2 + declinaison chien = chien
= concept ordre 2 = ordre 1 déclinaison

raisonnement sur chaque couche, ordre des concepts.

principes de logiques

//

A=>B

2 manières d'écrire:

B-A => table de vérité.

A-B

//

Methode 1:

//

A=>B ou A=>C
ie A ou B=>C

stockage de A-B et de A-C

si C=>D ?

B-A+D-C [-2..2] ordre 2
C-A+D-C = D-A [-1..1] ordre 1

//

(A et B)=>C C-A*B
C=>D

D-A*B

évaluation:

A vrai A=All1
B faux B=All0

liste des concepts:

A,1
B,0
C,?
D,?
C-A*B,1
D-A*B,1

(C-A*B),1 + C,1 = A*B,1

! B,0 -A*B,1 inconnu
A,1-A*B,1 : inconnu
A*B,1 / A,1 = B,1 : inconnu dans l'état
! A*B,1 / B,0 : impossible
idem D-A*B,1 : inconnu

=> C,?=C,0 !

//

A=>B
not (B=>C)

A-B
-(B-C)

A,1
B,?
C,?
B-A,1
-(B-A),0
-(C-B),1
B-C,0

B,1 ?
B,1 - A,1 = B-A,1 => B,1

c,1 ?
C,1 - (-(C-B),1) = -(B,1) = B,0 => C,0

C,0
(-(C-B),1) - C,0 ?
C-B,0 + C,0 =B,0 !

//

A=>B
not B=>C

B-A,1
C+B,1

A,1=>B,1

C,1?
C,1-(C+B,1) = -B,1 = B,0 => C,0

C,0 ?

C,0 -(C+B,1) = ordre 2
B,1 - (C+B,1) = -C,1 = C,0

ordre 2 : hierarchisation ?

///
Methode 2:
///

A=>B ou A=>C
ie A ou B=>C

stockage de A-B et de A-C

si C=>D ?

A-B+C-D	[0..2]	ordre 2
A-C+C-D = A-D	[0..1]	ordre 1

///

(A et B)=>C A*B-C
C=>D

A*B-D

évaluation:

A vrai A=All1
B faux B=All0

liste des concepts:

A,1
B,0
C,?
D,?
A*B-C,1
A*B-D,1

(A*B-C),1 + C,1 = A*B,1

! A*B,1 - B,0 = A*B,1 : inconnu
A*B,1 - A,1 : inconnu
A*B,1 / A,1 = B,1 : inconnu dans l'état
! A*B,1 / B,0 : impossible
idem A*B-D,1 : inconnu

=> C,?=C,0 !

//

A=>B
not (B=>C)

A-B
-(B-C)

A,1
B,?
C,?
A-B,1
-(A-B),0
-(B-C),1
B-C,0

B,1 ?
A,1 - B,1 = A-B,1 => B,1

c,1 ?
(-(B-C),1) - C,1 = -(B,1) = B,0 => C,0

C,0
(-(B-C),1) - C,0 ?
B-C,0 + C,0 =B,0 !

//

A=>B
not B=>C

A-B,1
-B-C,1

A,1=>B,1

C,1?
-B-C,1 + C,1 = -B,1 = B,0 => C,0

C,0 ?

-B-C,1 + C,0 = -B-C,1 - C,1 = -B-2*C,1 ordre 2
-B-C,1 + B,1 = -C,1 = C,0

-B-2*C,1 ordre 2 : hierarchisation ?

//

logique statistique:

A=>B
B=>C

prob(B)=prob(C)=prob(A)

A=>B
B=>C prob(B=>C)=p

prob(B)=prob(A)
prob(C)=prob(A)*prob(B=>C)

etc.

exemple:

A=>B prob(A=>B)=p1
not B=>C prob(not B=>C)=p2

A-B,1
-B-C,1

A,1 prob(A)=p0

A,1 prob(A)

=>B,1 prob(A)*prob(A=>B)

=> C,0 prob(A)*prob(A=>B)*prob(not B=>C)
=> C,1 prob(A)*prob(A=>B)*(1-prob(not B=>C))

=>B,0 prob(A)*(1-prob(A=>B))

=> C,1 (prob(A)*(1-prob(A=>B)))*prob(not B=>C)
=> C,0 (prob(A)*(1-prob(A=>B)))*(1-prob(not B=>C))

A,0 (1-prob(A))

=>B,1 (1-prob(A))*(1-prob(A=>B))

=> C,0 (1-prob(A))*(1-prob(A=>B))*prob(not B=>C)
=> C,1 (1-prob(A))*(1-prob(A=>B))*(1-prob(not B=>C))

=>B,0 (1-prob(A))*prob(A=>B)

=> C,1 (1-prob(A))*prob(A=>B)*prob(not B=>C)
=> C,0 (1-prob(A))*prob(A=>B)*(1-prob(not B=>C))

prob(C,0) =
prob(A)*prob(A=>B)*prob(not B=>C)+
(prob(A)*(1-prob(A=>B)))*(1-prob(not B =>C))+
(1-prob(A))*(1-prob(A=>B))*prob(not B=>C)+
(1-prob(A))*prob(A=>B)*(1-prob(not B=>C))

prob(C,0)= 4.p0.p1.p2 -2.(p1*p2+p0.p1+p0.p2) + p0 + p1 + p2
prob(C,1)= -4.p0.p1.p2 +2.(p1*p2+p0.p1+p0.p2) - p0 - p1 - p2 +1

p0=1 p1=p2
prob(C,1)=0

p1=1 p0=p2
prob(C,1)=2.p0.(1-p0)
p0=p2=1/2
prob(C,1)=1/2

p0=p1=1 p2=1/2
prob(C,1)=1/2

1	X	X	0
1/2	1	1/2	1/2
1	1	1/2	1/2

//

A=>B

singleton:
B-A:

0-0	=	0	A=B=0
1-1	=	0	A=B=1
1-0	=	1	A=0 B=1
0-1	=	-1	A=1 B=0

A-B:

0-0	=	0	A=B=0
1-1	=	0	A=B=1
0-1	=	-1	A=0 B=1
1-0	=	1	A=1 B=0

(A=>B) => (A=>B)

(A-B) - (A-B) = 0

=> t(A=>B) = (1-(B-A)) ou (1-(A-B)) ou ((1+(A-B)) ou (1+(B-A)))

t(A=>B): cas B-A:

1-(0-0)	=	1	A=B=0
1-(1-1)	=	1	A=B=1
1-(1-0)	=	0	A=0 B=1
1-(0-1)	=	2	A=1 B=0

t(A=>B): cas A-B:

1-(0-0)	=	1	A=B=0
1-(1-1)	=	1	A=B=1
1-(0-1)	=	2	A=0 B=1
1-(1-0)	=	0	A=1 B=0

A=>A

A-A=0

=> t(A=>A) = 1

A=>B
B=>A

t = t(A=>A) = t(B=>B) = 0 = t(A=>B et B=>A)
t = (A-B)*(B-A) = 2.A.B - A^2 - B^2

A B	A<=>B
0 0	0
1 1	0
0 1	-1
1 0	-1

table de vérité: (en logique classique)

0=>0 1
1=>1 1
0=>1 1
1=>0 0

En choisissant, t(A=>B) = (1-(A-B)), le raisonnement par l'absurde est modélisé sous la forme de deux opérateurs, donc d'ordre 2 et d'où l'unicité de cette modélisation.

Vulgarisation:

ordre 2 : tout est vrai et tout est faux
et:
ordre 0: on ne peut pas nier que du faux on ne peut déduire du faux (*)

la vérité est essentielle à la vie, c'est un algorithme récursif d'où la table de vérité.(*2)

(itérations sur la table de vérité)

(*) faux=>faux=>faux=>faux=>... n'a pas de terminaison et déduire n'est pas une simple implication !
(*2) cf. Horus