Intelligence Embarquée

Laboratoire ETIS
ENSEA - UCP- UMR CNRS 8051

Qu'est-ce qu'un système intelligent ?

Embarqué ou non ?
Embarqué ou embodied (incarné) ?

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

- Logics (inference, deduction, etc...)
- Reasoning
- Symbol manipulation
- Solution discovery
- Rules (rules based system, expert systems)
- Advanced Algorithmics
- State space search (depth first, A*, ...)

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

- Logics (inference, deduction, etc...)
- Reasoning
  - LISP [MIT,McCarthy,58]
  - Prolog [Colmerauer,60]
- Symbol manipulation
- Solution discovery
- Rules (rules based system, expert systems)
- Advanced Algorithmics
- State space search (depth first, A*, ...)

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

- Logics (inference, deduction, etc...)
- Reasoning
  - "computers can be used to manipulate symbols" [McCarthy 60]
- Symbol manipulation
  - "dialog with a computer ELIZA [Weiezbaum, 60]
- Solution discovery
- Rules (rules based system, expert systems)
- Advanced Algorithmics
- State space search (depth first, A*, ...)

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

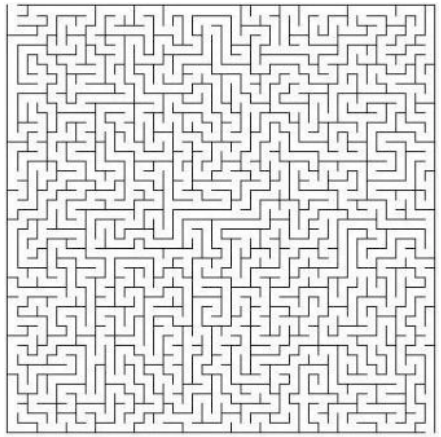
- Logics (inference, deduction, etc...)
- Reasoning
- Symbol manipulation
  - PLANNER : goal generation for program solving
- Solution discovery
  - GPS [Newel & Simon, 78]
- Rules (rules based system, expert systems)
  - 80's
  - Scripts language
- Advanced Algorithmics
- State space search (depth first, A*, ...)

Artificial Intelligence

Traditional opposition : *classical IA* / *connectionism*

- Logics (inference, deduction, etc...)
- Reasoning
- Symbol manipulation
- Solution discovery
- Rules (rules based system, expert systems)
- Advanced Algorithmics
- State space search (depth first, A*, ...)
  - A star [Hart & Al. 68]

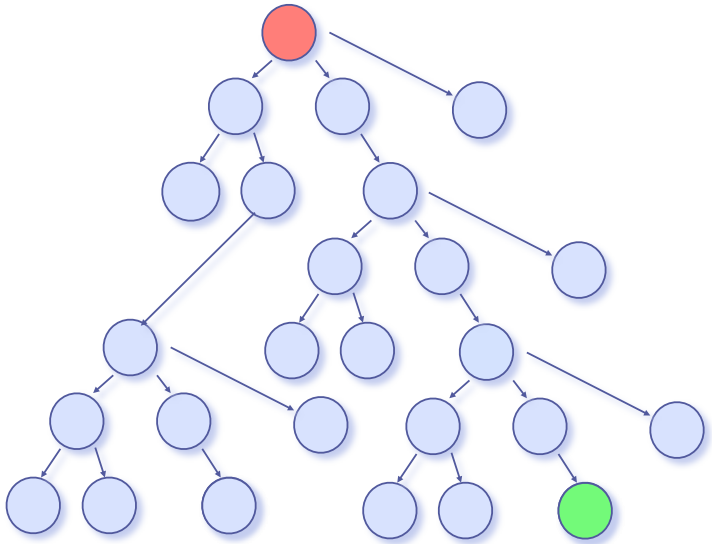
A*



A*

- On considère un espace d'états
- Résoudre un problème c'est
  - Partir d'un état particulier
  - Appliquer des opérations
  - Parvenir à l'état solution

A*



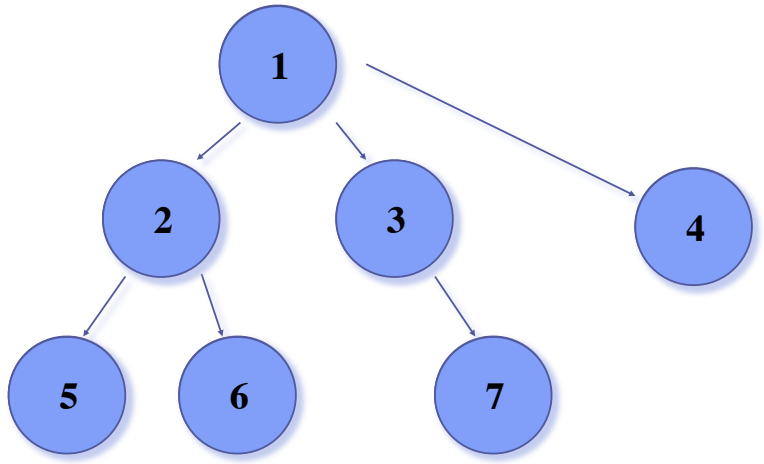
A*

- Simple
- Pas d'information a priori
- Sure
  - On sait qu'on trouvera la solution
- Problème:
  - temps de convergence

A*

- Parcours en largeur d'abord
- Parcours en profondeur d'abord

parcours largeur d'abord

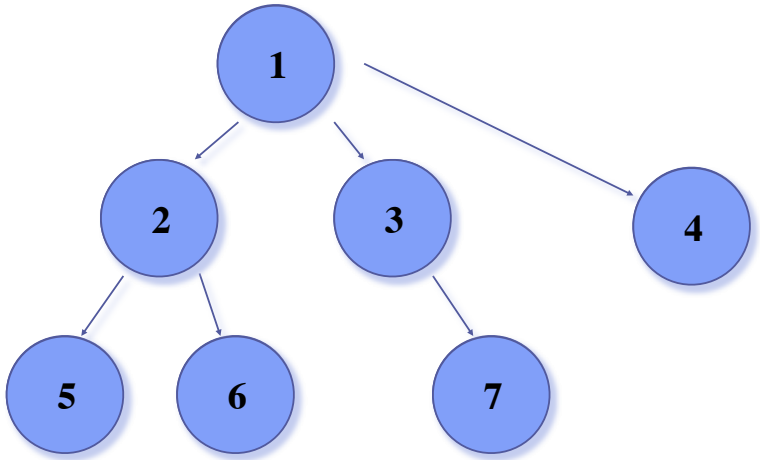


parcours largeur d'abord

1. Placer le sommet initial s0 dans OUVERT
2. tant qu'OUVERT n'est pas vide
   - soit n le premier élément d'OUVERT
   - mettre les fils de n en queue d'OUVERT
   - effectuer le traitement pour n
   - mettre n dans FERME

fin tant que

Parcours profondeur d'abord



Parcours profondeur d'abord

1. Placer le sommet initial s0 dans OUVERT
2. tant qu'OUVERT n'est pas vide et prof<prof_max
   - soit n le premier élément d'OUVERT
   - mettre les fils de n en tête d'OUVERT
   - effectuer le traitement pour n
   - mettre n dans FERME

fin tant que

A*

- Aider la recherche par une heuristique
  - Information a priori
  - « Distance » à la solution
- Fonction d'évaluation f(n)
  - Coût effectif g(n)
  - Coût restant h(n)

A*

Fonction d'évaluation f(n)

- Coût effectif g(n) : coût du parcours
- Coût restant h(n) : approximation du "coût" du parcours restant. i.e distance à l'objectif
- Attention : g et h sont exprimés selon de métriques différentes

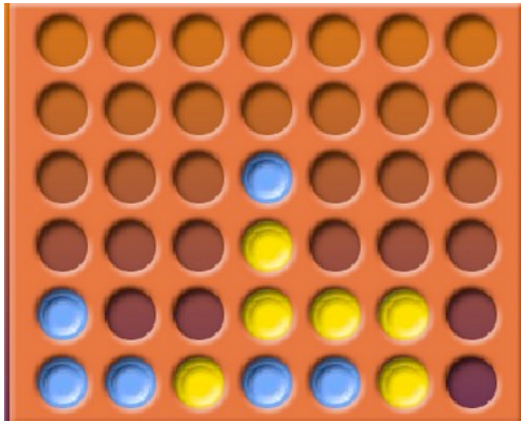
A*

Fonction heuristique : moyen d'estimer l'état courant par rapport à la solution

toute l'intelligence de la solution dépendra de l'heuristique

A*

Au tour des jaunes, qui gagne ?



A*

g doit être le plus petit possible : minimiser

le nombre de déplacement dans l'arbre (ou le graphe)

h doit être le meilleur possible (on se rapproche de la solution) -> plus h est petit,

meilleur est l'etat courant

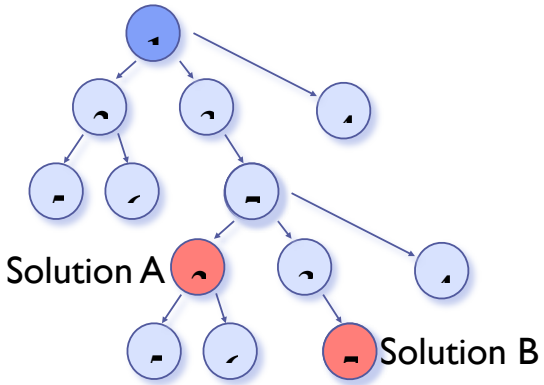
A*

Principe général :

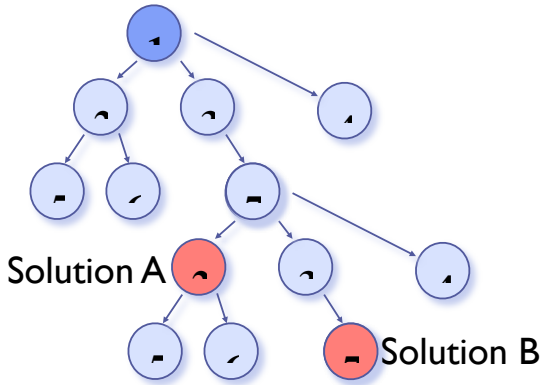
la valeur de f pour les état candidats va orienter l'exploration (choix du prochain état a traiter)

Pourquoi g+h ?

A*

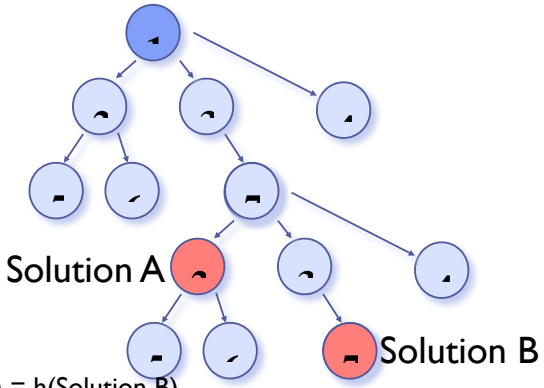


A*



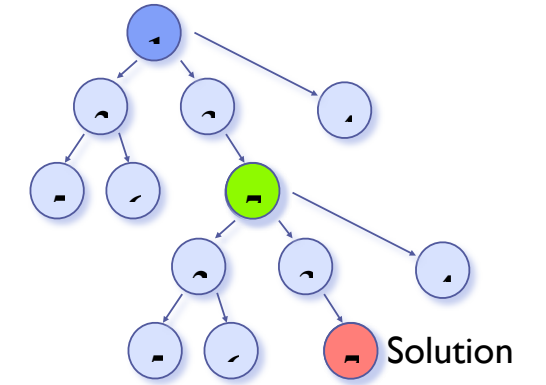
h(Solution A) = h(Solution B)

A*

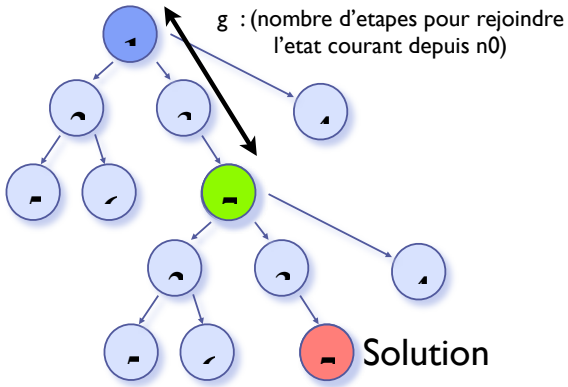


h(Solution A) = h(Solution B)
g(Solution A) < g(SolutionB)
onc fA<fB :A est une meilleure solution que B

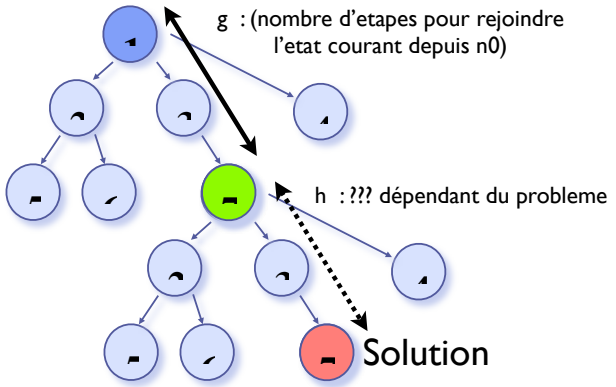
A*



A*

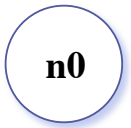


A*

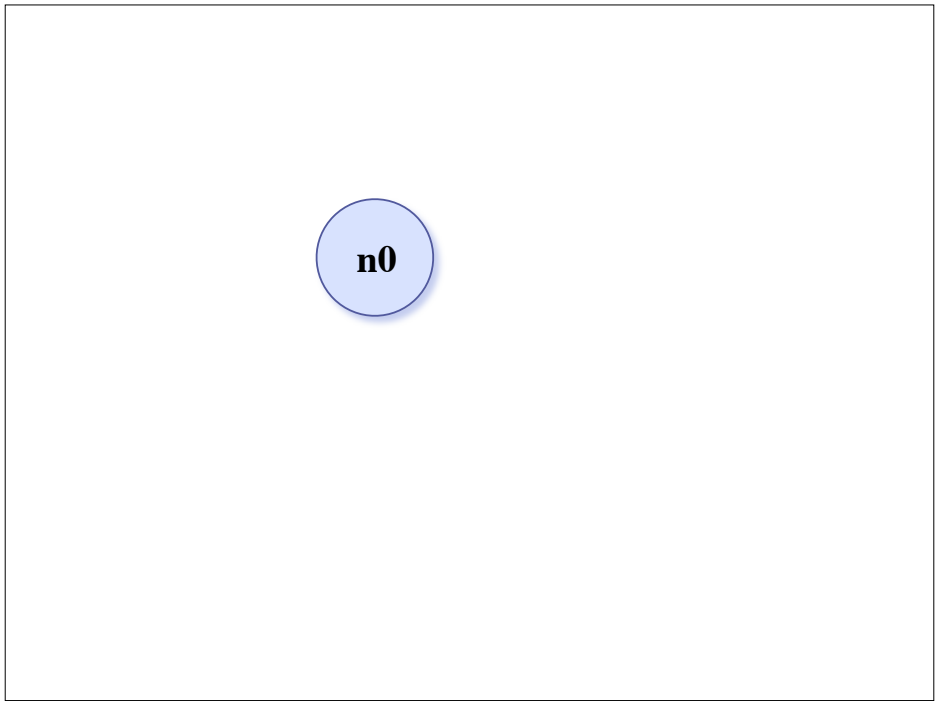


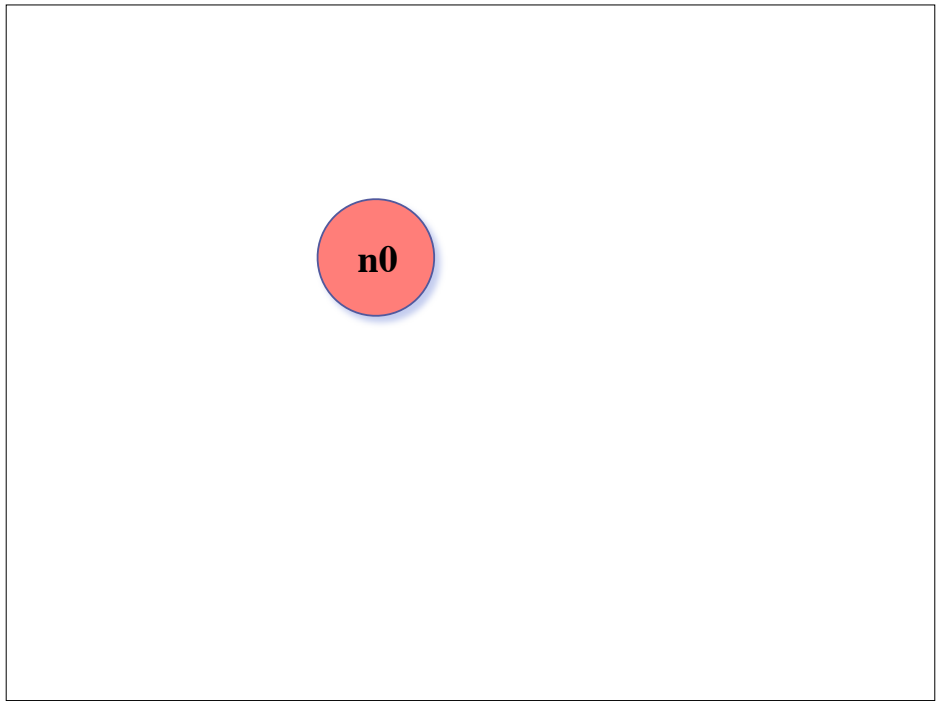
A*

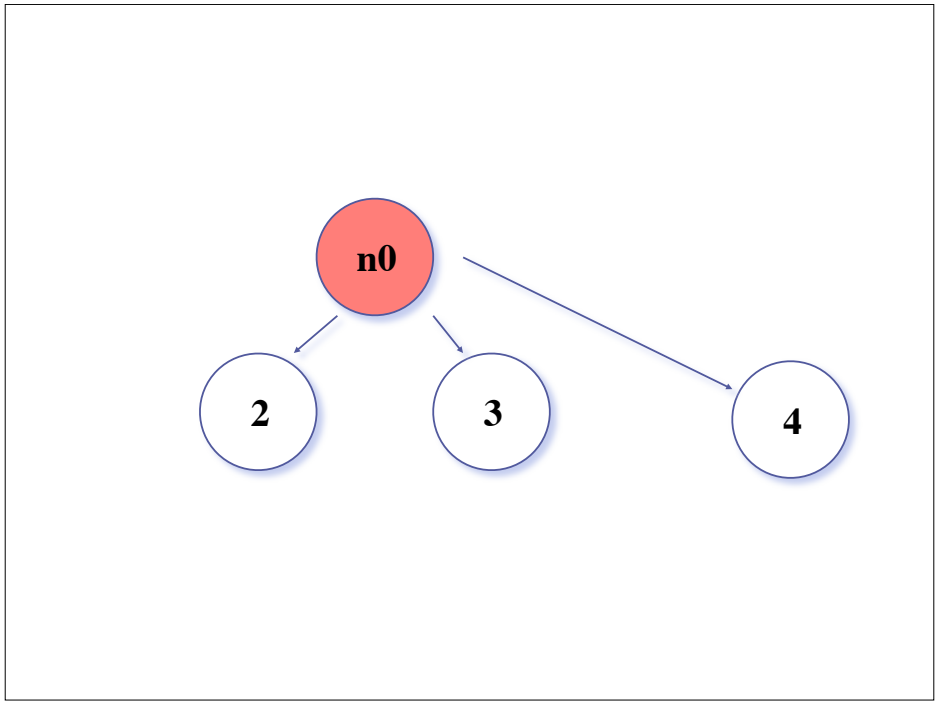
1. Placer le sommet initial n0 dans OUVERT; f(n0)=0; n=0
2. Si OUVERT vide => échec !
3. n = n+1; **le sommet ayant le plus faible f est mis dans FERME ; l'appeler n**
4. énumérer les successeurs du sommet n; les mettre a la fin de OUVERT et initialiser les pointeurs vers n. Si l' objectif atteint sortir.
5. Aller en 2

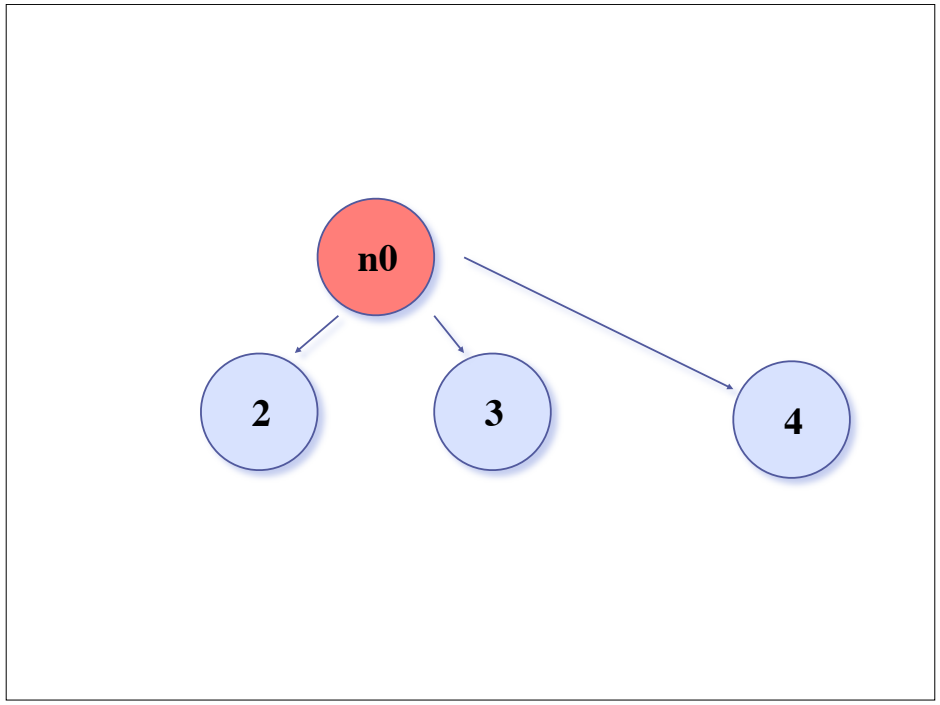


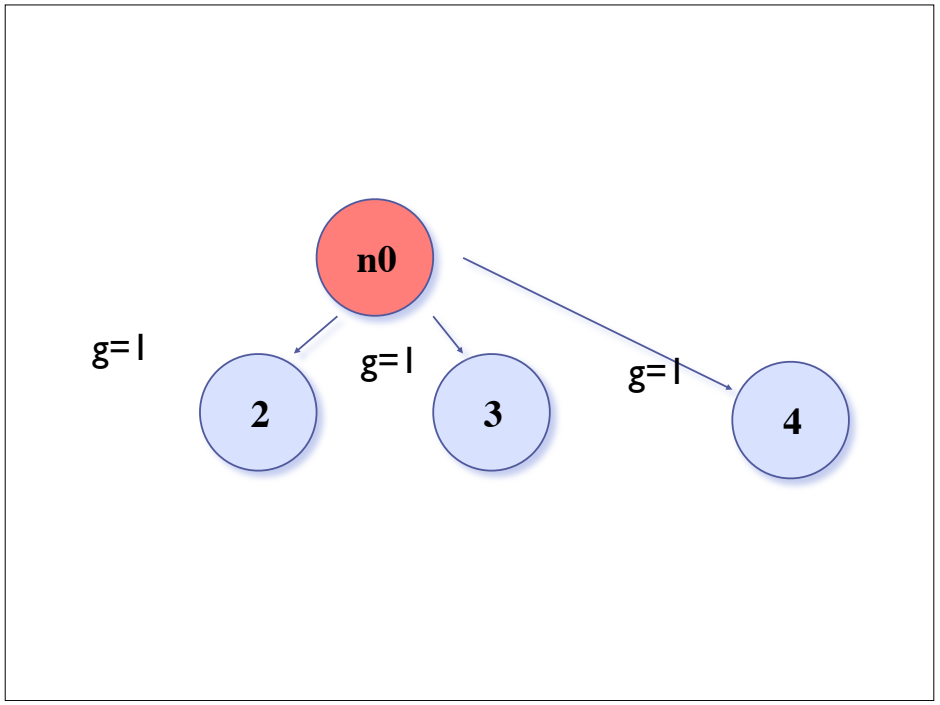
Blanc : etat généré
Bleu : dans ouvert
Rouge: dans fermé

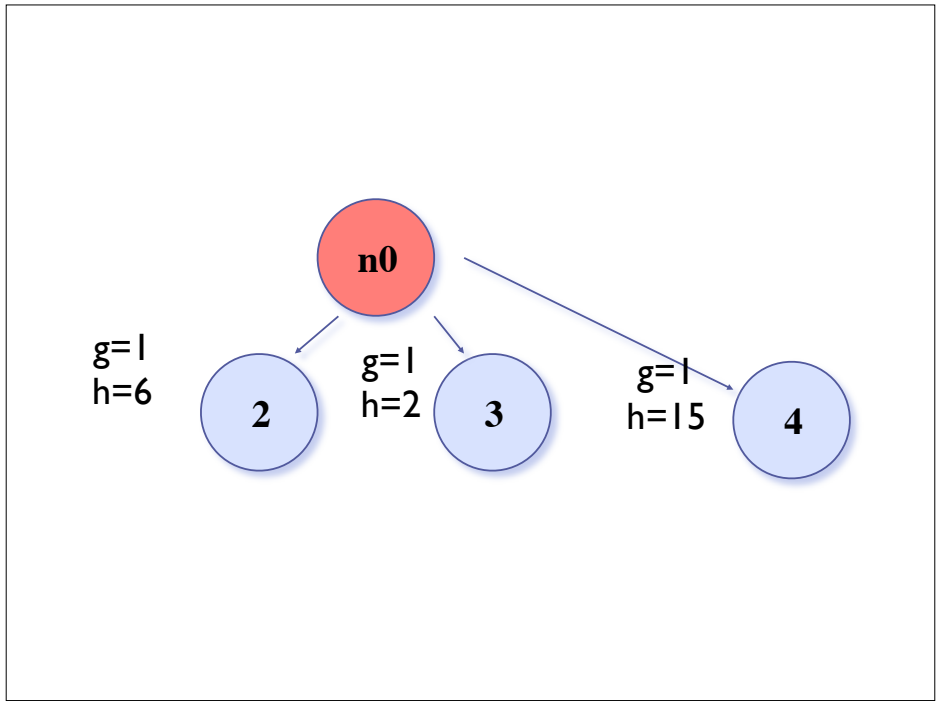

n0

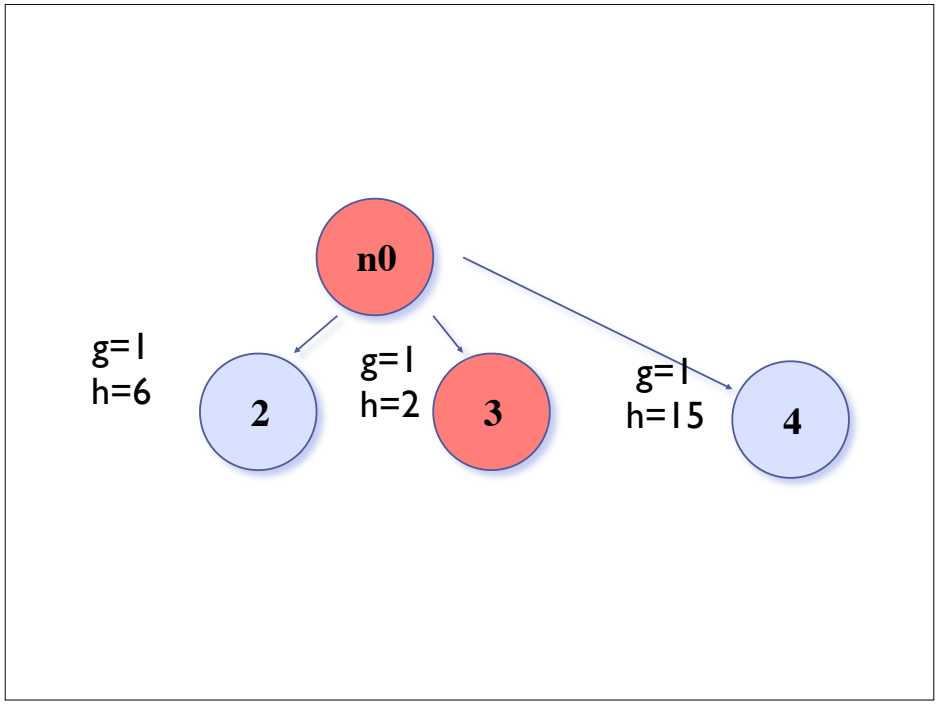

n0

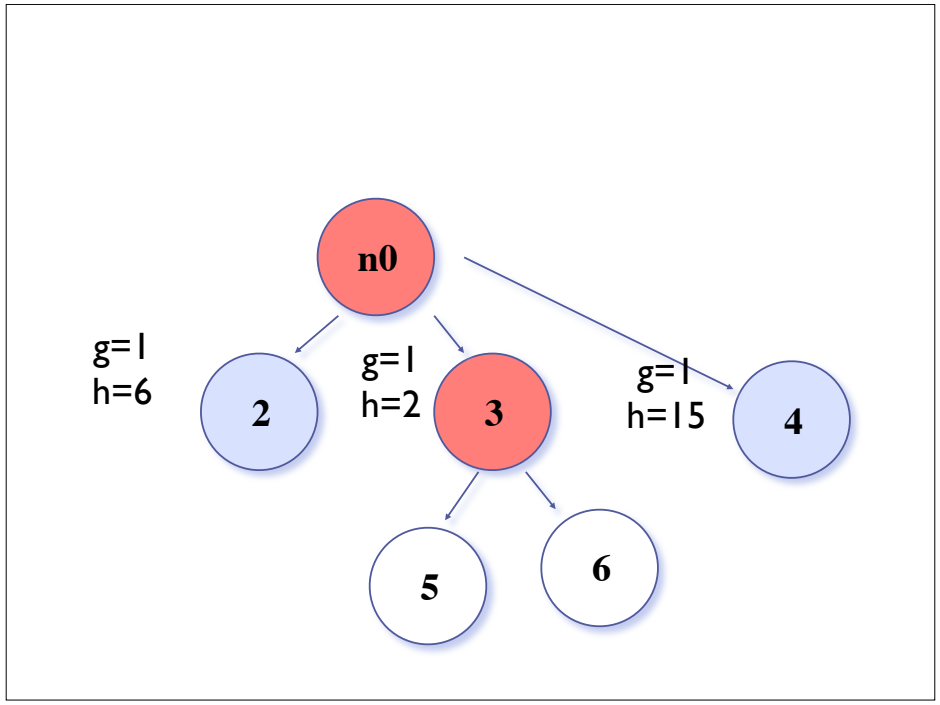

n0
2
3
4

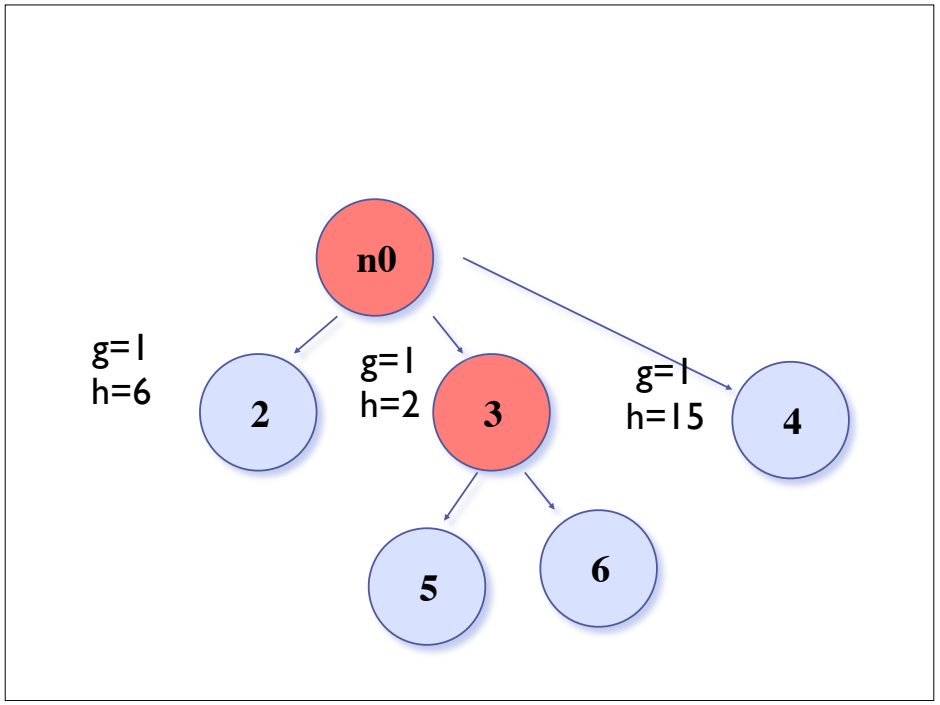

n0
2
3
4

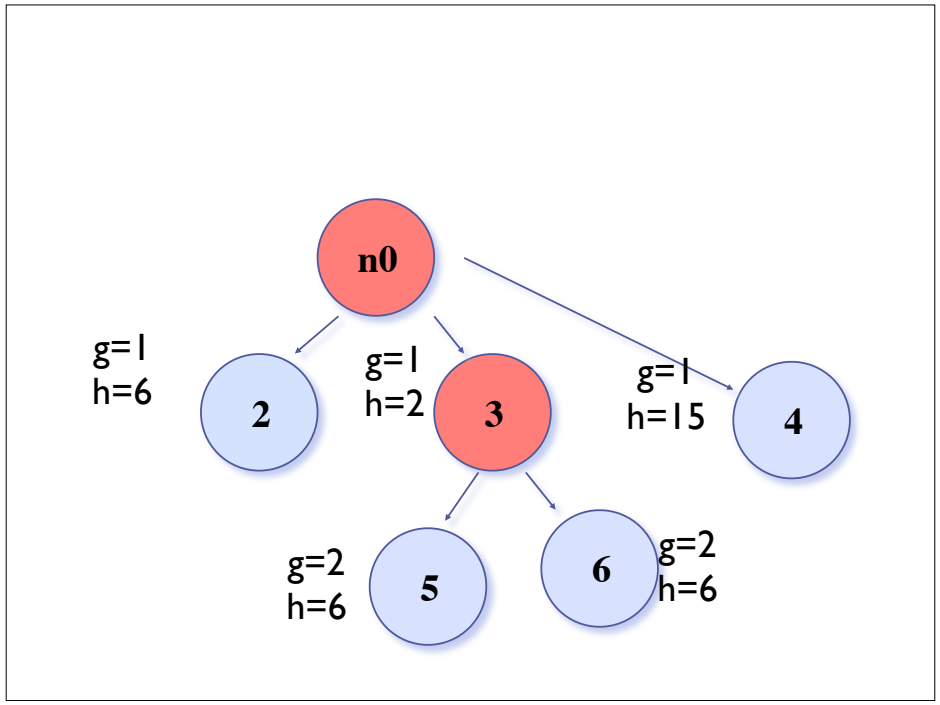

n0
g=1
2
g=1
3
g=1
4

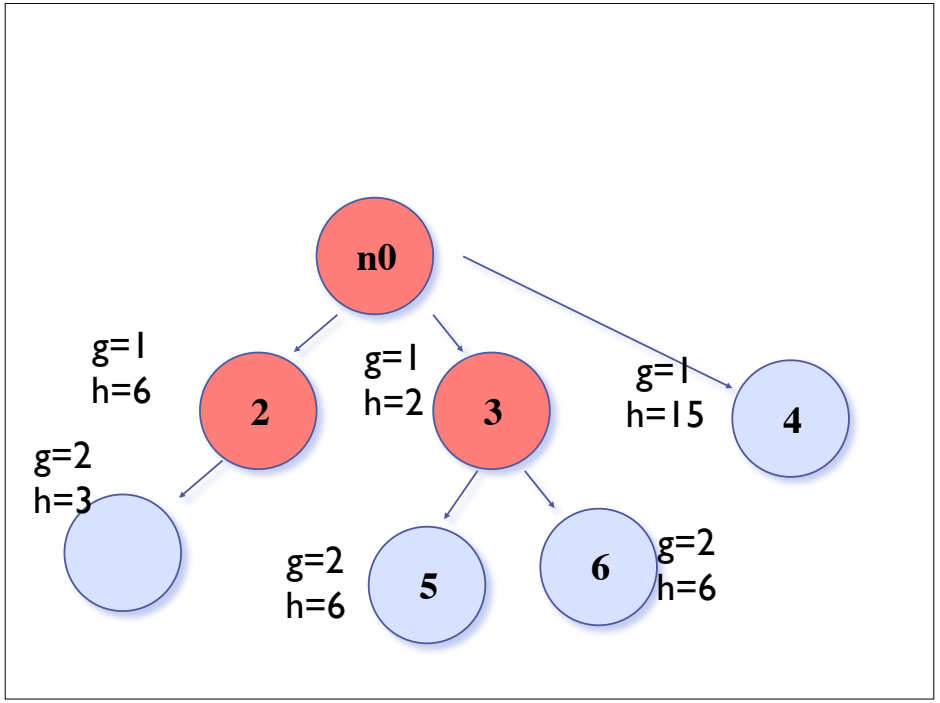

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4

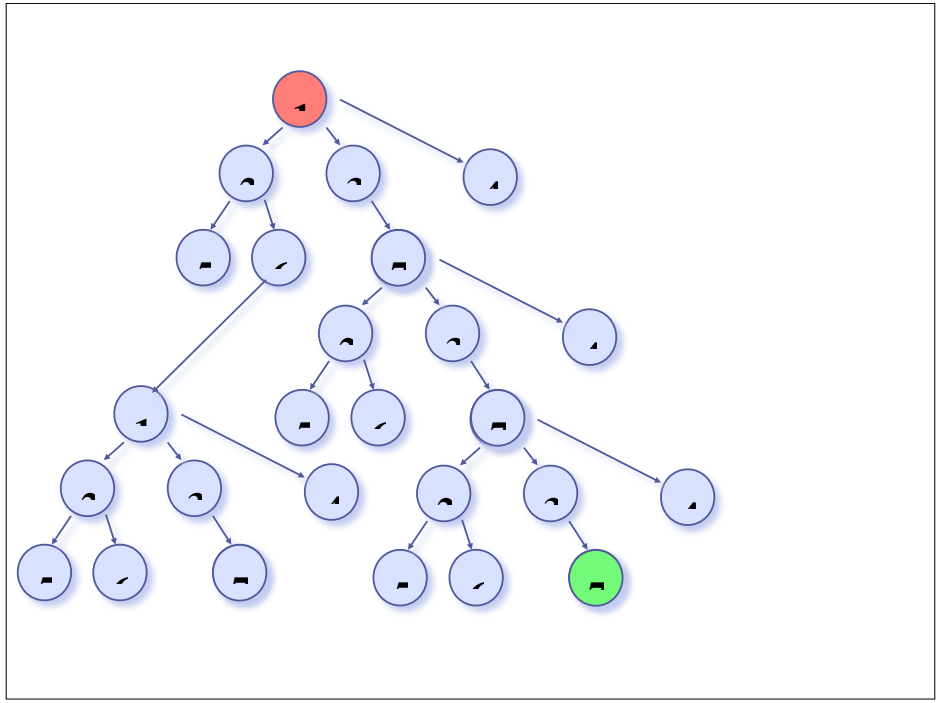

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4

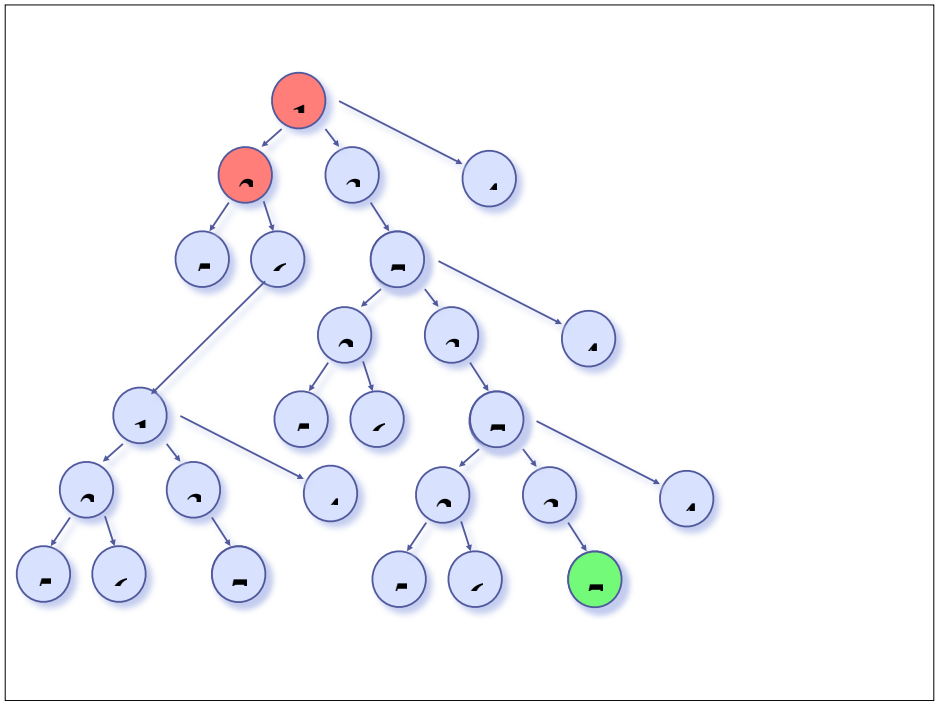

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4
5
6

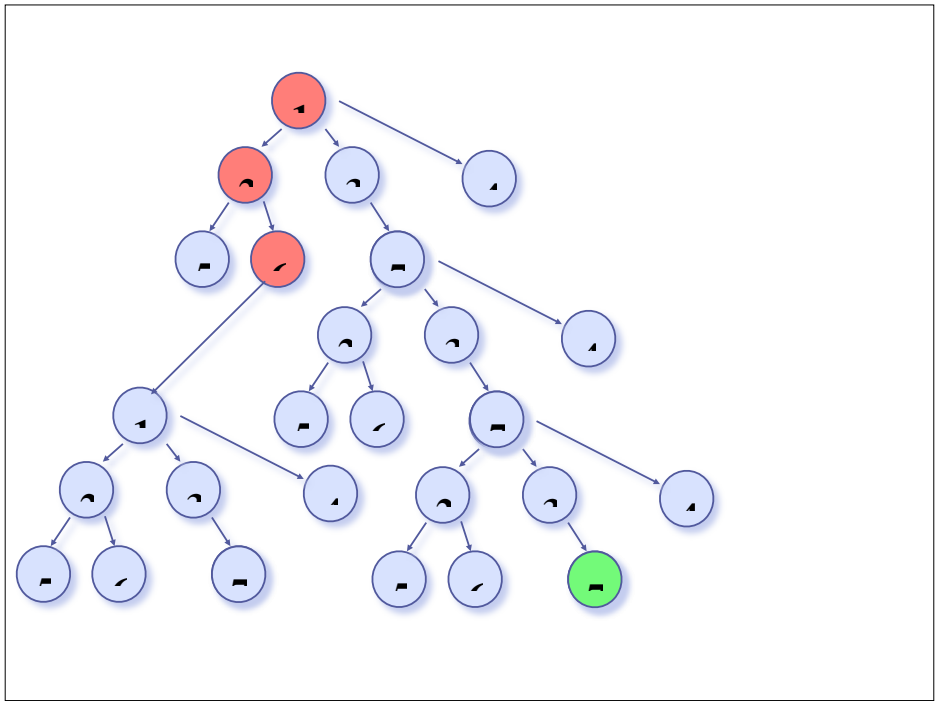

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4
5
6

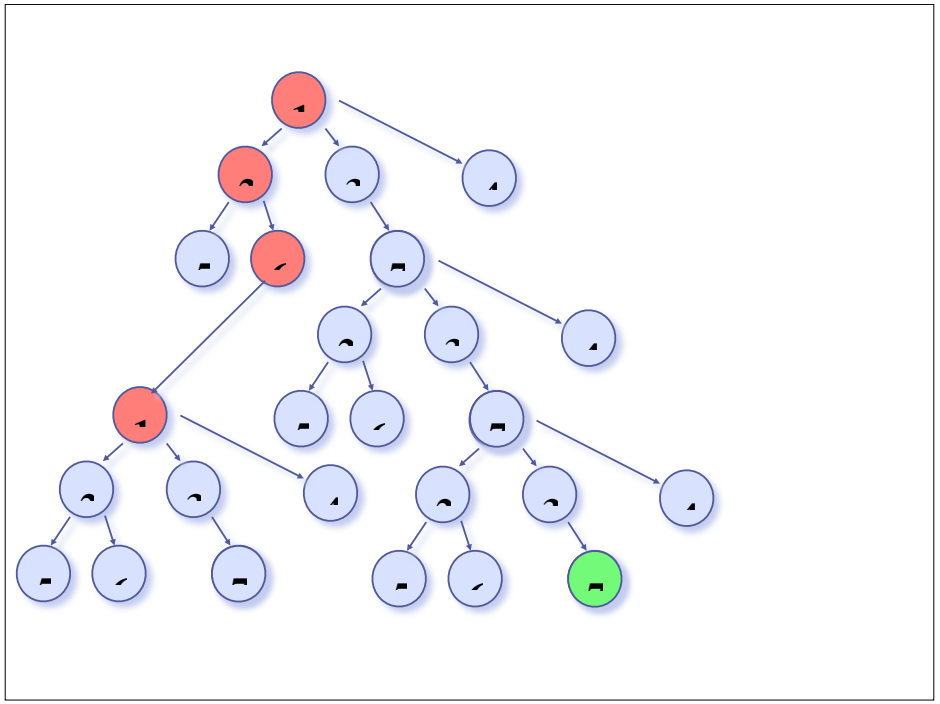

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4
g=2
h=6
5
6
g=2
h=6

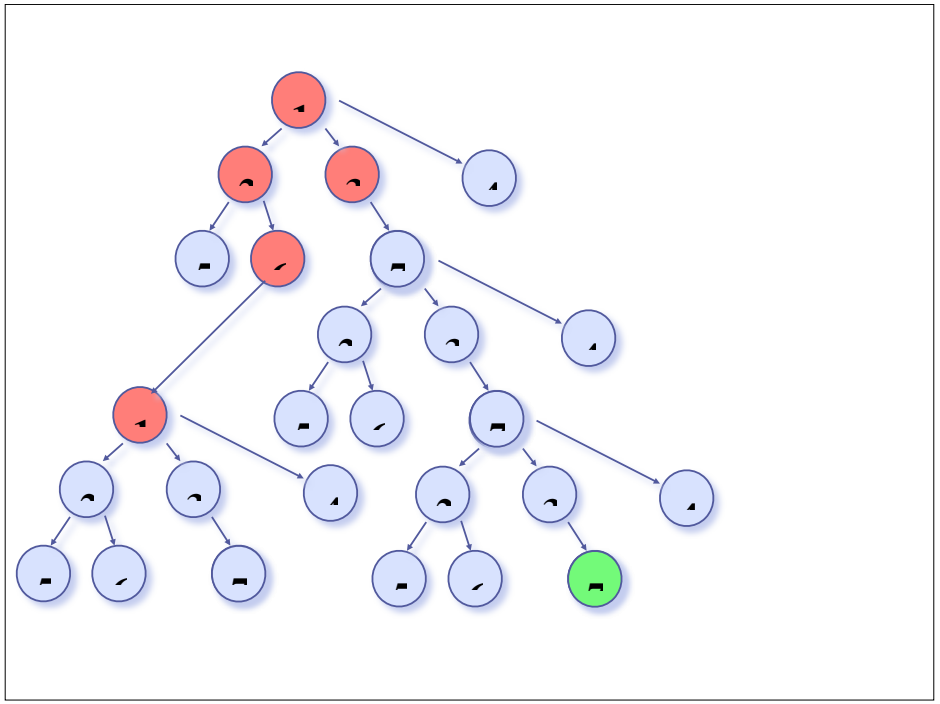

n0
g=1
h=6
2
g=1
h=2
3
g=1
h=15
4
g=2
h=3
g=2
h=6
5
6
g=2
h=6

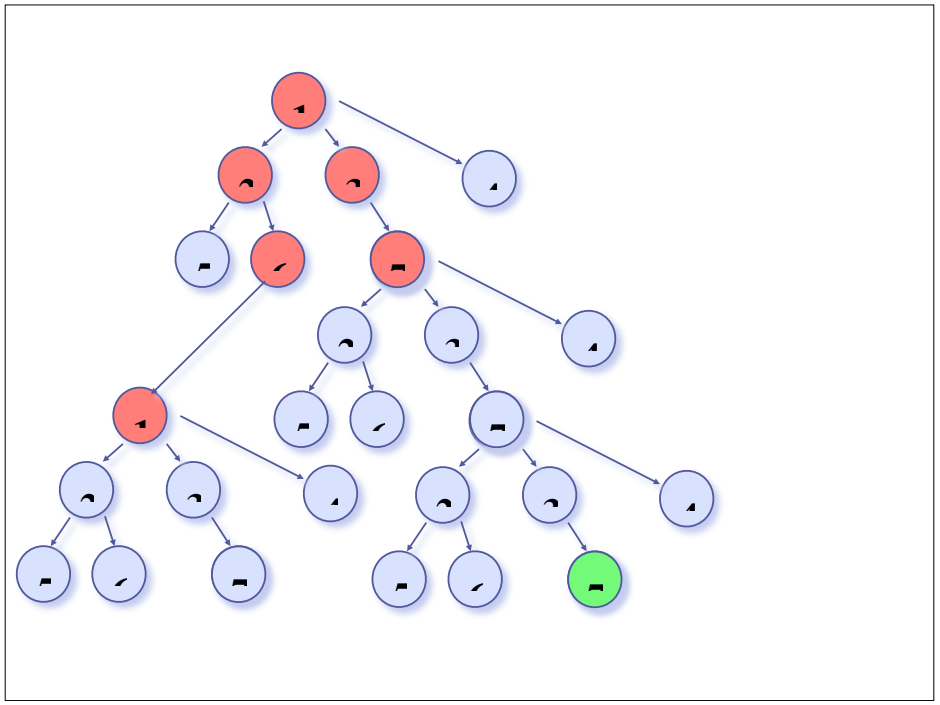


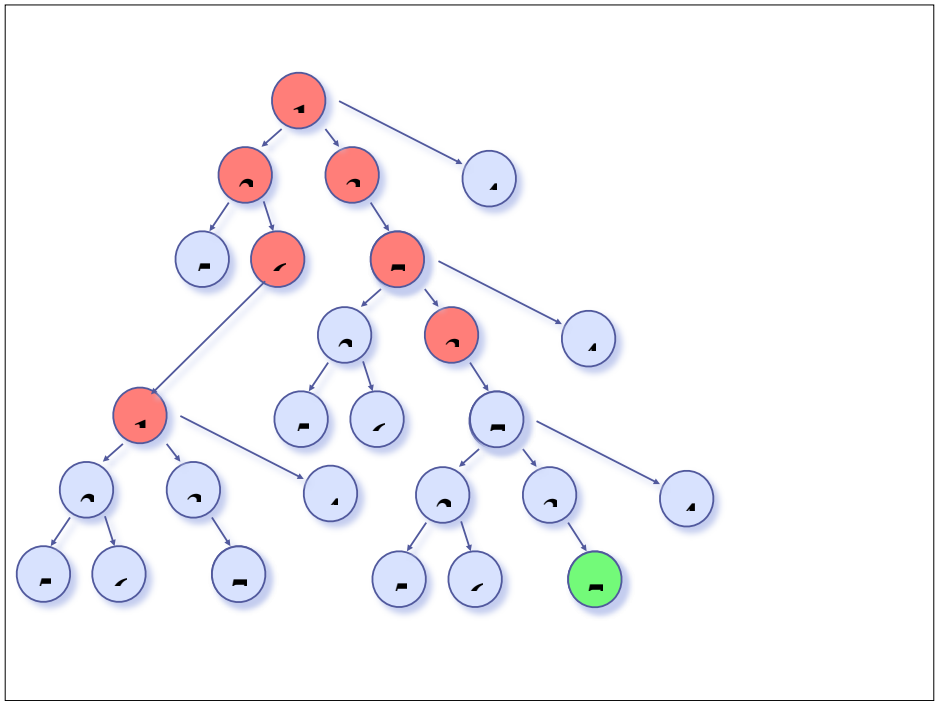


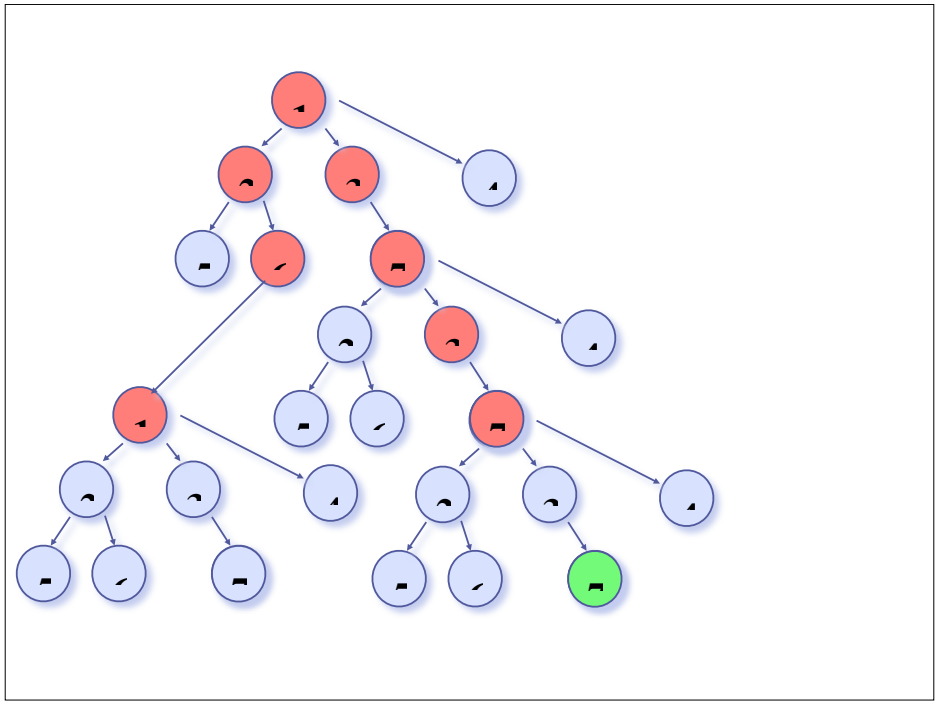


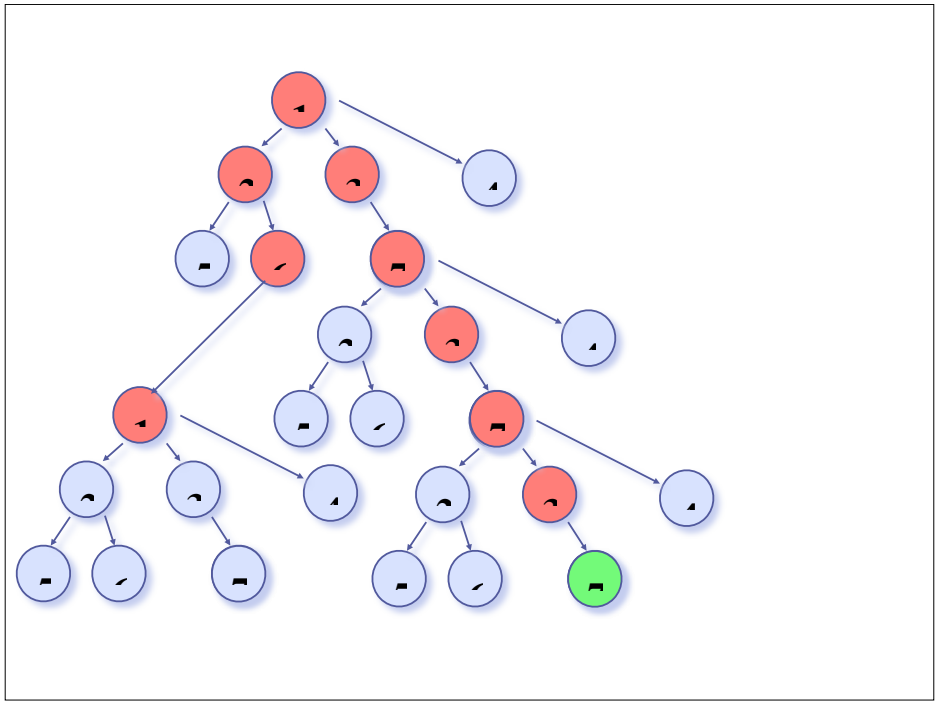


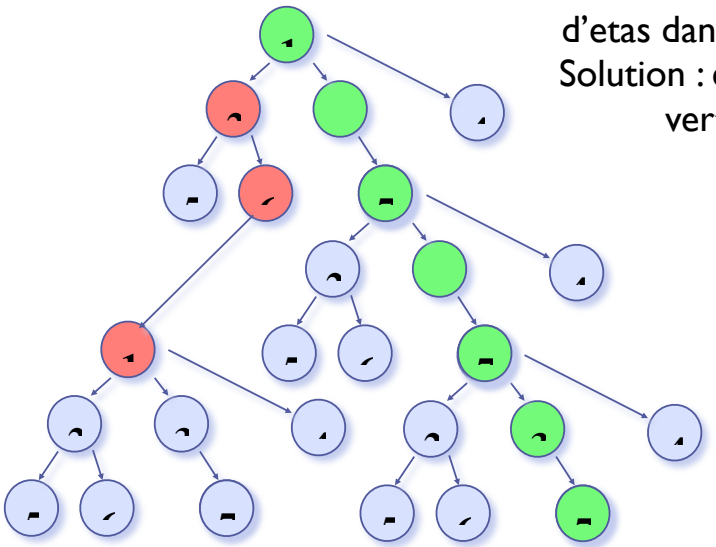












Etats visités : nb d'etas dans fermé
Solution : etats en vert

2	8	3
1	6	4
7		5

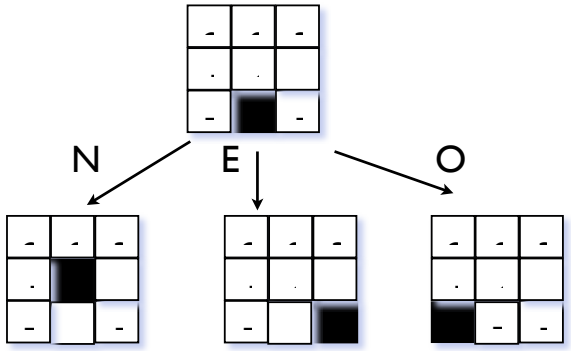
Etat de départ

objet etat_taquin
tableau de 9 valeurs
le 0 désigne le trou

1	2	3
8		4
7	6	5

Etat Solution

on cherche donc la
suite des coups a jouer
pour aller de l'état
de départ vers l'etat
solution



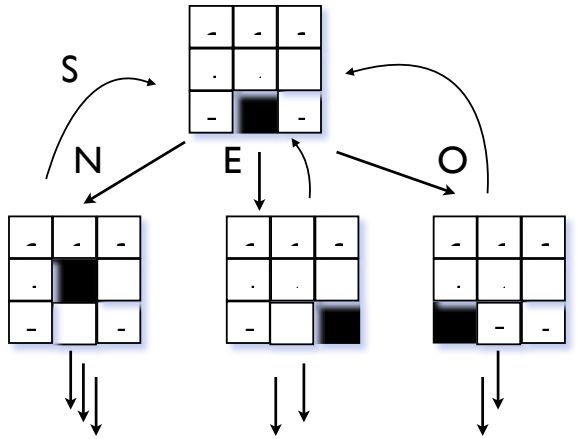
Etat de départ

generer_coup_suivant (etat_taquin etat_depart)
va nous retourner la liste des coups possible a partir de l'etat de départ
(attention au clonage d'un objet en java)

OUVERT et FERME vont donc être des liste d'éléments etat_taquin

chaque état contiendra un champ h, dédié au calcul de l'heuristique

ATTENTION, nous ne sommes plus dans un arbre, mais dans un graphe !



Etat de départ

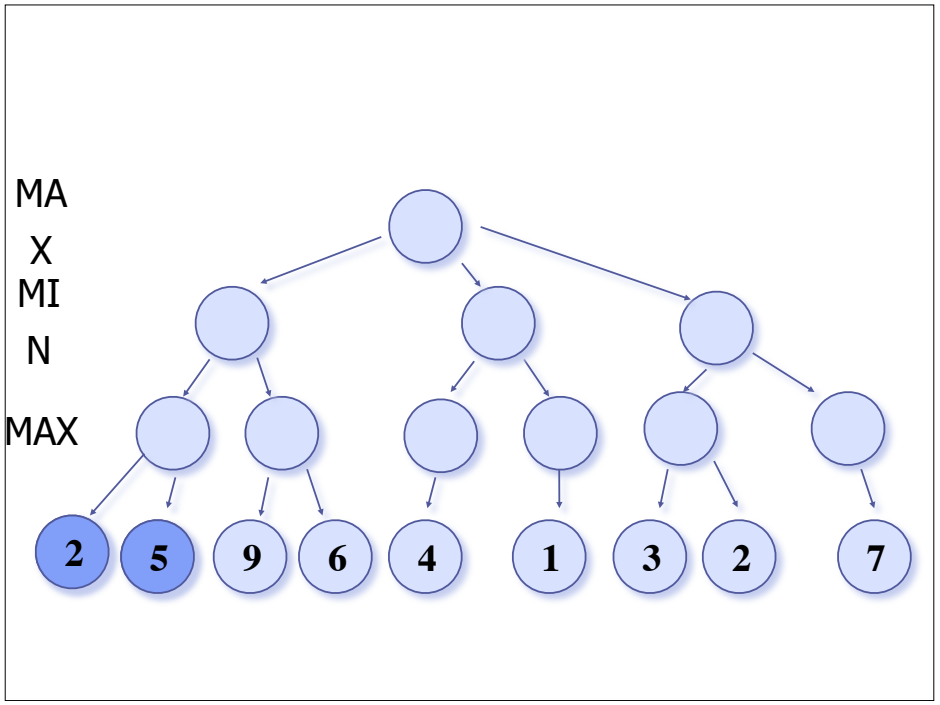
Application de A* aux graphes : cf algo du TP

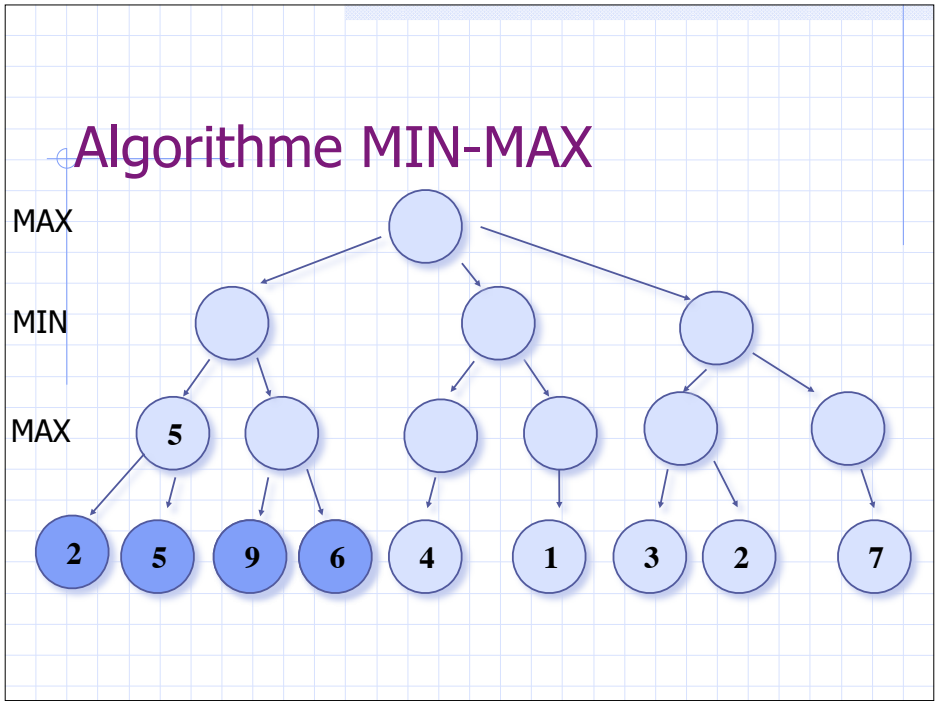
Deux cas nouveaux :

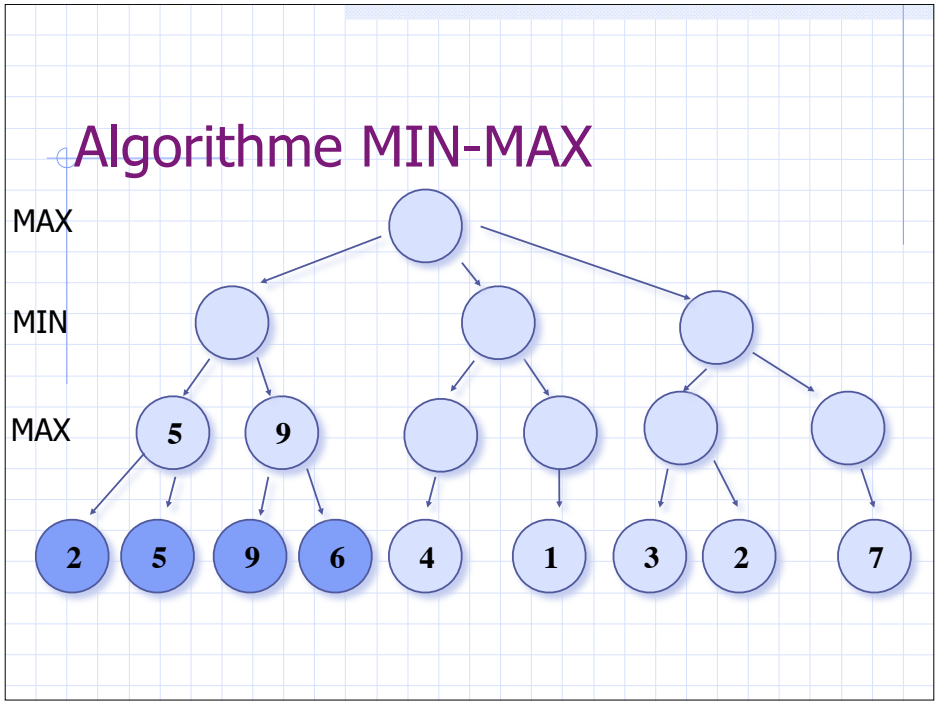
- Un nouvel état généré est déjà dans fermé :"on tourne en rond" ; on doit théoriquement réévaluer ce nouvel état (cas de jeux/problèmes ou l'on doit repasser
plusieurs fois par le même état pour trouver la solution)
Pour nous, la mesure g nous évitera de considérer le nouvel état

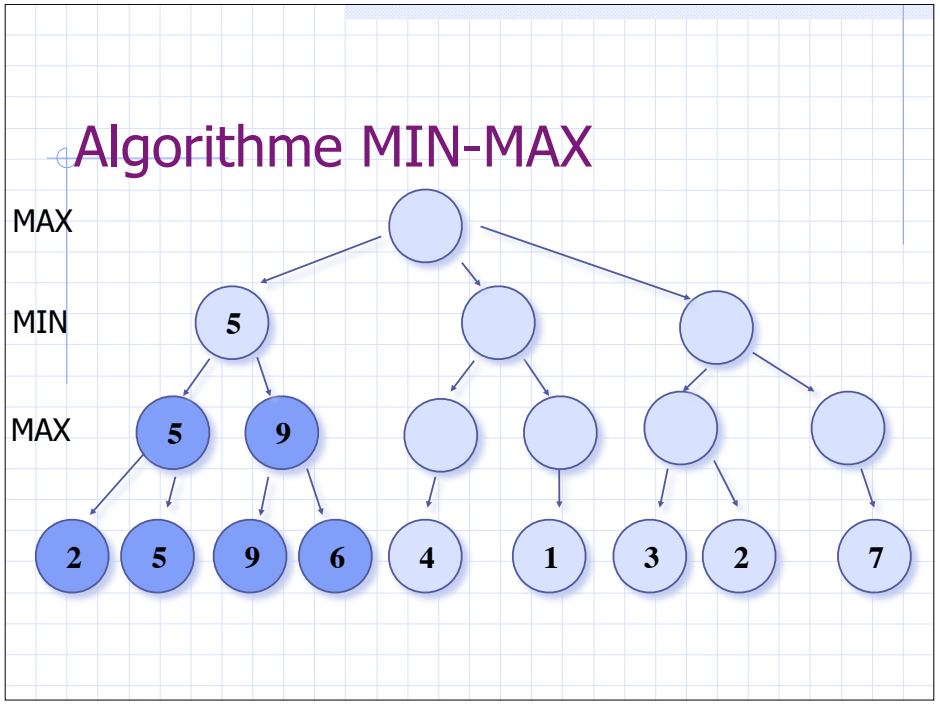
- un nouvel état généré est déja dans ouvert : seul le meilleur des deux reste dans ouvert.
Pour nous, la mesure g, nous evitera de considérer le nouvel état

- JE joue contre LUI
  - JE doit maximiser mes gains
  - LUI doit minimiser mes gains
- Représentation MIN-MAX

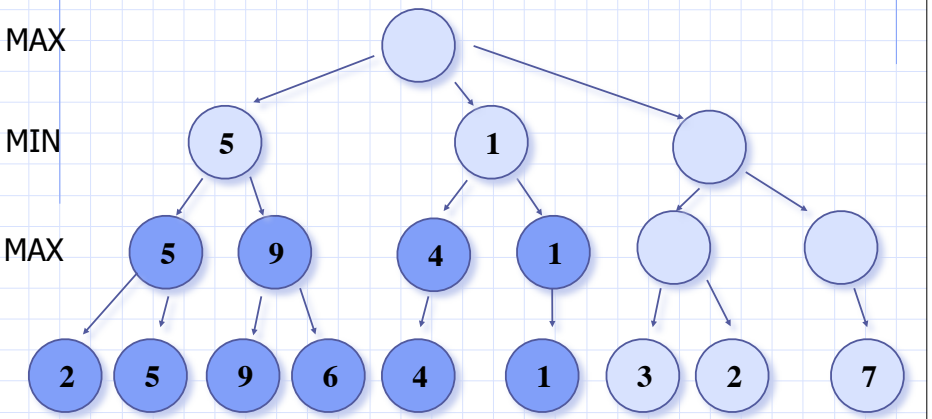

MA
X
MI
N
MAX
2
5
9
6
4
1
3
2
7


Algorithme MIN-MAX
MAX
MIN
MAX
5
2
5
9
6
4
1
3
2
7

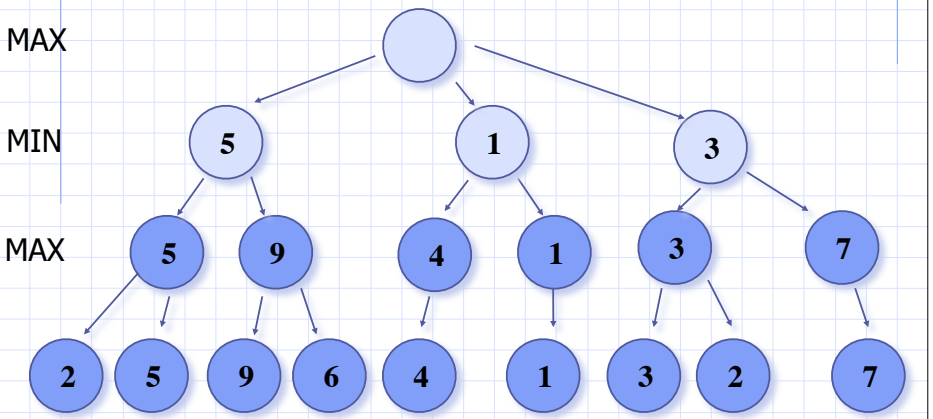

Algorithme MIN-MAX
MAX
MIN
MAX
5
9
2
5
9
6
4
1
3
2
7


Algorithme MIN-MAX
MAX
MIN
MAX
5
5
9
2
5
9
6
4
1
3
2
7

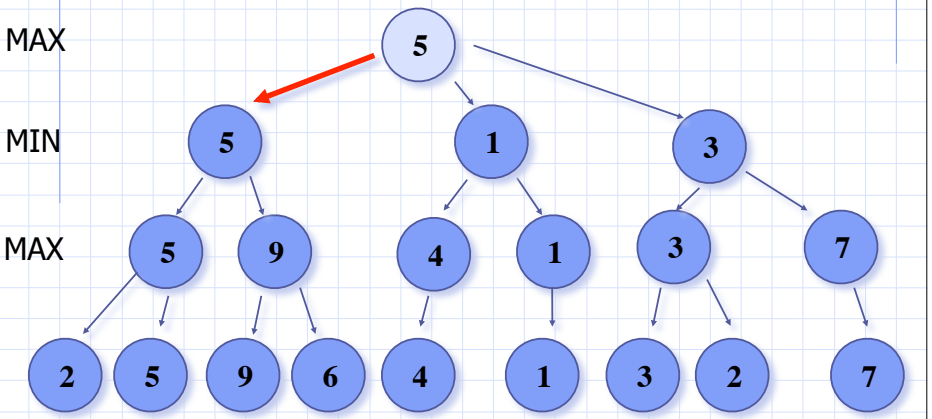
Algorithme MIN-MAX



Algorithme MIN-MAX



Algorithme MIN-MAX



min-max

PerAc

Traditional opposition : *classical IA* / *connectionism*

Achievements ?

PerAc

Traditional opposition : *classical IA* / *connectionism*

Achievement :

In 1997, Deep Blue super computer (IBM) beat world chess champion Gary Kasparov

... and Chess is *one* expression of intelligence (memory, symbol manipulation, adaptation...)

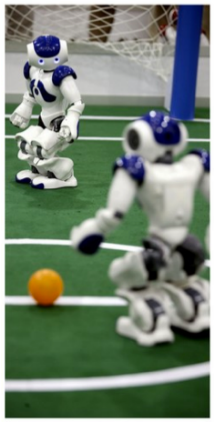
PerAc

Traditional opposition : *classical IA* / *connectionism*

...but

PerAc

Traditional opposition : *classical IA* / *connectionism*



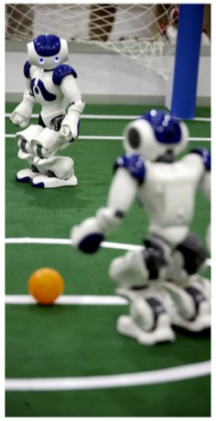
...but

Nao humanoid robot is still not able to compete at soccer with

a 3 years-old child...

PerAc

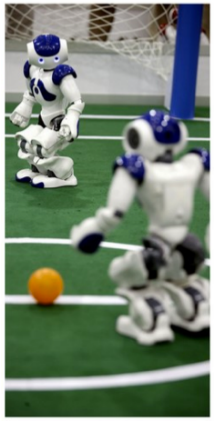
Traditional opposition : *classical IA* / *connectionism*



- we are able to beat the chess world champion....
- ...but not to play elementary ball game such as football
- Why ?

PerAc

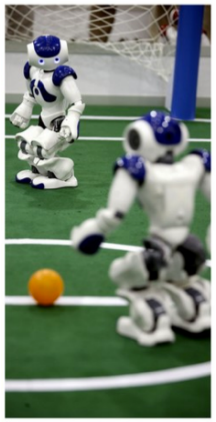
Traditional opposition : *classical IA* / *connectionism*



- perception
- action
- sensorimotor coordination
- memory
- fast decision making
- adaptation (“on line”)
- prediction / anticipation
- theory of mind ? strategy ?
- collective ?

PerAc

Traditional opposition : *classical IA* / *connectionism*

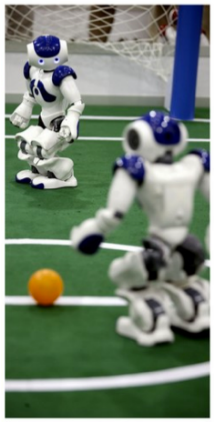


- perception
- action
- sensorimotor coordination
- memory
- fast decision making
- adaptation (“on line”)
- prediction / anticipation
- theory of mind ? strategy ?
- collective ?

All simultaneously...

PerAc

Traditional opposition : *classical IA* / *connectionism*



- perception
- action
- sensorimotor coordination
- memory
- fast decision making
- adaptation (“on line”)
- prediction / anticipation
- theory of mind ? strategy ?
- collective ?

All simultaneously

...with one brain

PerAc

Traditional opposition : *classical IA* / *connectionism*

- We are successful when the system (a computer) manipulates symbols...
- when the states are well *defined*
- *well identified*
- when the concepts are correctly *framed*
- when the world is easily *segmented*

Reasoning ?

Traditional opposition : *classical IA* / *connectionism*

- We are successful when the system (a computer) manipulates symbols...
- when the states are well *defined*
- *well identified*
- when the concepts are correctly *framed, formalised*
- when the world is easily *segmented*
- *Chess is perfect for that*

Reasoning ?

Traditional opposition : *classical IA* / *connectionism*

A. de Rengervé, J. Hirel, P Andry, M. Quoy, P. Gaussier -ETIS-Cergy
[BioRob2011]

- We fail when the system is immersed in the real world...
- always changing
- unpredictable
- things, peoples, objects, concepts are undefined
- noisy
- *A nightmare for Intelligent robotics*

Reasoning ?

Traditional opposition : *classical IA* / *connectionism*

- To bypass that these issues, we need :
  - *to let the system adapt*
  - *to let the system learn*
  - *build its own categories*
  - *to exploit the perception-action dynamics*

Reasoning ?

Traditional opposition : *classical IA* / *connectionism*

- To bypass that these issues, we need :
  - *to let the system adapt*
  - *to let the system learn*
  - *build its own categories*
  - *to exploit the perception-action dynamics*

The brain does it well : connectionism

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *object (legs, 4) and object(back,1) and object (seat,1)*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) and object(back,1) and object (seat,1)...*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) ~~and object(back,1)~~ and object (seat,1)...*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) ~~and object(back,1)~~ and object (seat,1)...*
  - *well...*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) ~~and object(back,1)~~ and object (seat,1)...*
  - *well...*
  - *..or (object, legs, 1) ?*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

(one of the world most known chair : Eames rocking chair)

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) ~~and object(back,1)~~ and object (seat,1)...*
  - *well...*
  - *..or (object, legs, 1) ?*
  - *..or 2 ?*

Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - *(object (legs, 4) or (object (legs,3)) ~~and object(back,1)~~ and object (seat,1)...*
  - *well...*
  - *..or (object, legs, 1) ?*
  - *..or 2 ? .....?*

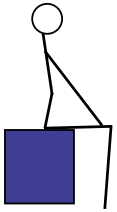
Reasoning...from perception ?

One more example : let's try to define an object : a chair

- Let's do it classical :
  - ~~*(object (legs, 4) or (object (legs,3)) and object(back,1) and object (seat,1)...*~~
  - ~~*well...*~~
  - ~~*..or (object, legs, 1) ?*~~
  - ~~*..or 2 ? .....?*~~
  - *....?*

Action matters !

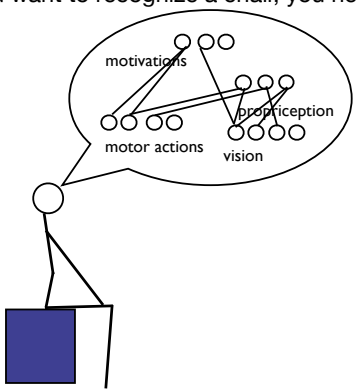
If you want to recognize a chair, you need to :



- **Build** the experience of seating
- to **categorize** it
  - *you need to have legs*
  - *you need to **need** to sit*

Action matters !

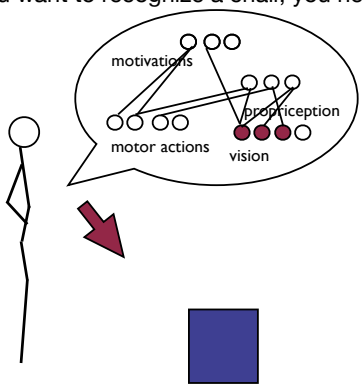
If you want to recognize a chair, you need to :



- **Build** the experience of seating
- to **categorize** it
  - *you need to have legs*
  - *you need to need to seat*
  - *merge the perceptions, the motor actions*
  - *to associate the objects vision with the whole*

Action matters...to produce perception

If you want to recognize a chair, you need to :



- **Build** the experience of seating
- to **categorize** it
  - *you need to have legs*
  - *you need to need to seat*
  - *merge the perceptions, the motor actions*
  - *to associate the objects vision with the whole*
  - *in order to generalize and recognize*

Action matters...to produce perception

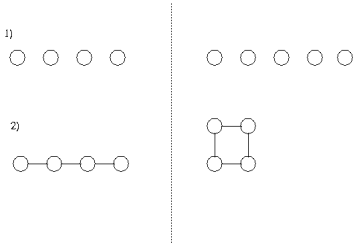
Connectionism is a part of this perception-action philosophy...

and neural networks is a tool to achieve such adaptive categorization

Notion de système

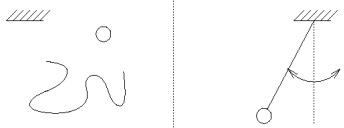
Etude des systèmes et de leurs propriétés:

Leur nombre (caractéristiques sommatrices)
Leurs espèces
Les relations entre éléments (caractéristiques constitutives)



Notion de système

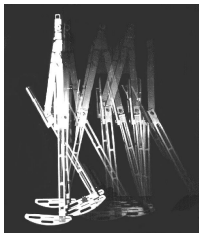
« Le tout est plus que la somme de ses parties »



Effet constitutif lié la dynamique
(propriété « émergente », Gestalt)

Notion de système : exemple avec la marche

Notion de système : exemple avec la marche



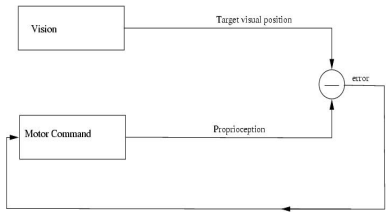
Notion de système : exemple avec la marche

Stabilisation d'un système

Nécessité d'introduire des boucles de régulation (asservissements)

Nécessité d'une copie efferente de l'action (proprioception ou retour interne) pour controler le système

principe homeostatique [ashby]:



Maitriser le controle moteur

Tracking/poursuite : rejoindre la cible visuelle

asservissement : pour atteindre la cible : réduire l'écart entre la valeur actuelle et la valeur de consigne.

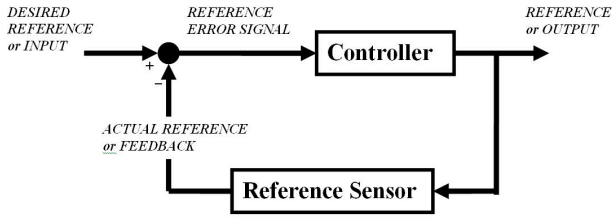
notion d'**erreur**.
boucle fermée.

Maitriser le controle moteur

Tracking/poursuite : rejoindre la cible visuelle

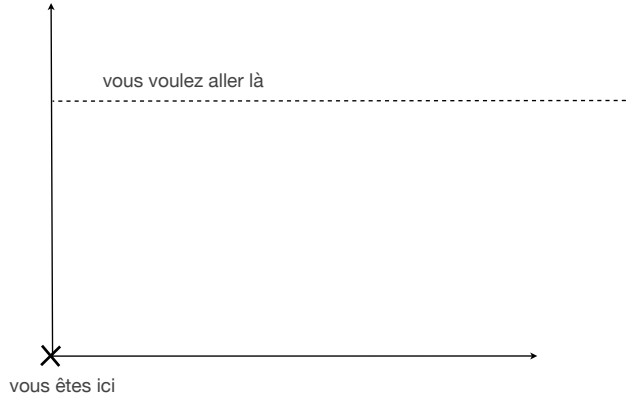
asservissement : pour atteindre la cible : réduire l'écart entre la valeur actuelle et la valeur de consigne.

notion d'**erreur**.
boucle fermée.



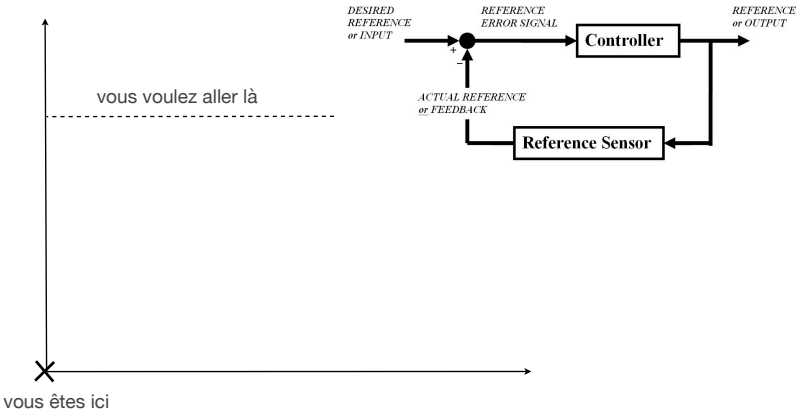
Maitriser le controle moteur

Tracking/poursuite : rejoindre la cible visuelle



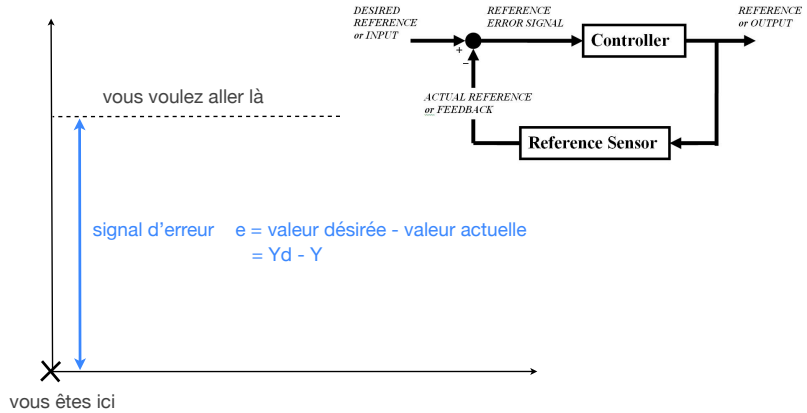
Maitriser le controle moteur

Tracking/poursuite : rejoindre la cible visuelle



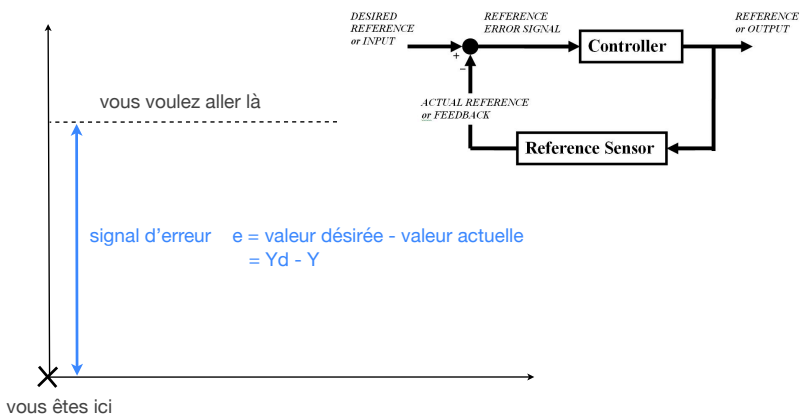
Maitriser le controle moteur

Tracking/poursuite : rejoindre la cible visuelle



Maitriser le controle moteur

1 ère solution : on envoie au contrôleur la totalité de l'erreur



Maitriser le contrôle moteur

1 ère solution : on envoie au contrôleur la totalité de l'erreur:

$$\text{Output} = e(t)$$

avec e(t), l'erreur calculée à l'instant t.

-simple.

- inconvénients ?

rien ne dit que l'erreur ne soit calculée dans l'espace moteur (il y a une transformation souvent complexe a faire). Donc rien ne dit que l'erreur ne corresponde à la commande a faire (une force, un temps, une pression, un courant, etc...).

même si la transformation d'espace est faite, donner une consigne égale à la totalité de l'erreur = grand mouvement = paramètres forts.
Ne marche que si la cible a rejoindre est immobile, où si l'effecteur est très rapide

inapproprié au contrôle en vitesse

Maitriser le contrôle moteur

Contrôle en position : la consigne est la position a atteindre.

Contrôle en vitesse : la consigne est une vitesse

Contrôle en force : la consigne est une force, une tension, une pression

Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec e(t), l'erreur calculée à l'instant t.

c une constante

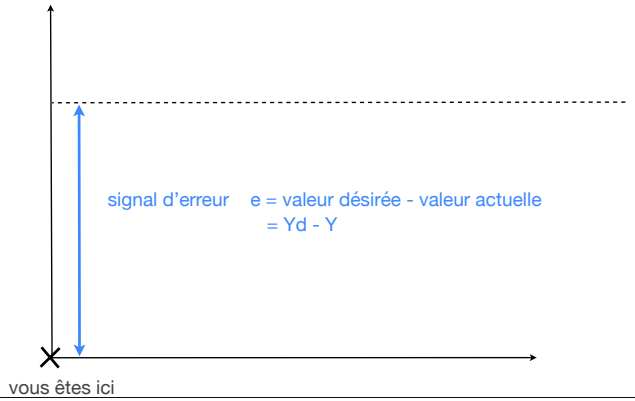
Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec e(t), l'erreur calculée à l'instant t.

c une constante



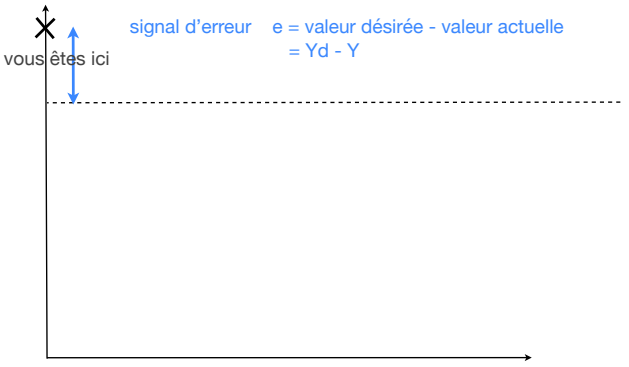
Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec e(t), l'erreur calculée à l'instant t.

c une constante



Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec e(t), l'erreur calculée à l'instant t.

c une constante

-simple.

- inconvénients ?

c étant une constante, si l'erreur est inférieure a c, on ne converge pas vers la solution

Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_P e(t)$$

avec e(t), l'erreur calculée à l'instant t.

kp un coefficient multiplicateur : facteur proportionnel à l'erreur

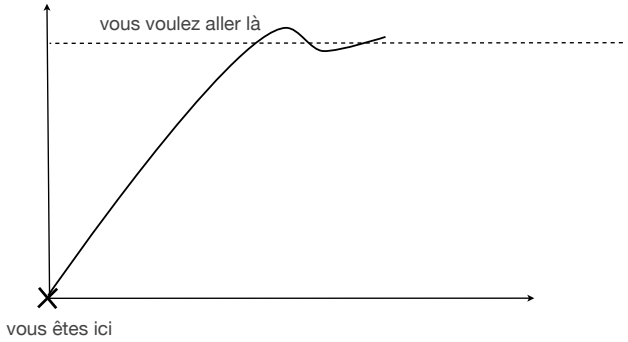
Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_P e(t)$$

avec e(t), l'erreur calculée à l'instant t.

kp un coefficient multiplicateur : facteur proportionnel à l'erreur



Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_P e(t)$$

avec e(t), l'erreur calculée à l'instant t.

kp un coefficient multiplicateur : facteur proportionnel à l'erreur

-simple.

- inconvénients ?

la convergence peut prendre du temps (amortissement vers la solution)

si le système rencontre une force opposée (frottement sec, frottement, objet), il restera empêché (la commande ne changera jamais et ne gagne pas en intensité).

Maitriser le contrôle moteur

si le système rencontre une force opposée (frottement sec, frottement, objet), il restera empêché (la commande ne changera jamais et ne gagne pas en intensité).

Nous allons prendre en compte l'accumulation de l'erreur : intégrale de l'erreur

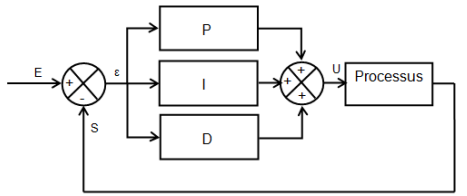
$$K_I \int e(t)\,dt$$

la convergence peut prendre du temps (amortissement vers la solution)

Nous allons prédire la direction du mouvement : dérivée de l'erreur

$$K_D \frac{d}{dt} e(t)$$

Maitriser le contrôle moteur

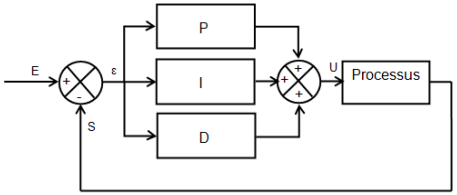


$$\text{Output} = K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

K_p: Proportional gain, a tuning parameter
K_i: Integral gain, a tuning parameter
K_d: Derivative gain, a tuning parameter
e: Error $= SP - PV$
t: Time or instantaneous time (the present)
τ: Variable of integration; takes on values from time 0 to the present t.

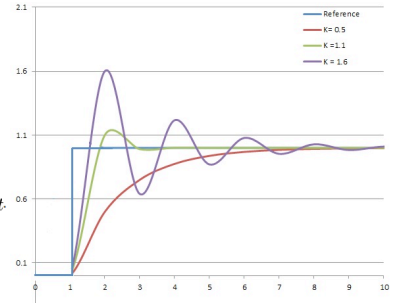
Maitriser le contrôle moteur



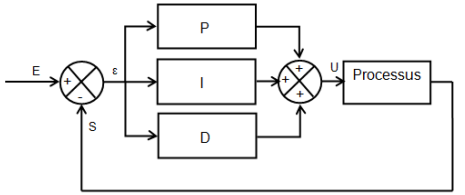
Output $= K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$

Where : e = Setpoint - Input

K_p: Proportional gain, a tuning parameter
K_i: Integral gain, a tuning parameter
K_d: Derivative gain, a tuning parameter
e: Error $= SP - PV$
t: Time or instantaneous time (the present)
τ: Variable of integration; takes on values from time 0 to the present t.



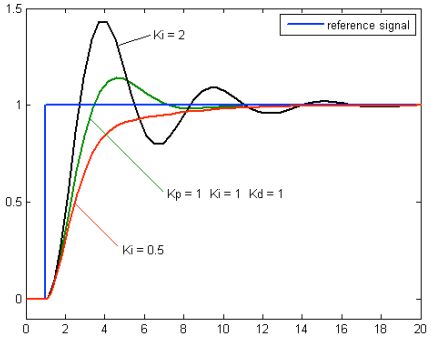
Maitriser le contrôle moteur



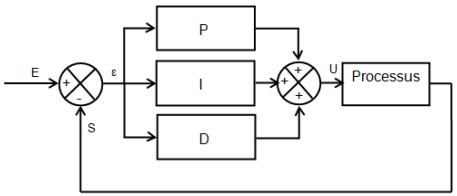
Output $= K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$

Where : e = Setpoint - Input

K_p: Proportional gain, a tuning parameter
K_i: Integral gain, a tuning parameter
K_d: Derivative gain, a tuning parameter
e: Error $= SP - PV$
t: Time or instantaneous time (the present)
τ: Variable of integration; takes on values from time 0 to the present t.



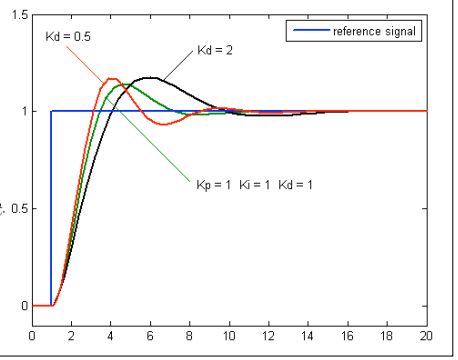
Maitriser le contrôle moteur



Output $= K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$

Where : e = Setpoint - Input

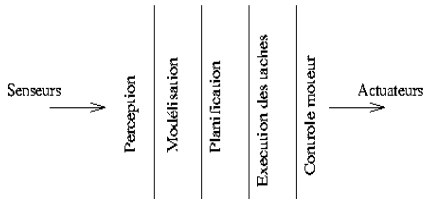
K_p: Proportional gain, a tuning parameter
K_i: Integral gain, a tuning parameter
K_d: Derivative gain, a tuning parameter
e: Error $= SP - PV$
t: Time or instantaneous time (the present)
τ: Variable of integration; takes on values from time 0 to the present t.



Approche fonctionnelle

Utilisation du formalisme logique (calcul des prédicats 1er ordre)

« High level reasoning » Robot Shakey (SRI, Menlo Park)



Approche fonctionnelle

Ex: Application au traitement d'images (séquence classique)

Formation d'image (capteur, numérisation)

Traitement d'image (filtrage, prétraitements, extration et fermeture des contours...)

Interprétation de l'image
(calcul param. Caractéristiques, reconnaissance et calcul des paramètres pertinents pour l'action visée)

Commande d'action
(action sur objet reconnu / IHM)

Approche fonctionnelle : limitations

Limitations :
En général, pb prise en compte des interactions avec l'env.
Peinture particulière pour les murs, les objets...

=> problèmes de perception

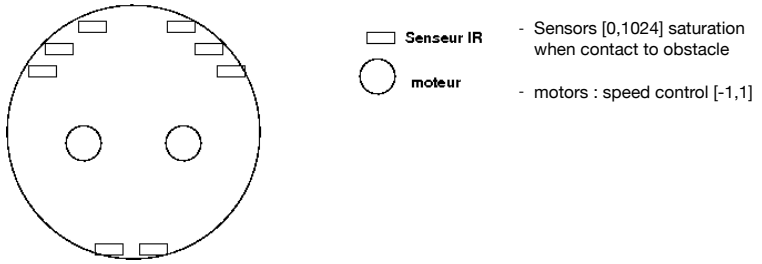
Approche comportementale, constructiviste

Approche ascendante/holistique

Tenir compte de l'observation « découverte » de propriétés émergentes non facilement prévisibles

example : vehicles

- Braitenberg's vehicles [Braitenberg80]
- You have to build a roomba robot that avoid obstacles.



example : vehicles

- Classical analysis:

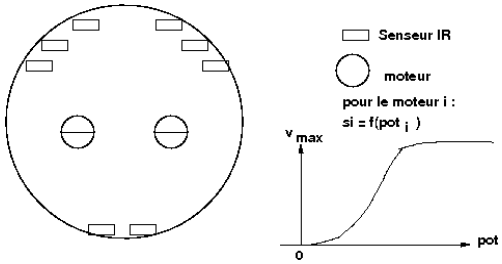
```
if obstacle_detected == left
      turn (angle_right)
if obstacle_detected == right
      turn (angle_left)
 if obstacle_detected == right&left
      ...
else ...
```

obstacle is an important symbol

Difficult to recognize, frame, (remember the chair...)

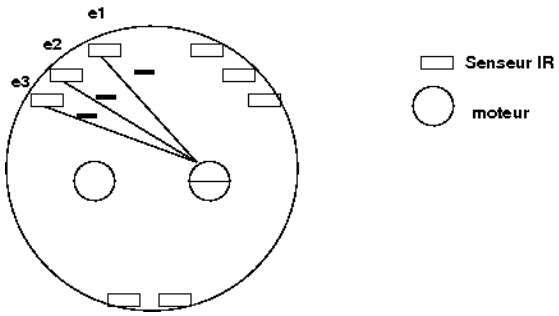
example : vehicles

- normalize sensor activities [0,1024] -> [0,1]



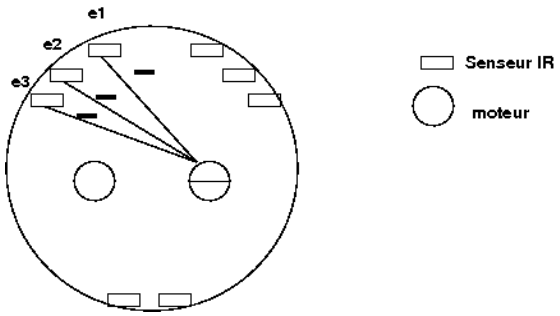
example : vehicles

- build connection (weight 1/3 to normalize) to the opposite motor.
- suppose the motor is activated by a neuron
- use identity transfer function



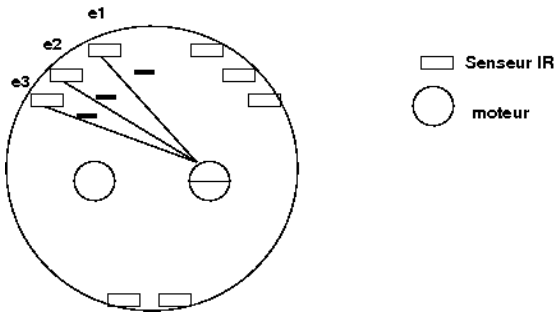
example : vehicles

- sensor saturation induce a diminution of the neuron potential : the opposite motor runs slower, the robot turn



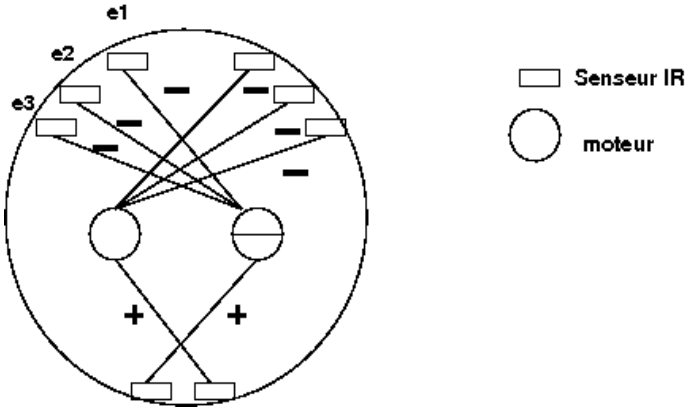
example : vehicles

- Code : only simple operations: pot(i) = e1*w1+e2*w2+e3*w3



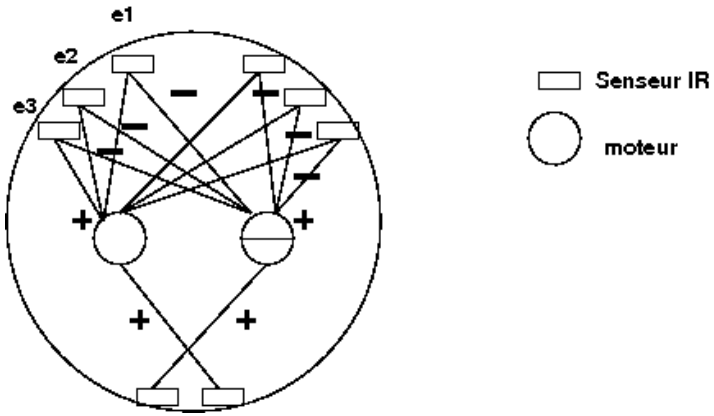
example : vehicles

- Complete architecture:



example : vehicles

- enhancement :

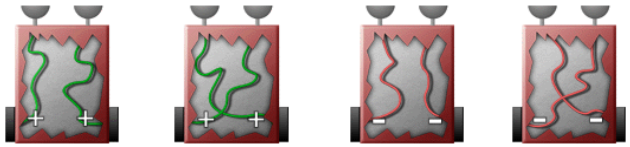


example : vehicles

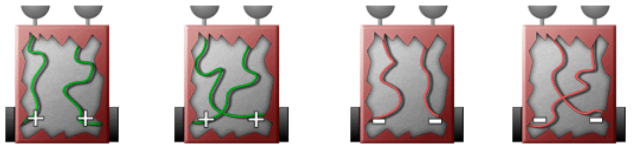
- only 2 neurons
- topology matters
- No “if”, just a vector product : light calculation.
- No notion of obstacle

- behaviors can be stacked with more sensors :
  - obstacle avoidance and phototaxis =
  - stacking circuits

example : vehicles



example : vehicles



fear angry love exploration

Connectionism is a part of this perception-action philosophy...

and neural networks is a tool to achieve such adaptive categorization

Overview

Introduction : from biology to the formal neuron

Part I : supervised learning

- perceptron
  - simple rule
  - Widrow Hoff rule
  - limitations
- associative memories
- multi-layer perceptron
- backpropagation

Part II : unsupervised learning

- brain mechanisms
  - competition and cooperation
  - WTA
- Self Organizing Maps
  - Kohonen maps
  - K-means (analogy)
- Let's put it all together
  - ART

From biology...

The brain is the main unit for information processing.

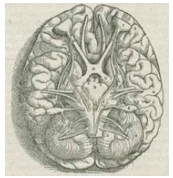
It is composed of 100 billions of nervous cells: the neurons

The neurons are connected in networks in order to :

- monitor
- regulate
- modulate

all the function of the organism.

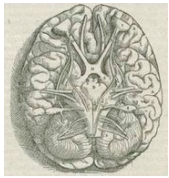
Moreover : the “organ” of intelligence



From biology...

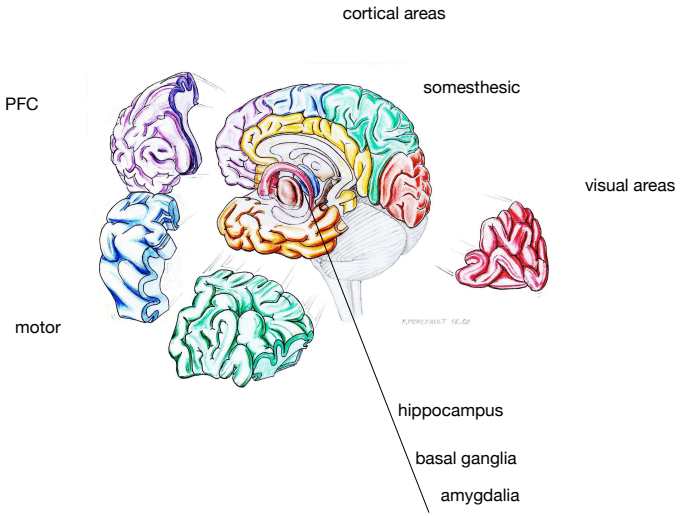
Cognition : all the mental processes involved in the scaffolding of *our* reality, et the basis of reasoning and all the high-level functions.

- perception
- memories
- building categories
- learning
- inhibition
- action selection
- *representations*



Information processing : input -> evaluation->decision->action

From biology...



From biology...

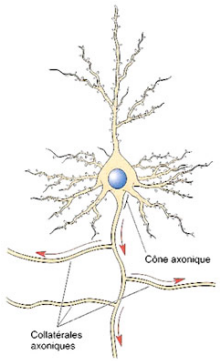
Neuron : the main unit for information processing

- Dendrit
- Axon
- Soma
- Synapses

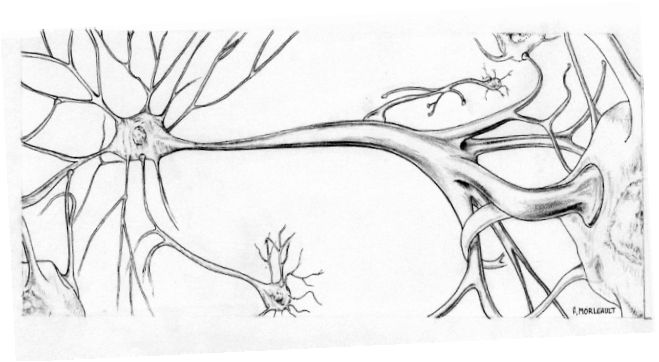
Electric information propagation

uni-directional

from one unit to the other



From biology...



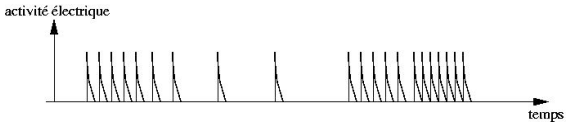
A single neuron can be 1m long

A single neuron can have up to 10 000 connections with other ones

(average of 1000 connections per neurons)

From biology...

But very slow

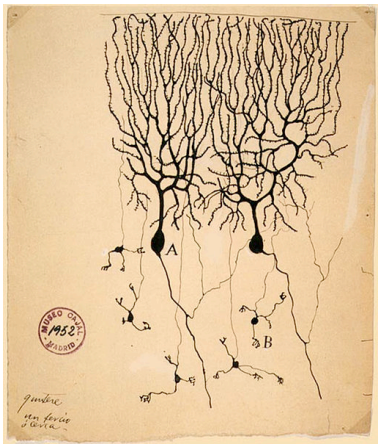


- If you measure the speed of information transmission, you get a rate of 300HZ
- 300HZ = frequency of the action potential : Spike train
- very slow, when compared to modern computer buses (circa 1GHZ)

From biology...

But...

- Multiple connections
- massive parallelism : continuous flow of information tat can be processed by multiples areas at the same time (asynchronous calculation - no main clock-)
- Adaptation : the synapse can adapt to modify the information intensity transmitted by a neuron to the other: learning



From biology...

comparison brain/ modern computer

Von Neumann Architecture	Brain
Computation and memory: separated and centralized	Computation and memory integrated and distributed
program = sequence of instructions	calculus = multiple constraints satisfaction
execution of one process at a time	permanent combination of multiples different information sources
1 to 8 very fast processors t	hundreds of billions of slow connected units

...to information processing...

...to information processing...

At the basis any behavior, is information processing

2 different kinds :

- Chemical, long term (LT) “message sending” **: hormones.** chemical and not restricted to a given receptor, diffusion in all the body, transported by the blood. Slow process.

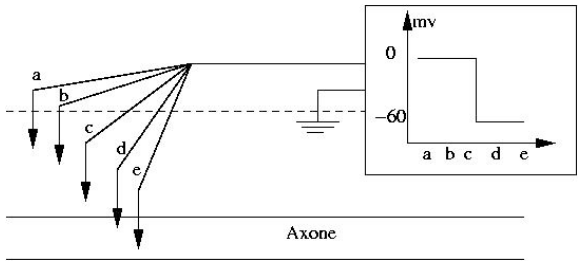
- Electro-chemical, short term, **potential trains** sent by neurons. “Fast”

  - *One neuron produces electrical potential changes, and the changes (the potential variation) are transmitted all along the membranes : emission of electrical signals*

  - *The communication int the inter-neural space (between the synapse and the dendrite) is made by production of chemical substances, the neurotransmitters*

...to information processing...

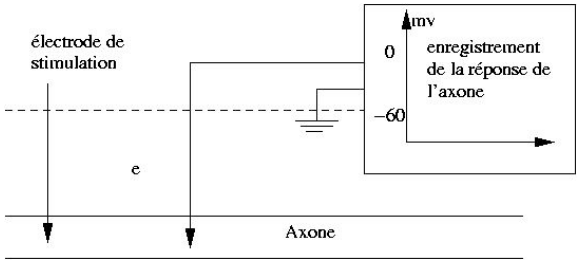
- suppose you have a multimeter to measure the value of the electrical potential inside and outside the axon membrane



- default resting potential (of the neuron): -66 mv : the inside of the axone is negatively charged (difference between a,b,c, and d, e)

...to information processing...

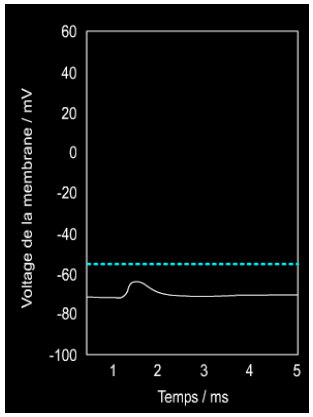
- Suppose you plug an electrode and send a given voltage in the membrane of the axone...



- ...and you record the axon’s membrane reaction at a given point

...to information processing...

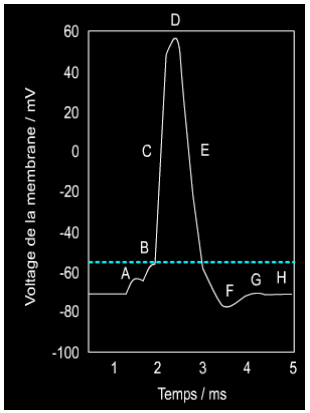
- Stimuli under 10 or 15mv : no significative response from the membrane



...to information processing...

- Depolarization > 10 or 15 mv : strong reaction called action potential .

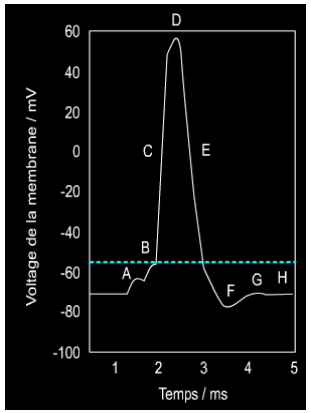
- spike reaching +60mv (C-D)
- overshoot
- whatever the stimulation is (>15mn), the spike is the same
- stereotyped
- followed by a short undershoot (F-G) during no new stimulation is possible: explain the 300Hz limit



...to information processing...

- Depolarization > 10 or 15 mv : strong reaction called action potential .

- The spike will propagate all along the axon with conservation of shape and amplitude
- traveling electrical wave
- 30 m/s



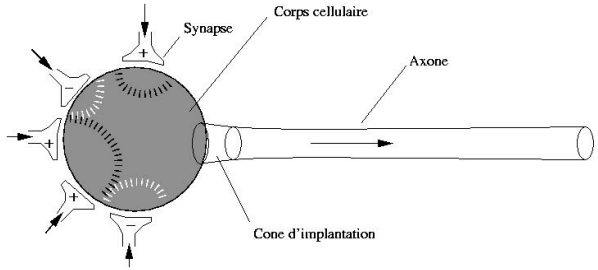
...to information processing...

- If the intensity of the trigger has no effect on the shape of the peak, it will nevertheless influence the amount of peaks generated.
- the stronger the stimulus is, the greater is the number of peaks (frequency max : 300hz)
- The axon is providing a frequency code for the intensity of the stimulus
- potential trains
- naturally, the trigger of the pikes potential will depend on the activity of the neuron's nucleus : the soma

...to information processing...

- The neuron's nucleus will act as an *adder*, summing the incoming potentials (arriving potentials from the dendrites).
- it is called the soma (soma = sum in latin)
- temporal en spatial sum according to :
  - the synapses positions on the dendrite
  - the frequency of the peaks
- The soma will trigger the axon if a given threshold is overshoot
- non-linearity : thresholds + saturation

...to information processing...



...to a simple formal neuron

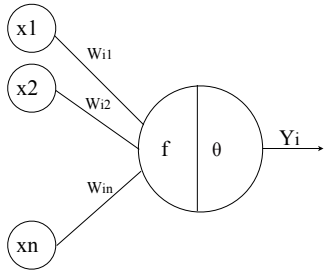
Formal neuron i

- main processing unit
- combined in networks
- calculate the incoming potentials, the Xi [-1,1]:
- summation in an internal potential Poti [-1,1]
- Threshold θ [-1,1]
- deliver one output Yi [-1,1]
- For our introduction we won't use the frequency coding : just a value expressing the mean number of emitted potential
- McCulloc & Pitts model [MacCullocs & Pitts48]

See the works of Thorpe & Al. for models of neurons coding the frequency of the action potential : spiking neurons

...to a simple formal neuron

Formal neuron i



...to a simple formal neuron

Formal neuron i

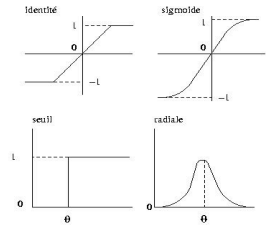
- calculate the incoming potentials, the Xi [-1,1]:
- summation in an internal potential Poti [-1,1]
- Threshold [-1,1]
- deliver one output Yi [-1,1]
- we have seen that the information transmission between two neurons is made by emission of neurotransmitters.
- Coefficient W [-1,1] of the incoming potential

...to a simple formal neuron

Formal neuron i

- Internal potential of neuron i : poti = $\sum$ wij.xj
- activation of output : Yi = f(poti)
- with f, the transfer function according to the model

For example :

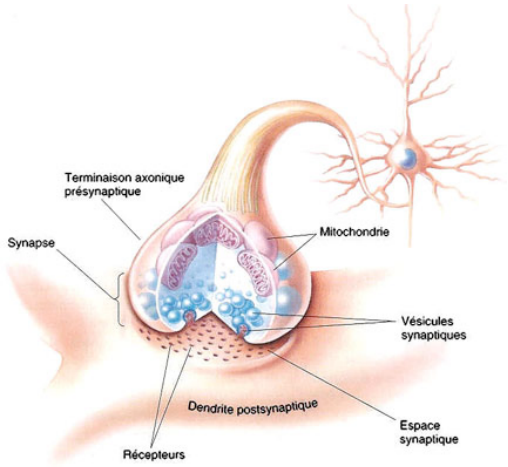


identity function

$$f(x) = \begin{cases} 0 \text{ if } x \leq \alpha \\ 1 \text{ if } x > \beta \\ x \text{ if } x \in [\alpha,\beta] \end{cases}$$

with $\theta = \beta - \alpha$

Learning



Learning

- nervous circuits : also crucial for memory (no dedicated centralised structure).
- Observation : an informal experience is correlated to measurable neuro-chemical and neuro-anatomical modifications in the brain
- Physiological modification of synapses:
  - Pré-synaptic : increase release of neurotransmitters
  - Post-synaptic : increase sensitiveness of the receptive membrane
- Structural modifications :
  - the frequent "use" of a circuit induce an increase of the synaptic contacts

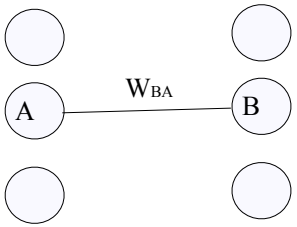
Learning

Hebb Rule [Hebb 49] :

"*when cell A excites by its axon cell B and, in a repeated and persistent manner, it triggers impulsion of B, a process of metabolic change happens in one or two of both cells, driving to an significant increase of the efficiency of A to generate an impulsion in B, among the other cells*"

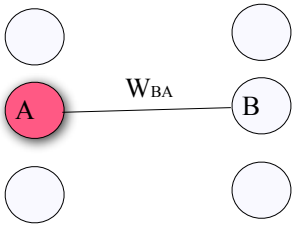
Learning

Hebb Rule [Hebb 49] :



Learning

Hebb Rule [Hebb 49] :

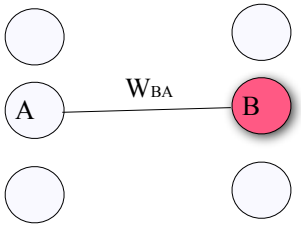


The sole A activity is not enough to induce B activation

$A^*W_{BA} < \theta_B$

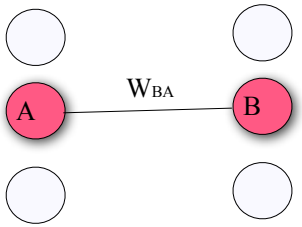
Learning

Hebb Rule [Hebb 49] :



Learning

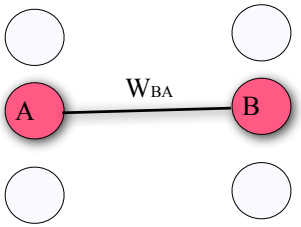
Hebb Rule [Hebb 49] :



For some reason... coactivation of A and B...

Learning

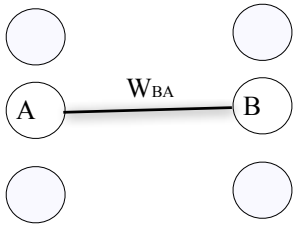
Hebb Rule [Hebb 49] :



For some reason... coactivation of A and B...repeatedly increase of the connection W_{BA}

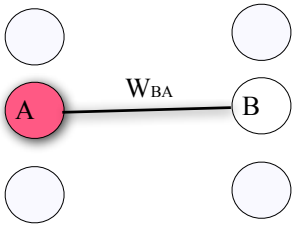
Learning

Hebb Rule [Hebb 49] :



Learning

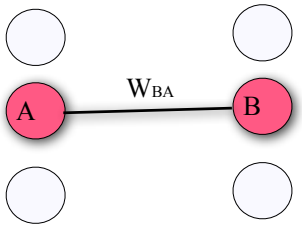
Hebb Rule [Hebb 49] :



The sole A activity is now enough ...

Learning

Hebb Rule [Hebb 49] :



The sole A activity is not enough to overshoot B threashold

$A*W_{BA} > \theta_B$

Learning

- Hebb rule

$$Wij\ (t+1) = wij\ (t) + eps.Yj.Yi$$

i.e :

$$\Delta Wij = eps.Yj.Yi$$

with :

eps : learning speed

Supervised learning

- goal :
- make the network learn to categorize inputs
  - pattern recognition
  - class separation
- method :
  - train the network from different inputs
  - test the network generalization

Supervised learning

- goal :
- make the network learn to categorize inputs
  - pattern recognition
  - class separation
- method :
  - train the network from different inputs
  - test the network generalization

Supervised learning

The network will learn

- structural changes
- modification of Wij in order to obtain the correct output
- the expected result is a correct categorization
- learning is iterative: not too fast, not too slow
- learning rate epsilon
- test the generalization

Supervised learning

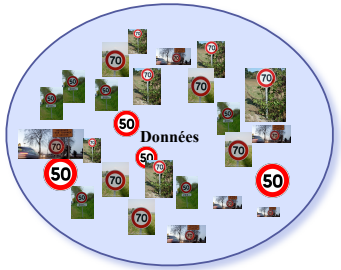
How to guide the learning ?

- fundamental notion of **error**
- supervised : we expect a given answer
- at each time step, we calculate the error
- error : (desired ouput - output)
- while there is an error : we change the weights

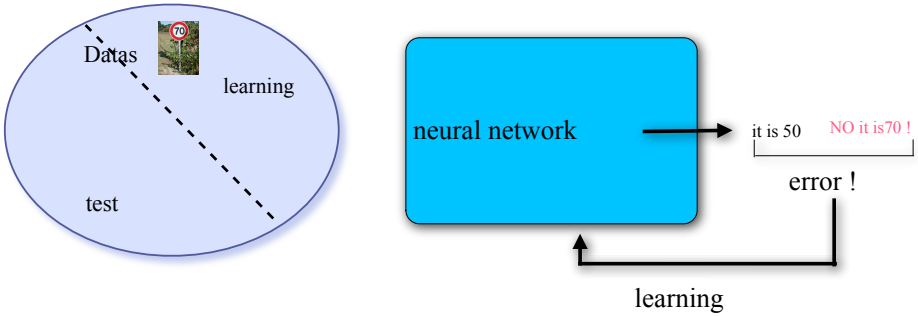
Supervised learning

Example

Supervised learning



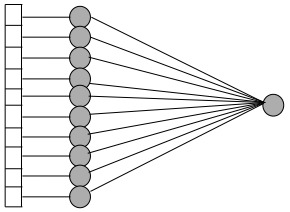
Supervised learning



Perceptron

Architecture

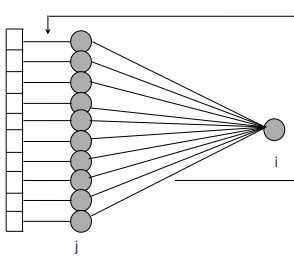
- Input, the retina : raw numerical information
- A first layer of neurons : one-one connections with the retina (normalization only)
- A last layer, called decision layer : the output of the system.



Perceptron

Architecture

- Input, the retina : raw numerical information
- A first layer of neurons : one-one connections with the retina (normalization only)
- A last layer, called decision layer : the output of the system.

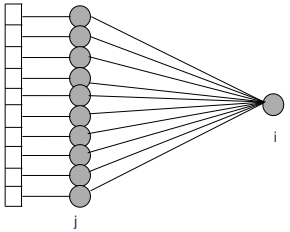


- non-modifiable weight. direct transmission, W=1/(raw-max)
- modifiable weights. learning !
- Wij= random between [-1/n,1/n]

Perceptron

Learning : the simple rule

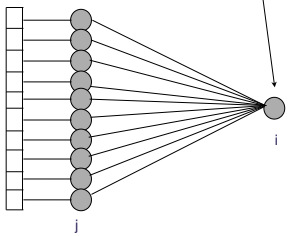
- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$



Perceptron

Learning : the simple rule

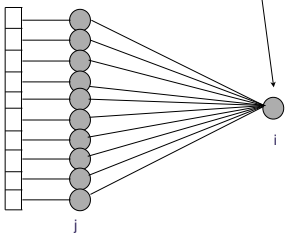
- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$



Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$

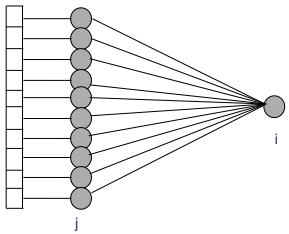


Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - Y_i).X_j$

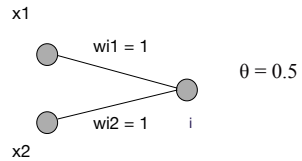
error



Perceptron

Graphical interpretation

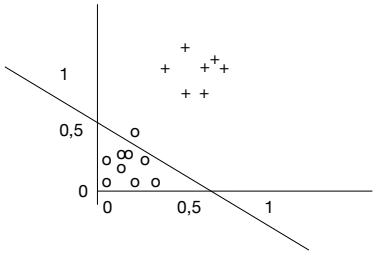
- suppose the following simple perceptron :



- Draw the plane (here, the line) that separate the space in two categories.
- what happend if wi1 is equal to 2 ?
- what happend if θ is equal to 1 ?

Perceptron

Graphical interpretation

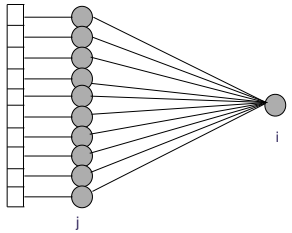


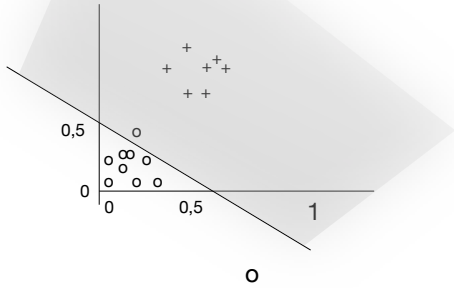
- input space separation
- categorization -> two areas, two categories in the input space
- change Wij -> change the frontier -> change the output -> change the error

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$



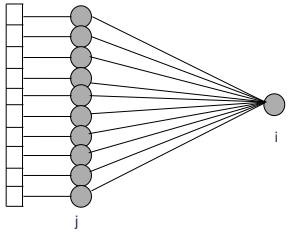


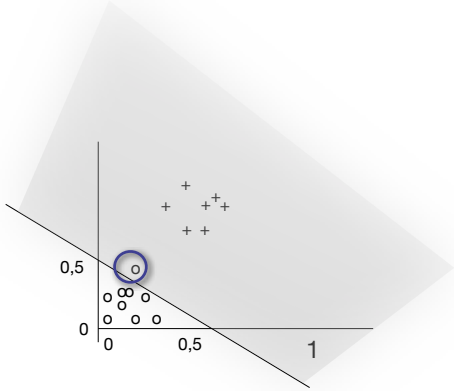
o

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$





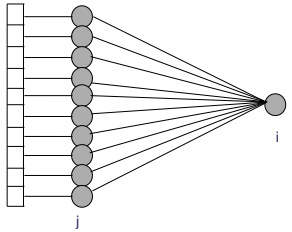
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

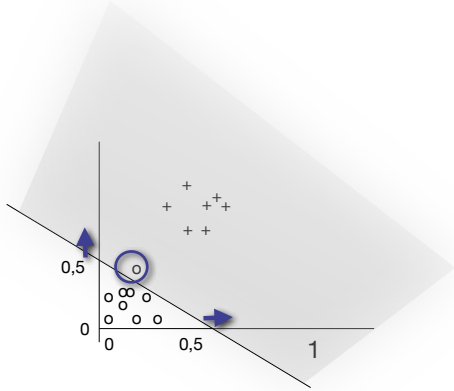
o

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$





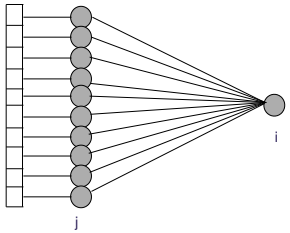
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

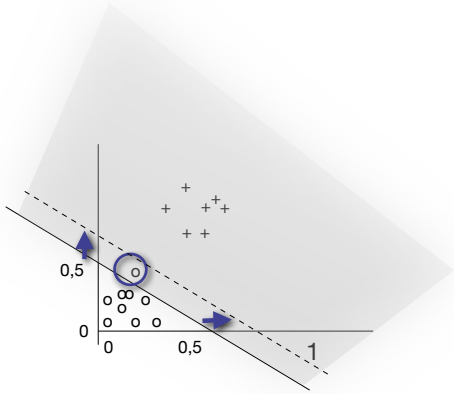
o

Perceptron

Learning : the simple rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-Y_i).X_j$





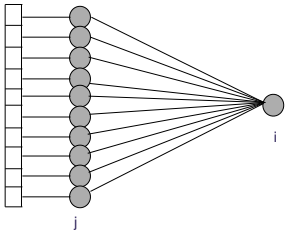
- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

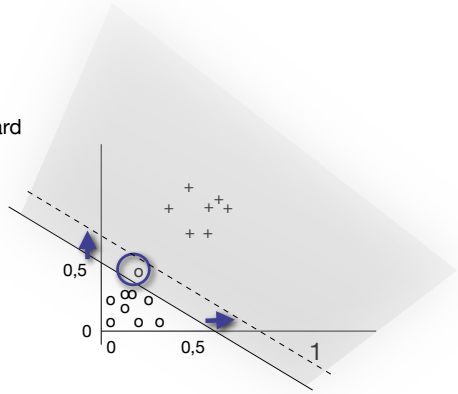
o

Perceptron

Learning : the simple rule

- Step by step (eps) progress toward space separation
- not accurate
- learning stops once no error





- Error !
- Yi is 1 and Yd is 0
- error is -1, both weights are lowered by eps

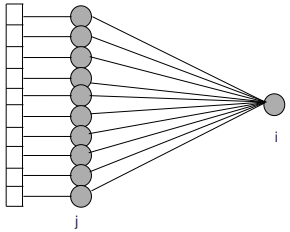
o

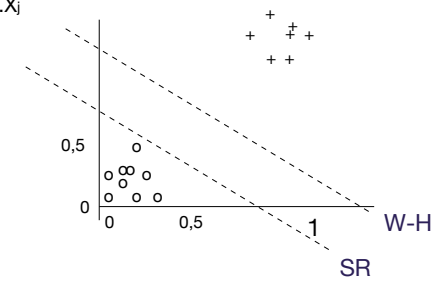
Perceptron

Learning : Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-\sum(wij.Xj)).X_j$

error





Mathematical explanation

Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d-\sum(wij.Xj)).X_j$

error

- Principle : we want our network to allways provide the right answer

Mathematical explanation

Widrow-Hoff rule

- $w_{ij}(t+1) = w_{ij}(t) + eps.(Y_d - \sum(wij.Xj)).X_j$

error

- Principle : we want our network to allways provide the right answer
- means : always provide the lowest error as possible
  - toward 0
  - For all inputs examples

Mathematical explanation

Linear regression and least mean square rule

- we introduce the global quadratic error : sum of the errors of all examples : $E^2 = \sum e^2$
- with e : perceptron error with a single example : $e = (Yd - \sum(wij.Xj))$
-

Mathematical explanation

Linear regression and least mean square rule

- we want to minimize E but :
  - At each time we only have acces to e
  - we can only change W
- So, we have to express E according to e and then according to W

- if $E = \sum (Yd - \sum(wij.Xj))^2$ then we will minimise E by descending the gradient of e, at each time step, changing W
- with $e = u(w)^2$ and $e' = 2u' . u(W)$
- $e' = 2 \ (\sum(wij.Xj)) .(Yd - \sum(wij.Xj)) = 2.Xi. (Yd - \sum(wij.Xj))$

Mathematical explanation

Linear regression and least mean square rule

- introducing eps being a learning speed parameter, we get :

$$\Delta wij = eps .Xi. (Yd - \sum(wij.Xj))$$

- that is to say :

$$wij (t+1) = wij(t) + eps .Xi. (Yd - \sum(wij.Xj))$$

- that is sufficient to ensure a step by step minimization of E
- = descent of the error's gradient

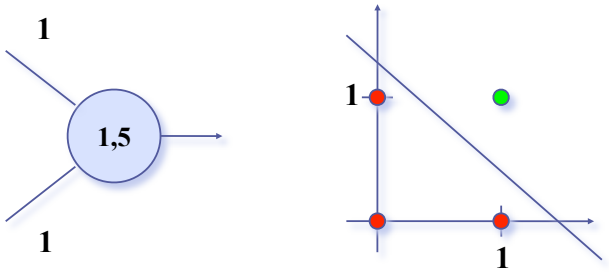
multi-layer perceptron

The network will learn

- structural changes
- modification of Wij in order to obtain the correct output
- the expected result is a correct categorization
- learning is iterative: not too fast, not too slow
- learning rate epsilon
- test the generalization

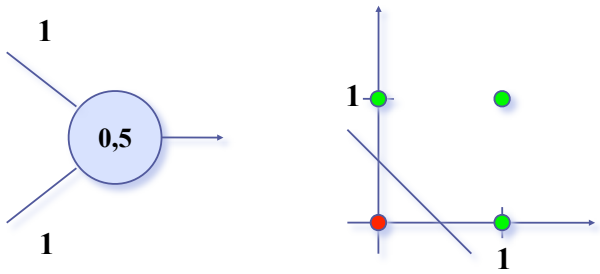
Perceptron

- logical AND



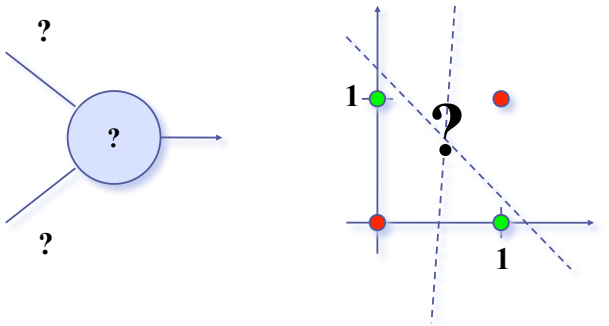
Perceptron

- logical OR



Perceptron

- XOR ?

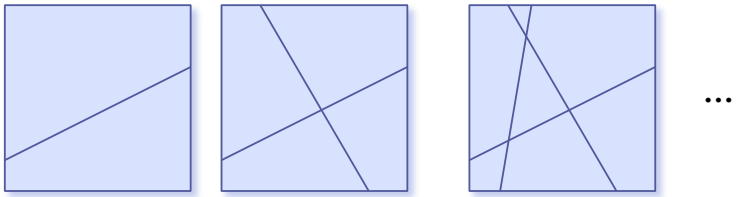


Perceptron

- limited to linear separation
- **what to do ?**
  - increase the number of layers and combine the outputs

Perceptron

- limited to linear separation
- **what to ?**
  - increase the number of layers and combine the outputs



1 layer **2 layers** **3 layers** ...

multi-layer perceptron

The backpropgation algorithm looks for the minimum of the error function in weight space using the method of gradient descent

- the combination of weights that minimize this error = solution
- computation of the gradient of error at each time step
- switch to continuous activation function = continuous error function

$s_c : \mathbb{R} \to (0,1)$ defined by the expression

$$s_c(x) = \frac{1}{1+e^{-cx}}.$$

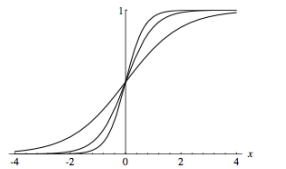
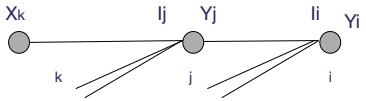


Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

back - propagation

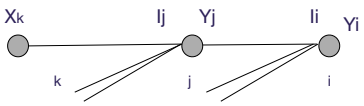


rule : Δwij = eps.di.Xj

with : last layer : di = Yd - Yi

hidden layer : dk = sum (di.wij) . f'(pj)

back - propagation

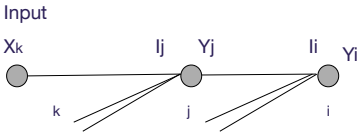


rule : Δwij = eps.di.yj

with : last layer : di = Yd - Yi Widrow-Hoff

hidden layer : dk = sum (di.wij) . f'(pj) with pj = ∑Yj Wij

back - propagation

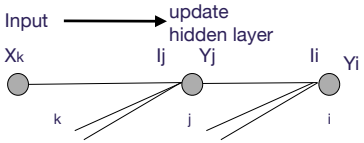


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

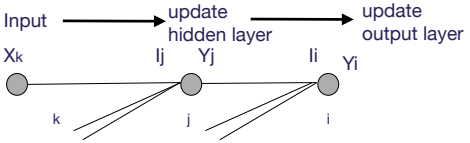


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

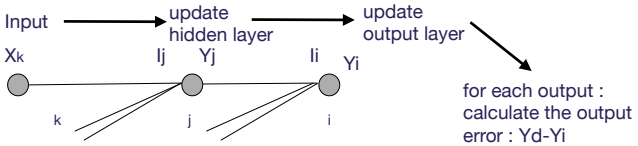


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

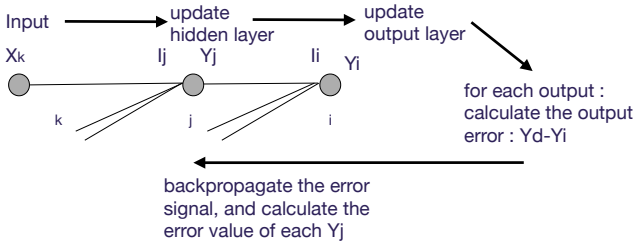


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

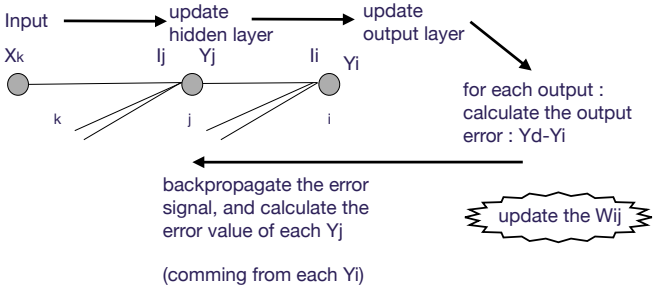


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

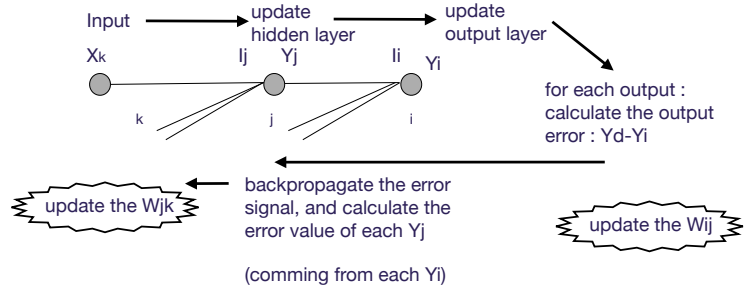


It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation



It mean that we will have to propagate the error backward to calculate the Wij and Wjk.

Activiy is computed forward

learning is computed backward

back - propagation

In the same maner as the WH rule, the BP allow to a gradient descent of the error.

Global error : $E=\sum_{p} e^{2}$

local error : e = Yd-Yi

Yi = f(pi) and pi = ∑Wij.Yj (pi is the potential of neuron i)

Once again, we can express the derivative of E, according to W:

$$\frac{\partial E}{\partial Wij} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial Yi} \cdot \frac{\partial Yi}{\partial pi} \cdot \frac{\partial pi}{\partial Wij}$$

back - propagation

In the same maner as the WH rule, the BP allow to a gradient descent of the error.

Global error : $E=\sum_{p} e^{2}$

local error : e = Yd-Yi

Yi = f(pi) and pi = ∑Wij.Yj (pi is the potential of neuron i)

Once again, we can express the derivative of E, according to W:

ei -1 f'(pi) yj

$$\frac{\partial E}{\partial Wij} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial Yi} \cdot \frac{\partial Yi}{\partial pi} \cdot \frac{\partial pi}{\partial Wij}$$

back - propagation

In the same maner as the WH rule, the BP allow to a gradient descent of the error.

Global error : $E=\sum_{p} e^{2}$

local error : e = Yd-Yi

Yi = f(pi) and pi = ∑Wij.Yj (pi is the potential of neuron i)

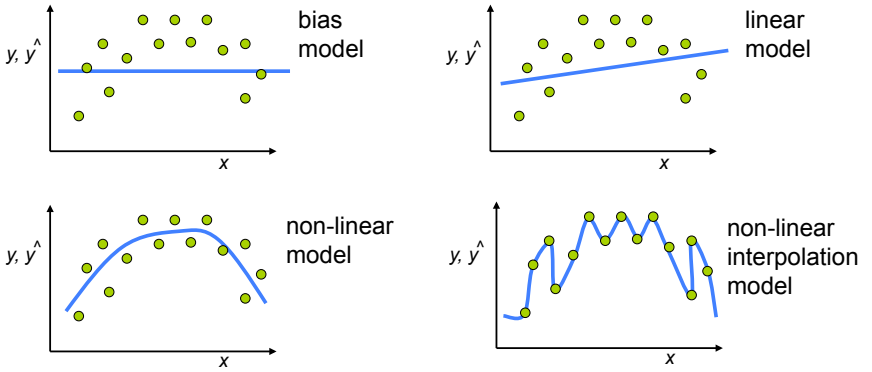
Once again, we can express the derivative of E, according to W:

ei -1 f'(pi) yj

$$\frac{\partial E}{\partial Wij} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial Yi} \cdot \frac{\partial Yi}{\partial pi} \cdot \frac{\partial pi}{\partial Wij}$$

$$= -1 \cdot ei \cdot f'(pi) \cdot yj$$

non-linear regression



Beyond supervision

What happens when there is no supervision

- number / structure of classes is unknown
- Yd can not be known in advance
- Error gradient can not be computed
- back propagation can not be computed

Beyond supervision

we can still split the space in k classes : (but is k appropriate ?)

K-means algorithm

partitions the points in the n-by-p data matrix X into k clusters.

Rows of X correspond to points, columns correspond to variables.

iterative partitioning

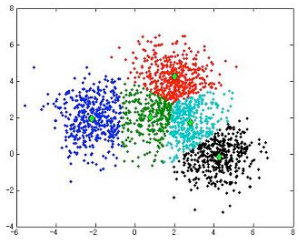
minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances.

By default, kmeans uses squared Euclidean distances.

Beyond supervision

we can still split the space in k classes : (but is k appropriate ?)

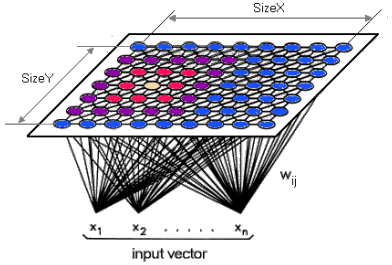
K-means algorithm

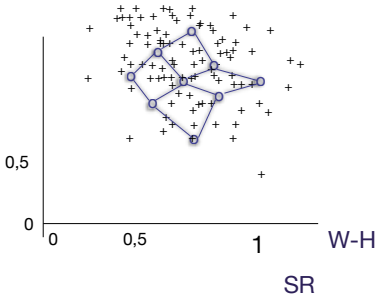


Beyond supervision

Let the network learn the topology of the input space:

kohonen, SOM

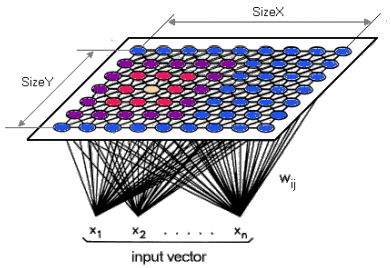


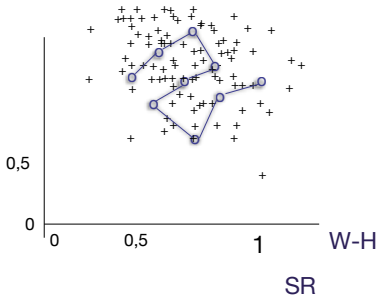


Beyond supervision

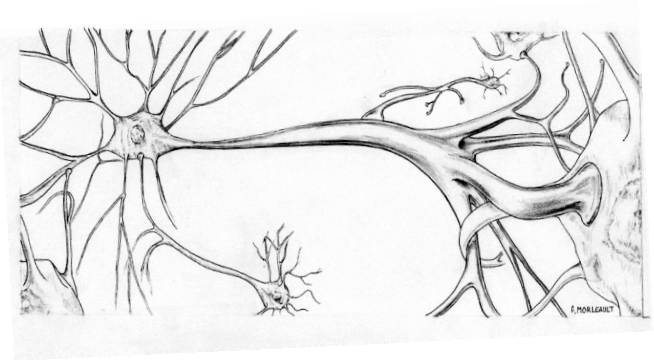
Let the network learn the topology of the input space:

kohonen, SOM





Apprentissage et Adaptation



Laboratoire ETIS
ENSEA - UCP- UMR CNRS 8051

Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement

Apprentissage de séquences temporelles

Apprentissage non supervisé

Les systèmes classiques nécessitent le corrigé d'un expert

The backpropgation algorithm looks for the minimum of the error function in weight space using the method of gradient descent

- the combination of weights that minimize this error = solution
- computation of the gradient of error at each time step
- switch to continuous activation function = continuous error function

$s_c : \mathbb{R} \to (0,1)$ defined by the expression

$$s_c(x) = \frac{1}{1+e^{-cx}}.$$

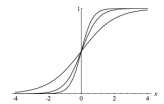


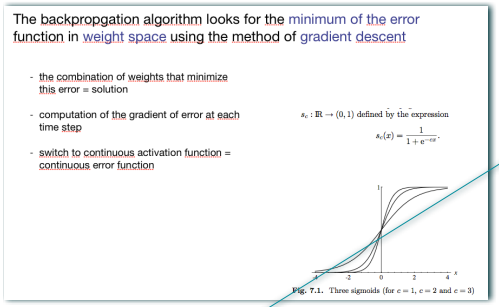
Fig. 7.1. Three sigmoids (for $c = 1$, $c = 2$ and $c = 3$)

$$\Delta w_{kh}^j = -\alpha \cdot \delta_k^j \cdot y_h^{j-1}$$

$$\delta_k^j = \begin{cases} y_k^j - y_{d_k} & \text{pour la dernirre couche} \\ \left[\sum_{i \in \text{couche } j+1} \delta_i^{j+1} \cdot w_{ik}^{j+1} \right] \cdot \sigma'(p_k^j) & \text{pour les autres couches} \end{cases}$$

Apprentissage non supervisé

Les systèmes classiques nécessitent le corrigé d'un expert



Corrigé

$$\Delta w_{kh}^j = -\alpha \cdot \delta_k^j \cdot y_h^{j-1}$$

$$\delta_k^j = \begin{cases} y_k^j - y_{d_k} & \text{pour la dernirre couche} \\ \left[\sum_{i \in \text{couche } j+1} \delta_i^{j+1} \cdot w_{ik}^{j+1} \right] \cdot \sigma'(p_k^j) & \text{pour les autres couches} \end{cases}$$

Apprentissage non supervisé

Les systèmes classiques nécessitent le corrigé d'un expert

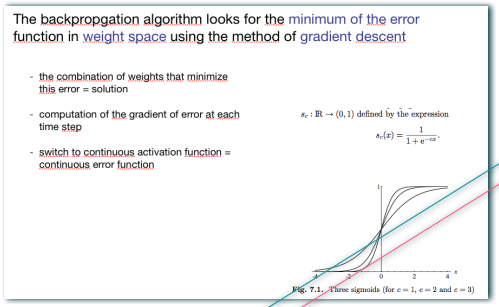
Corrigé
Erreur

$$\Delta w_{kh}^j = -\alpha \cdot \delta_k^j \cdot y_h^{j-1}$$

$$\delta_k^j = \begin{cases} y_k^j - y_{d_k} & \text{pour la dernirre couche} \\ \left[\sum_{i \in \text{couche } j+1} \delta_i^{j+1} \cdot w_{ik}^{j+1} \right] \cdot \sigma'(p_k^j) & \text{pour les autres couches} \end{cases}$$

Apprentissage non supervisé

Les systèmes classiques nécessitent le corrigé d'un expert



Corrigé
Erreur

$$\Delta w_{kh}^j = -\alpha \cdot \delta_k^j \cdot y_h^{j-1}$$

Signal d'erreur

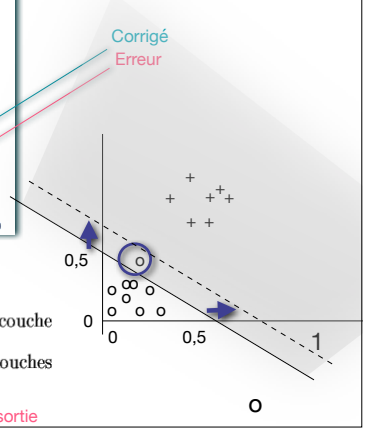
$$\delta_k^j = \begin{cases} y_k^j - y_{d_k} & \text{pour la dernirre couche} \\ \left[\sum_{i \in \text{couche } j+1} \delta_i^{j+1} \cdot w_{ik}^{j+1} \right] \cdot \sigma'(p_k^j) & \text{pour les autres couches} \end{cases}$$

Contribution des erreurs des couches de sortie

Apprentissage non supervisé

Les systèmes classiques nécessitent le corrigé d'un expert

Corrigé
Erreur



$$\Delta w_{kh}^j = -\alpha \cdot \delta_k^j \cdot y_h^{j-1}$$

Signal d'erreur

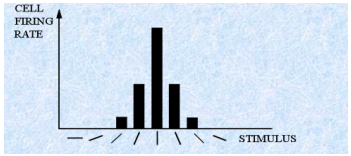
$$\delta_k^j = \begin{cases} y_k^j - y_{d_k} & \text{pour la dernirre couche} \\ \left[\sum_{i \in \text{couche } j+1} \delta_i^{j+1} \cdot w_{ik}^{j+1} \right] \cdot \sigma'(p_k^j) & \text{pour les autres couches} \end{cases}$$

Contribution des erreurs des couches de sortie

Apprentissage non supervisé

Un réseau peut il apprendre par lui-même ?

les systèmes biologiques le font.



Cf cours de L.Hafemeister

Effets de la privation visuelle durant la période critique du développement de la vision

Chez le chat :

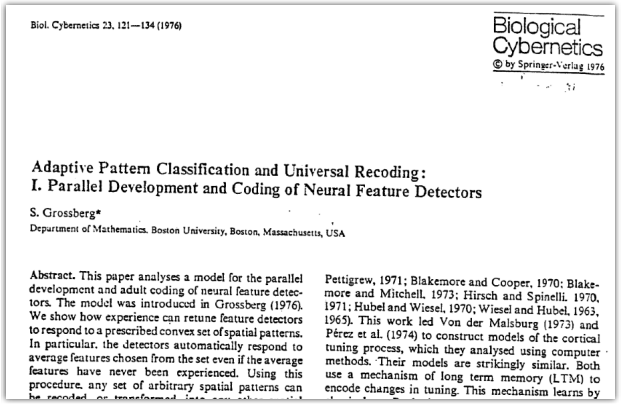
David Hubel et Torsten Wiesel ont montré que si un chaton est privé d'expérience visuelle normale pendant une période critique au début de sa vie, le câblage des neurones de son cortex visuel en est irréversiblement altéré. Un seul des deux yeux d'un chaton fut bouché dès les premiers jours de sa vie et on le laissa se développer ainsi jusqu'à l'âge adulte (atteint environ à 6 mois chez le chat) où on lui ouvrit l'œil à nouveau.

Les enregistrements électrophysiologiques dans son cortex visuel montrèrent qu'un nombre anormalement faible de neurones réagissaient à cet œil et un nombre anormalement élevé à l'œil qui était resté ouvert. Le chat était quasiment aveugle de l'oeil qui avait été bouché.

Fait remarquable, si l'on bloque l'œil d'un chat adulte durant un an, les réponses des cellules du cortex visuel demeurent en tous points identiques à celles d'un chat normal. Des expériences ultérieures ont montré que l'occlusion de l'œil n'est efficace que si la privation a lieu au cours des trois premiers mois de la vie chez le chat.

Pas de corrigé : le système apprends de manière **autonome**.

Premier exemple : INSTAR


Biol. Cybernetics 23, 121—134 (1976)

Biological Cybernetics

© by Springer-Verlag 1976

Adaptive Pattern Classification and Universal Recoding:
I. Parallel Development and Coding of Neural Feature Detectors

S. Grossberg*

Department of Mathematics, Boston University, Boston, Massachusetts, USA

Abstract. This paper analyses a model for the parallel development and adult coding of neural feature detectors. The model was introduced in Grossberg (1976). We show how experience can retune feature detectors to respond to a prescribed convex set of spatial patterns. In particular, the detectors automatically respond to average features chosen from the set even if the average features have never been experienced. Using this procedure, any set of arbitrary spatial patterns can

Pettigrew, 1971; Blakemore and Cooper, 1970; Blakemore and Mitchell, 1973; Hirsch and Spinelli, 1970, 1971; Hubel and Wiesel, 1970; Wiesel and Hubel, 1963, 1965). This work led Von der Malsburg (1973) and Pérez et al. (1974) to construct models of the cortical tuning process, which they analysed using computer methods. Their models are strikingly similar. Both use a mechanism of long term memory (LTM) to encode changes in tuning. This mechanism learns by

Grossberg, S., *Adaptive pattern classification and universal recoding: I Parallel development and coding of neural feature detectors*, Biol Cybern Volume: 23 Issue: 3 Pages: 121-34 (1976).

Premier exemple : INSTAR

Principe du “Winer Takes All” - WTA-

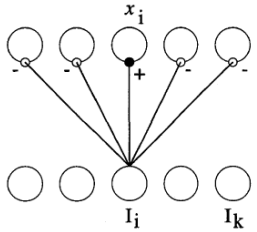
système compétitif

règle d’apprentissage simple

pas de corrigé en dehors du pattern d’entrée

mécanismes :

- LTM et STM
- Adaptation et tuning
- Inspiré des propriétés du cortex visuel des jeunes chats (cf cours de L. Hafmeister)



Premier exemple : INSTAR

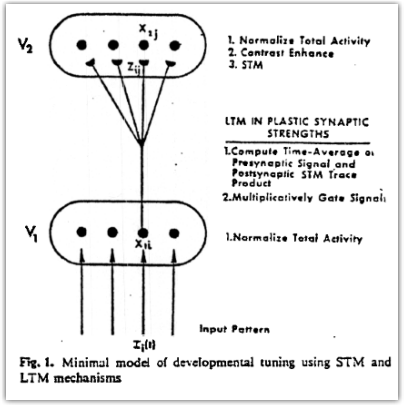


Fig. 1. Minimal model of developmental tuning using STM and LTM mechanisms

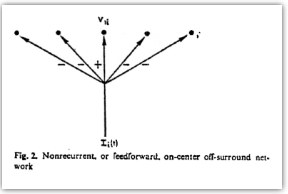


Fig. 2. Nonrecurrent, or feedforward, on-center off-surround network

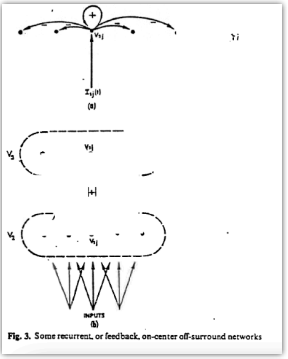


Fig. 3. Some recurrent, or feedback, on-center off-surround networks

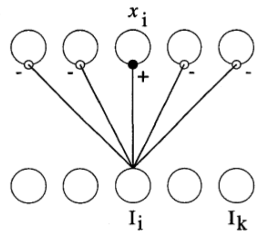
Premier exemple : INSTAR

Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.
La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern*.



non-recurrent, on center, off surround network

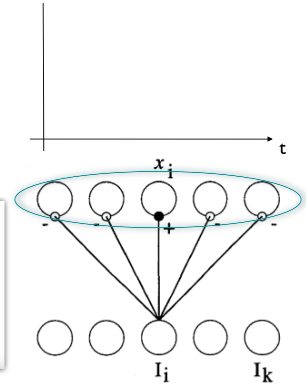
Premier exemple : INSTAR

Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.
La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern*.



non-recurrent, on center, off surround network

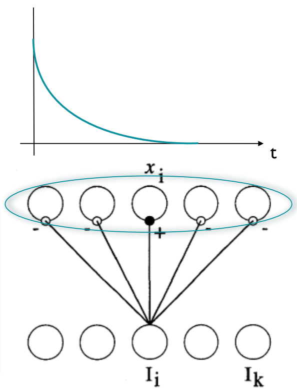
Premier exemple : INSTAR

Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.
La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern*.



non-recurrent, on center, off surround network

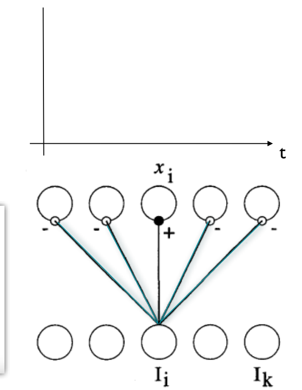
Premier exemple : INSTAR

Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.
La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern*.



non-recurrent, on center, off surround network

Premier exemple : INSTAR

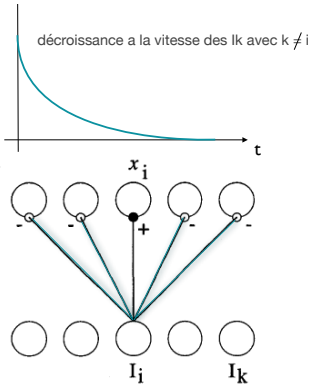
Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.

La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern.*



non-recurrent, on center, off surround network

Premier exemple : INSTAR

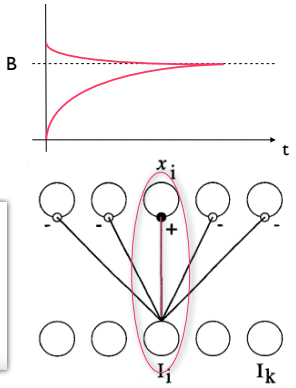
Tunning : la connectivité spécifique du INSTAR va imposer une compétition pour une seule activité gagnante : production d'un code du patern d'entrée.

x_i est la valeur instantanée du potentiel d'un neurone i du groupe de sortie. I_k est l'activité du neurone k de la couche d'entrée.

La dynamique de chacun des neurones de la couche de sortie est modélisée par l'équation :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot I_i - x_i \sum_{k \neq i} I_k$$

On pose A et $B > 0$, $I = \sum_{i=1}^{N} I_i$ et $\theta_i = \frac{I_i}{I}$. Le vecteur $\theta = (\theta_1, ..., \theta_N)^T$ est appelé *reflectance pattern.*



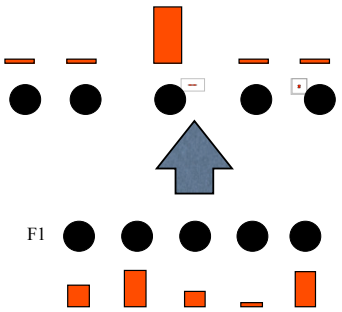
non-recurrent, on center, off surround network

Premier exemple : INSTAR

1. Propagation : $x_i = \sum_j i_j w_{ij}$

2. Tunning : près un régime transitoire, on obtient la valeur d'équilibre dx/dt=0 et un "code vainqueur" émerge.

de plus, le code produit est normalisé (B est la valeur maximum)



Premier exemple : INSTAR

Extension : l'ensemble des propriétés précédentes peut être assurée par un RCF : recurrent competitive Field (toujours le modèle INSTAR - Grossberg76)

Un mécanisme de compétition à une couche a aussi été proposé par Grossberg (voir figure 8). L'équation différentielle régissant le comportement du réseau s'écrit alors :

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot [f(x_i) + I_i] - x_i \sum_{k \neq i} [f(x_k) + I_k]$$

Avec $f(.)$ la fonction de sortie du neurone.

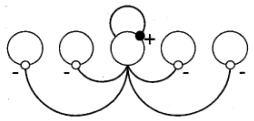


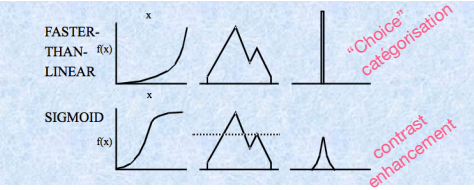
Figure 8. *Compétition sur une couche*

Premier exemple : INSTAR

avec des propriétés supplémentaires selon le choix de f(x) :

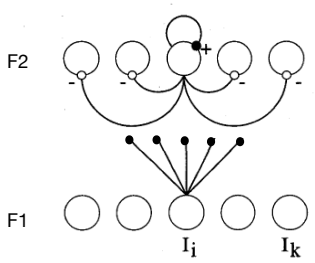
$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \cdot [f(x_i) + I_i] - x_i \sum_{k \neq i} [f(x_k) + I_k]$$

- un effet de mémoire : STM
- un effet de compétition stricte
- un effet de rehaussement de contrastes



Premier exemple : INSTAR

INSTAR complet :



COMPETITION ALLOWS ONLY ONE OR A RELATIVELY SMALL NUMBER OF CELLS TO BECOME ACTIVE

ONLY WEIGHTS PROJECTING TO ACTIVE CELLS ARE ALLOWED TO INCREASE

Premier exemple : INSTAR

The Instar Learning Law

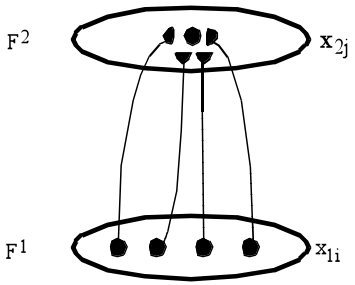
Grossberg (1976) studied the effects of using an "instar" learning law with Hebbian growth and post-synaptically gated decay in the F1 to F2 weights

If F2 is winner-take-all, learning will only occur in the weight vector projecting to the F2 cell that won the competition;

This is **Competitive learning**

$$\dot{z}_{ij} = (-z_{ij} + x_{1i}\, x_{2j})$$

$$= -z_{ij} x_{2j} + x_{1i} x_{2j}$$

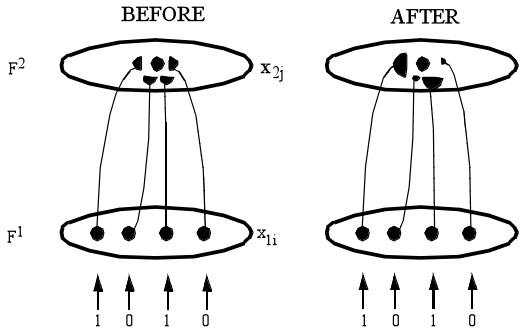


Premier exemple : INSTAR

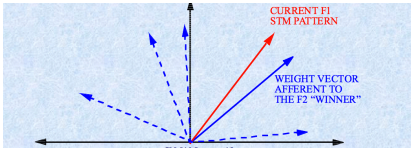
What happens to the afferent weight vector for the F2 cell that won the competition?

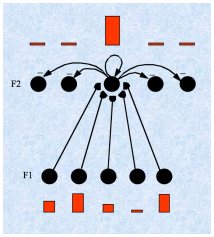
$$\dot{z}_{ij} = (-z_{ij} + x_{1i}\, x_{2j})$$

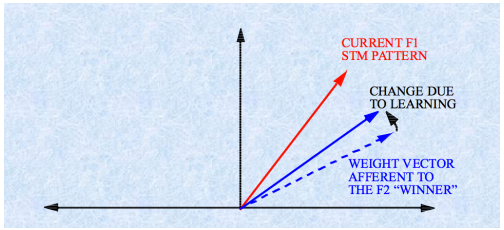
The weight vector becomes more like the current input vector.



Premier exemple : INSTAR

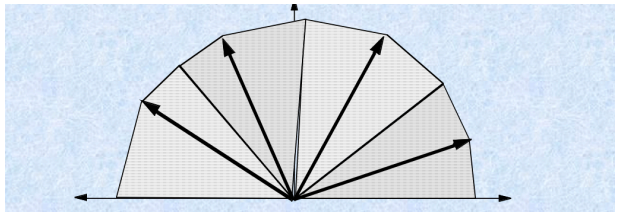






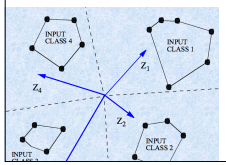
Premier exemple : INSTAR

Ainsi, chaque neurone de F2, peut se spécialiser pour une "zone" de l'espace d'entrée



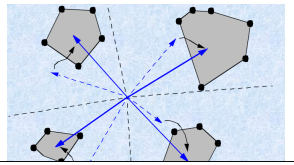
Mais il n'est pas garanti que le réseau se **stabilise** sur une partition de l'espace d'entrée.

Cela va dépendre : des conditions d'initialisation des poids F1->F2



de la répartition de l'espace d'entrée

sparse pattern theorem [Grossberg]



Premier exemple : INSTAR

-> renforcement de la sélectivité par l'apprentissage : **catégorisation**

-> explication d'un opérateur "max" biologique

-> apprentissage sans corrigé (vs perceptron/MLP/etc)

-> base de construction de la théorie ART

Mais

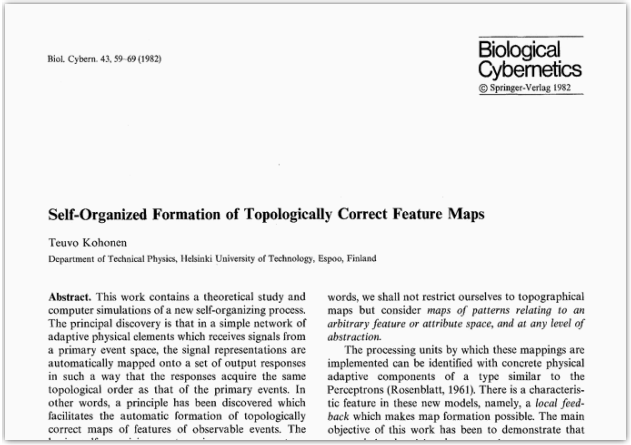
-> pb de stabilité

-> pas d'ordre local

-> pas d'ordre global dans le réseau (bien que compétition)

-> très dépendant des entrées / conditions d'initialisation

Cartes auto-organisatrices



Biol. Cybern. 43, 59–69 (1982)

Biological Cybernetics

© Springer-Verlag 1982

Self-Organized Formation of Topologically Correct Feature Maps

Teuvo Kohonen

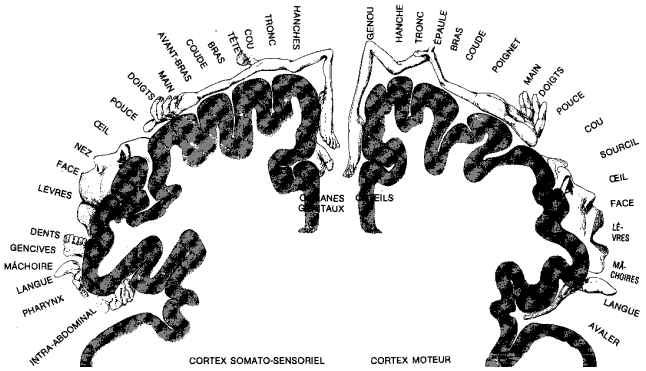
Department of Technical Physics, Helsinki University of Technology, Espoo, Finland

Abstract. This work contains a theoretical study and computer simulations of a new self-organizing process. The principal discovery is that in a simple network of adaptive physical elements which receives signals from a primary event space, the signal representations are automatically mapped onto a set of output responses in such a way that the responses acquire the same topological order as that of the primary events. In other words, a principle has been discovered which facilitates the automatic formation of topologically correct maps of features of observable events. The

words, we shall not restrict ourselves to topographical maps but consider *maps of patterns relating to an arbitrary feature or attribute space, and at any level of abstraction.*

The processing units by which these mappings are implemented can be identified with concrete physical adaptive components of a type similar to the Perceptrons (Rosenblatt, 1961). There is a characteristic feature in these new models, namely, a *local feedback* which makes map formation possible. The main objective of this work has been to demonstrate that

Cartes auto-organisatrices



Cartes auto-organisatrices

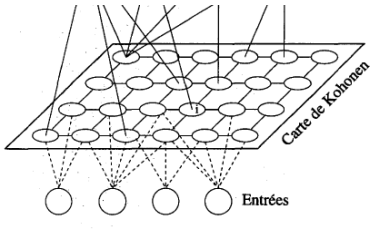
homonculus :

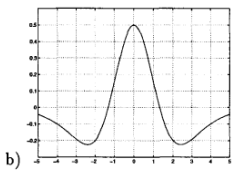
Cartes auto-organisatrices

comme dans le modèle INSTAR :
- des connections latérales définissent un système de compétition pour l'élection du code gagnant

- des connections des Entrées vers la Carte réalisent un apprentissage du vecteur d'entrée par le code gagant (mais pas que)

a la différence du modèle INSTAR :
prise en compte de la topologie de la couche supérieure.
la proximité des neurones de la carte définit une topologie, qui va exercer une contrainte sur l'apprentissage : les voisins du gagnant réalisent aussi un apprentissage du vecteur d'entrée : coopération





Cartes auto-organisatrices

Le mécanisme d'apprentissage est le suivant. La carte comporte N neurones organisés sur une grille rectangulaire. Chaque neurone est relié à M entrées. Le vecteur des poids est donné par $\mathbf{W_i} = [W_{i1}, ..., W_{iM}]^T$. Le vecteur d'entrée est donné par $\mathbf{I} = [I_1, ..., I_M]$.

- Les poids des liaisons latérales sont initialisés aléatoirement.
- On calcule la distance entre le vecteur présenté et le vecteur de poids de chaque neurone :

$$d_i = \sum_{j=1}^{M} (I_j - W_{ij})^2$$

- On choisit le neurone dont la distance avec le vecteur d'entrée est la plus petite.
- Les poids de ce neurone sont mis à jour selon l'équation :

$$W_{kj}(t+1) = W_{kj}(t) + \eta(t) \cdot (I_j(t) - W_{kj}(t))$$

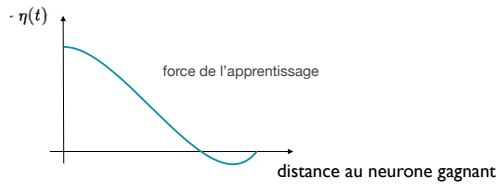
- Le coefficient d'apprentissage $\eta(t)$ est choisi dans $[0, 1]$. Il doit décroître peu à peu.
- Le voisinage des neurones est mis à jour de manière à se rétrécir au cours du temps.
- On recommence sur l'ensemble des motifs jusqu'à la stabilisation complète des poids.

Cartes auto-organisatrices

Tuning : balance entre la compétition et la coopération.

métaphore de la pâte à tarte

- $\eta(t)$ varie dans le temps



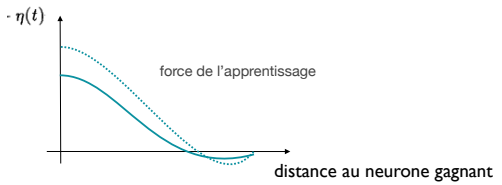
Cartes auto-organisatrices

Tuning : balance entre la compétition et la coopération.

métaphore de la pâte à tarte

- $\eta(t)$ varie dans le temps

un peu plus tard...



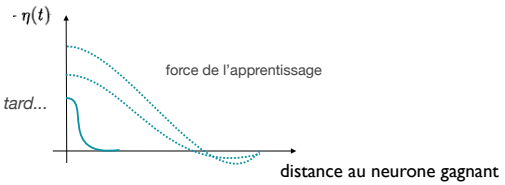
Cartes auto-organisatrices

Tuning : balance entre la compétition et la coopération.

métaphore de la pâte à tarte

- $\eta(t)$ varie dans le temps

encore un peu plus tard...



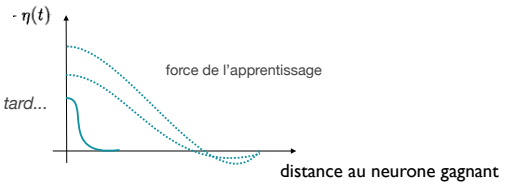
Cartes auto-organisatrices

Tuning : balance entre la compétition et la coopération.

métaphore de la pâte à tarte

- $\eta(t)$ varie dans le temps

encore un peu plus tard...



comment faire ?

Cartes auto-organisatrices

```
double dog0[NB_DOG]={1.0,0.8,0.6,0.4,0.2,-0.1,-0.1};
double dog1[NB_DOG]={0.7,0.5,0.3,0.1,0.0,-0.05,0.0};
double dog2[NB_DOG]={0.3,0.2,0.1,-0.02,0.0,0.0,0.0};
double dog3[NB_DOG]={0.2,0.01,0.0,0.0,0.0,0.0,0.0};
double eps=EPS;
```

Cartes auto-organisatrices

```
/*retourne l'index du neurone gagnant*/
int calcul_dist_min(Point ville, Point w[NB_NEURONES]){
  double dist_min=2.0,d;
  int i, indice_min;

  for(i=0; i<NB_NEURONES; i++){
    d=sqrt((w[i].x-ville.x)*(w[i].x-ville.x)+(w[i].y-ville.y)*(w[i].y-ville.y));
    if(d<dist_min){ dist_min=d; indice_min=i; }
  }
  return indice_min;
}
```

Cartes auto-organisatrices

```
void maj_poids(Point ville,Point w[NB_NEURONES],int gagnant,double dog[NB_DOG],double eps){

  int i;

  w[gagnant].x+=eps*dog[0]*(ville.x-w[gagnant].x);
  w[gagnant].y+=eps*dog[0]*(ville.y-w[gagnant].y);

  for(i=1; i<NB_DOG; i++){
      if(gagnant+i<NB_NEURONES){
    w[(gagnant+i)%NB_NEURONES].x+=eps*dog[i]*(ville.x-w[(gagnant+i)%NB_NEURONES].x);
    w[(gagnant+i)%NB_NEURONES].y+=eps*dog[i]*(ville.y-w[(gagnant+i)%NB_NEURONES].y);
      }
    if(gagnant-i>=0){
      w[gagnant-i].x+=eps*dog[i]*(ville.x-w[gagnant-i].x);
      w[gagnant-i].y+=eps*dog[i]*(ville.y-w[gagnant-i].y);
    }
    else{
      w[(gagnant-i)+NB_NEURONES].x+=eps*dog[i]*(ville.x-w[(gagnant-i)+NB_NEURONES].x);
      w[(gagnant-i)+NB_NEURONES].y+=eps*dog[i]*(ville.y-w[(gagnant-i)+NB_NEURONES].y);
    }
  }
```

Cartes auto-organisatrices

```
int i;
Point ville;
int gagnant;
if(calc){
  for(i=0;i<NB_VILLE*1;i++){
    ville=choix_ville(villes);
    gagnant=calcul_dist_min(ville,w);
    cpt++;

    if(cpt<4*COEF*NB_VILLE){
  maj_poids(ville,w,gagnant,dog0,eps);
    }
    if(cpt==4*COEF*NB_VILLE){
  calc=0;
  maj_poids(ville,w,gagnant,dog1,eps);
    }
    if((cpt>4*COEF*NB_VILLE)&&(cpt<7*COEF*NB_VILLE)){
  maj_poids(ville,w,gagnant,dog1,eps);
    }
    if(cpt==7*COEF*NB_VILLE){
  calc=0;
  maj_poids(ville,w,gagnant,dog2,eps);
    }
    if((cpt>7*COEF*NB_VILLE)&&(cpt<10*COEF*NB_VILLE)){
  maj_poids(ville,w,gagnant,dog2,eps);
    }
    if(cpt==10*COEF*NB_VILLE){
  calc=0;
  maj_poids(ville,w,gagnant,dog3,eps);
    }
    if(cpt>10*COEF*NB_VILLE){
  maj_poids(ville,w,gagnant,dog3,eps);
    }
  }
  glutPostRedisplay();
}
```

Bilan intermédiare

Bilan intermédiaire :

Le modèle INSTAR pose les bases du bénéfice de la compétition :

- assure qu'un code réponde de manière sélective à un pattern d'entrée (catégorisation sans supervision).
- fournit une explication biologique de l'opérateur max

Le modèle des SOM prolonge ce bénéfice :

- la coopération lors de la phase d'apprentissage assure la généralisation

Mais,

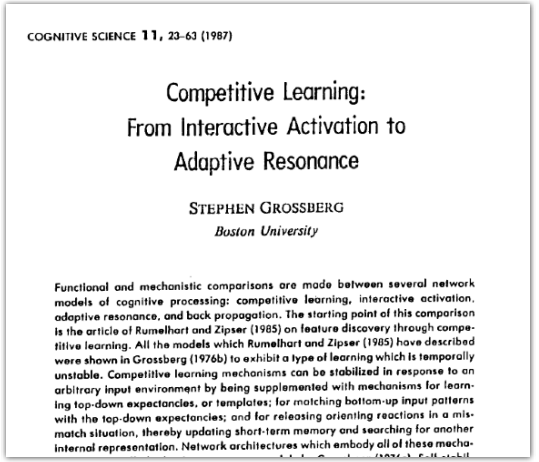
- la capacité de généralisation d'INSTAR est limitée
- les SOM réclament énormément d'itération d'apprentissage (plusieurs centaines de milliers)
- les deux solutions sont peu "plastiques" : que se passe t'il si une nouvelle catégorie apparait ? -> déstructuration des SOM - reprendre l'apprentissage de zéro ?
- le nombre de catégories est fixé à l'avance (INSTAR,SOM,K-mean, etc...)

Dans ce cas, comment imaginer la conception de systèmes intelligents dont la réponse est dépendante d'un environnement ou la formation des catégories est progressive ?

Nous n'avons plus besoin de signal d'erreur. $y_k^j - y_{d_k}$

Mais alors : qu'est-ce qui le remplace et permet d'effectuer l'apprentissage ?

Adaptive Resonance Theory



COGNITIVE SCIENCE 11, 23-63 (1987)

Competitive Learning:
From Interactive Activation to
Adaptive Resonance

STEPHEN GROSSBERG
Boston University

Functional and mechanistic comparisons are made between several network models of cognitive processing: competitive learning, interactive activation, adaptive resonance, and back propagation. The starting point of this comparison is the article of Rumelhart and Zipser (1985) on feature discovery through competitive learning. All the models which Rumelhart and Zipser (1985) have described were shown in Grossberg (1976b) to exhibit a type of learning which is temporally unstable. Competitive learning mechanisms can be stabilized in response to an arbitrary input environment by being supplemented with mechanisms for learning top-down expectancies, or templates; for matching bottom-up input patterns with the top-down expectancies; and for releasing orienting reactions in a mismatch situation, thereby updating short-term memory and searching for another internal representation. Network architectures which embody all of these mecha-

ART

Idée : résoudre le dilemme entre stabilité et Plasticité.

Un système qui a appris une catégorisation correctement est dit **stable**. On gèle alors généralement les paramètres qui ont servi l'apprentissage pour conserver les stabilité des réponses et permettre l'utilisation. *phase d'apprentissage* / *phase d'utilisation*.

Un système qui apprend est dit **plastique**, car il se modifie rapidement pour apprendre les catégories de l'espace d'entrée. Il n'est pas stable dans le sens ou l'ensemble des réponses du réseau sont influencées par le nouvel apprentissage.

Vitesse d'apprentissage

$$W_{kj}(t+1) = W_{kj}(t) + \eta(t) \cdot (I_j(t) - W_{kj}(t))$$

ART

Proposition de [Carpenter&Grossberg86]

Utiliser un ensemble de groupes neuronaux en mesure de calculer l'erreur d'une catégorie sélectionnée par rapport aux vecteurs d'entrées.

Cette erreur sera comparée à un paramètre de **vigilance**

Le résultat de la comparaison décidera de la décision de créer une nouvelle catégorie ou de renforcer une catégorie déjà existante (moyennage du code).

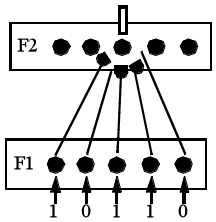
Système de va-et-viens entre les poids du code gagnant et le pattern d'entrée : **Résonance**.

Liée a la question sur INSTAR : comment assurer que dans un environnement non sparse, les poids de F2 ne soient pas re-codés en permanence (stabilité des catégories) ?

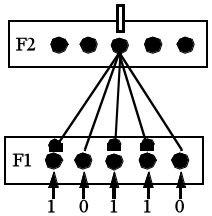
ART

Top-down Feedback : OUTSTAR

des poids apprennent le pattern d'entrée si la cellule post-synaptique est active



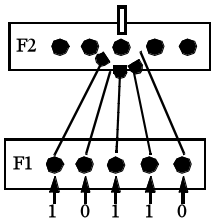
d'autres poids apprennent le code si la cellule pre-synaptique est active



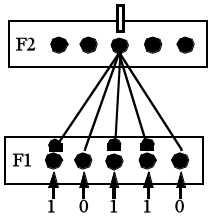
ART

Top-down Feedback : OUTSTAR

des poids apprennent le pattern d'entrée si la cellule post-synaptique est active



d'autres poids apprennent le code si la cellule pre-synaptique est active



Ces deux règles peuvent être différentes (Hebb)

ART

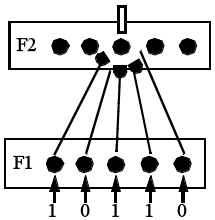
Top-down Feedback : OUTSTAR

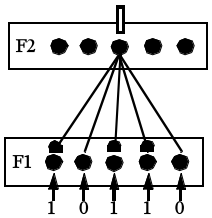
competition pour le code de la catégorie

des poids apprennent le pattern d'entrée si la cellule post-synaptique est active

top-down prototype moyen de la catégorie

d'autres poids apprennent le code si la cellule pre-synaptique est active

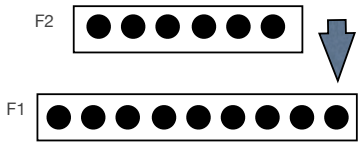




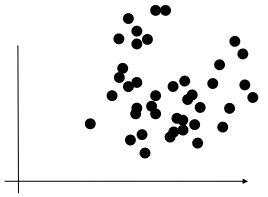
Ces deux règles peuvent être différentes (Hebb)

ART

Notion de "catalogue" par résonance

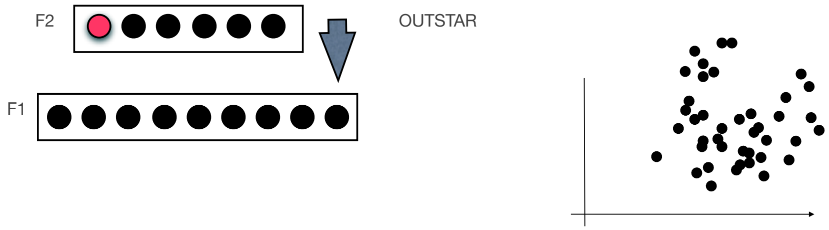


OUTSTAR



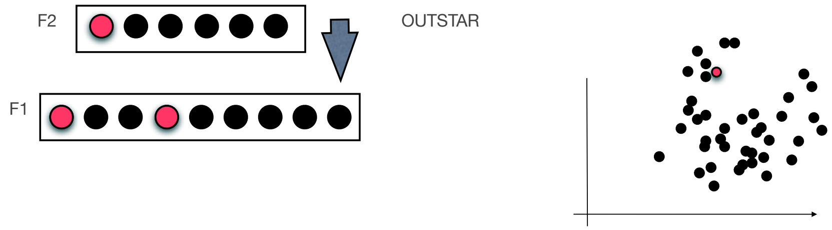
ART

Notion de “catalogue” par résonance



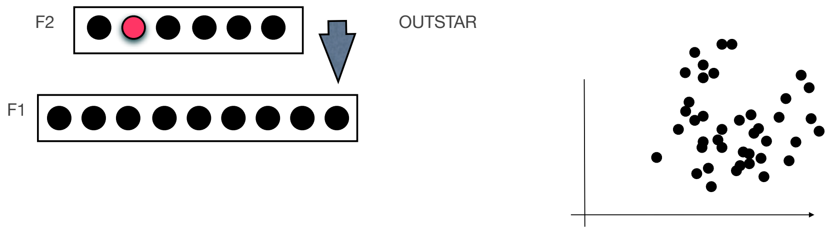
ART

Notion de “catalogue” par résonance



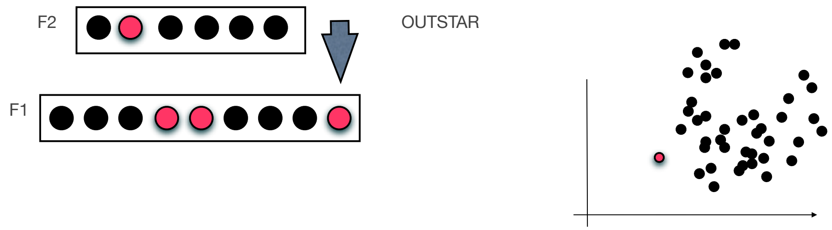
ART

Notion de “catalogue” par résonance



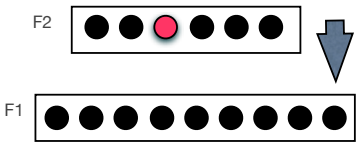
ART

Notion de “catalogue” par résonance

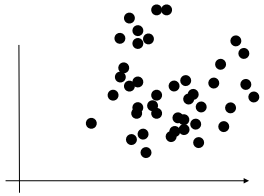


ART

Notion de “catalogue” par résonance

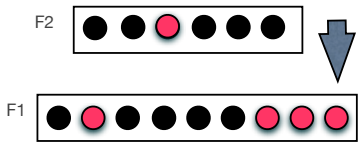


OUTSTAR

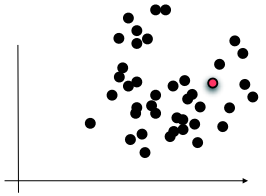


ART

Notion de “catalogue” par résonance



OUTSTAR



ART

An Introduction to Computing with Neural Nets

Richard P. Lippmann

Abstract

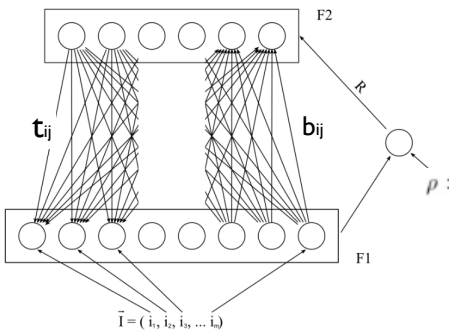
Artificial neural net models have been studied for many years in the hope of achieving human-like performance in the fields of speech and image recognition. These models are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. Computational elements or nodes are connected via weights that are typically adapted during use to improve performance. There has been a recent resurgence in the field of artificial neural nets caused by new net topologies and algorithms, analog VLSI implementation techniques, and the belief that massive parallelism is essential for high performance speech and image recognition. This paper provides an introduction to the field of artificial neural nets by re-

using massively parallel nets composed of many computational elements connected by links with variable weights.

Computational elements or nodes used in neural net models are nonlinear, are typically analog, and may be slow compared to modern digital circuitry. The simplest node sums N weighted inputs and passes the result through a nonlinearity as shown in Fig. 1. The node is characterized by an internal threshold or offset θ and by the type of nonlinearity. Figure 1 illustrates three common types of nonlinearities; hard limiters, threshold logic elements, and sigmoidal nonlinearities. More complex nodes may include temporal integration or other types of time dependencies and more complex mathematical operations than summation.

Neural net models are specified by the net topology,

ART



Summary of ART 1

(Lippmann, 1987). N = number of F1 units.

Step 1: Initialization

$$t_{ij} = 1, b_{ij} = \frac{1}{1+N}$$

Set vigilance parameter $\rho : 0 \leq \rho \leq 1$

Step 2: apply new input (binary x_i)

Step 3: compute F2 activation

$$\mu_j = \sum_{i=1}^{N} b_{ij} x_i$$

Step 4: find best matching node k, where $\mu_k \geq \mu_j \quad \forall j$.

Step 5: vigilance test

$$|X| = \sum_{i=1}^{N} x_i \qquad |T \cdot X| = \sum_{i=1}^{N} t_{ik} x_i$$

$$\text{Is } \frac{|T \cdot X|}{|X|} > \rho ?$$

If **no**, go to step 6. If **yes** go to step 7.

Step 6: mismatch/reset: set $\mu_k = 0$ and go to step 4.

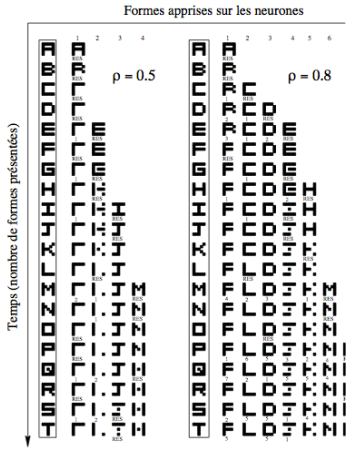
Step 7: resonance — adapt best match

$$t_{ik} \leftarrow t_{ik} x_i$$

$$b_{ik} \leftarrow \frac{t_{ik}}{.5 + \sum_{i=1}^{N} t_{ik} x_i}$$

Step 8: Re-enable all F2 units and go to step 2

ART



11. *Exemple de formation de catégories pour deux valeurs du paramètre de vigilance.*

Counter Propagation

[Nielsen86]

Counter Propagation

[Nielsen86]

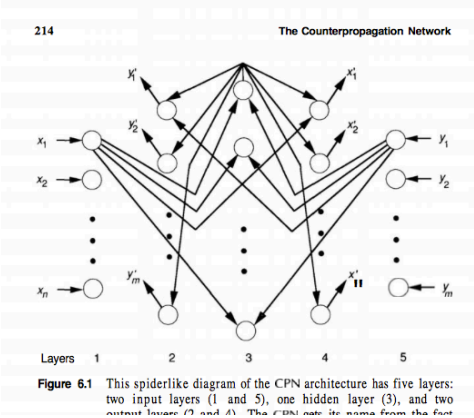


Figure 6.1 This spiderlike diagram of the CPN architecture has five layers: two input layers (1 and 5), one hidden layer (3), and two output layers (2 and 4). The CPN gets its name from the fact

PerAc

ELSEVIER

Robotics and Autonomous Systems

Robotics and Autonomous Systems 16 (1995) 291–320

PerAc: A neural architecture to control artificial animals

Philippe Gaussier [a,*], Stéphane Zrehen [b]

[a] *ENSEA ETIS, 6 av du Ponceau, 95014 Cergy Pontoise Cedex, France*
[b] *Laboratoire de Microinformatique, EPFL-DI, CH-1015 Lausanne, Switzerland*

Abstract

In this paper, we propose a new neural architecture called PerAc witch is a systematic way to decompose the control of an autonomous robot in perception and action flows. The PerAc architecture is used for the simulation of a vision system with a moving eye and then for landmark-based navigation on a mobile robot to learn without any a priori symbolic representation.

Keywords: Sensory-motor loop; Vision; Navigation; Neural building block

PerAc

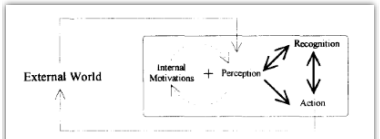


Fig. 3. The concept of PerAc architecture to control autonomous robots.

PerAc

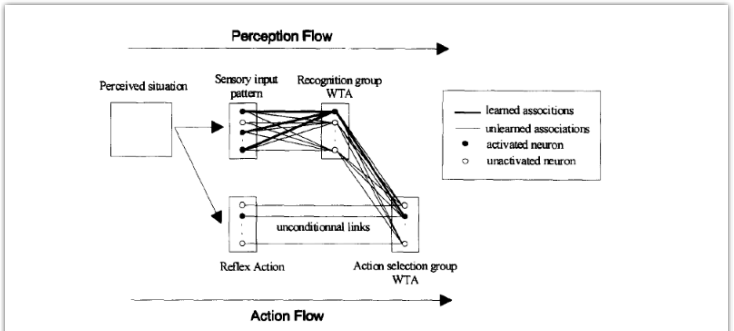


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

PerAc

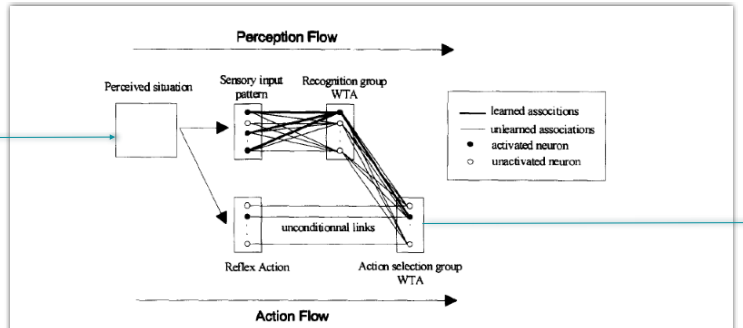


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

Rassemblons...

Counter Propagation [Nielsen]

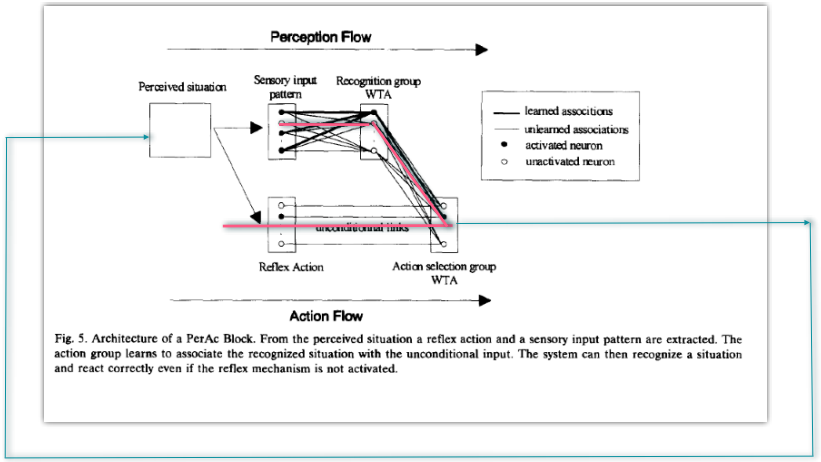


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

Rassemblons...

Counter Propagation [Nielsen]

WTA, ~~Kohonen,~~ ART (SAW)

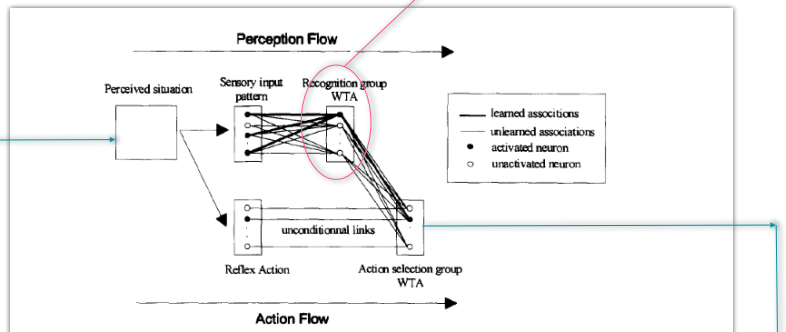


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

Rassemblons...

Counter Propagation [Nielsen]

WTA, ~~Kohonen,~~ ART (SAW)

LMS

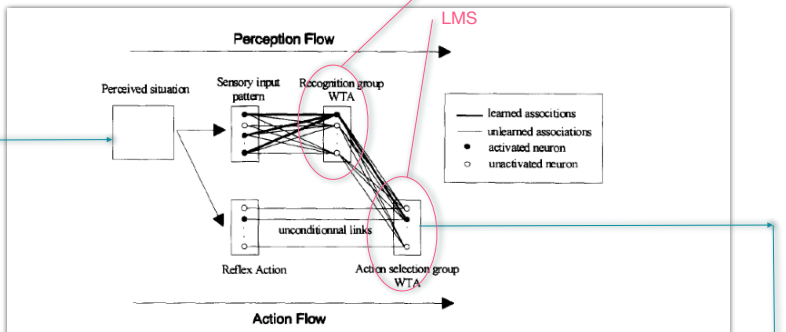


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

Rassemblons...

Receptive Fields /vision (Hafemeister)

CounterPropagation [Nielsen86]

WTA, ~~Kohonen,~~ ART (SAW)

LMS

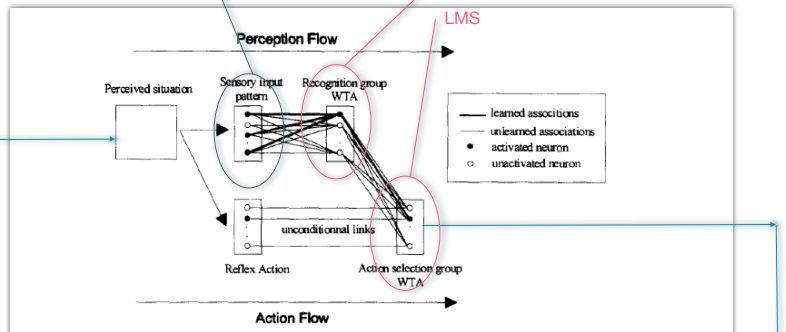


Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

application

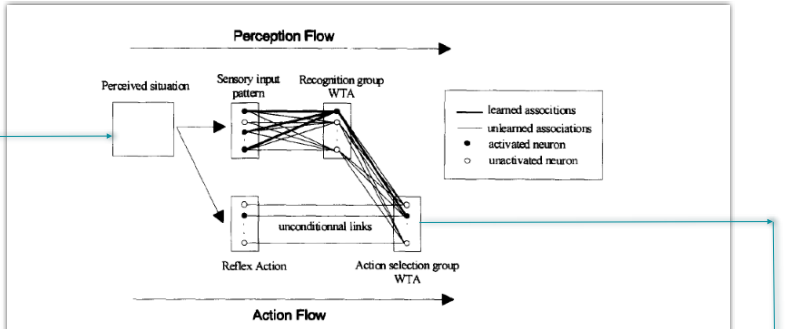
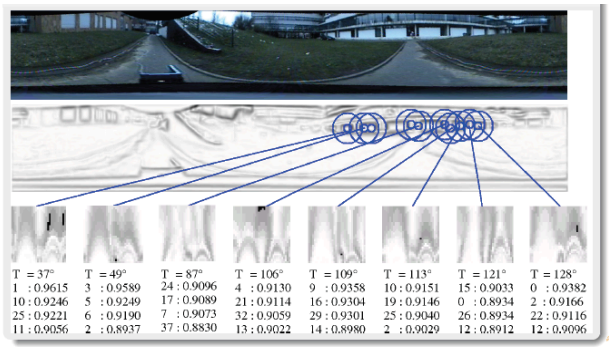
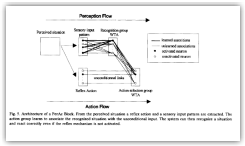


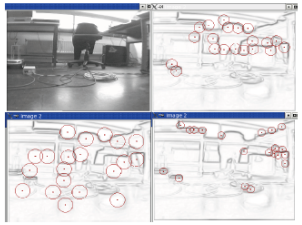
Fig. 5. Architecture of a PerAc Block. From the perceived situation a reflex action and a sensory input pattern are extracted. The action group learns to associate the recognized situation with the unconditional input. The system can then recognize a situation and react correctly even if the reflex mechanism is not activated.

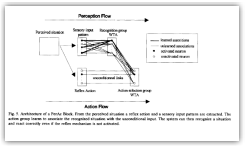
application



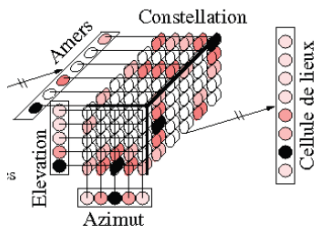


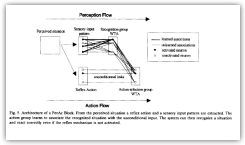
application



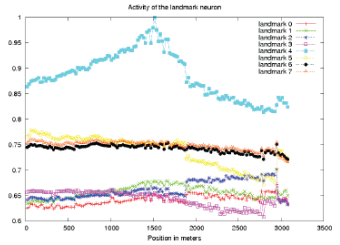


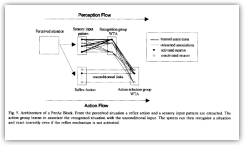
application



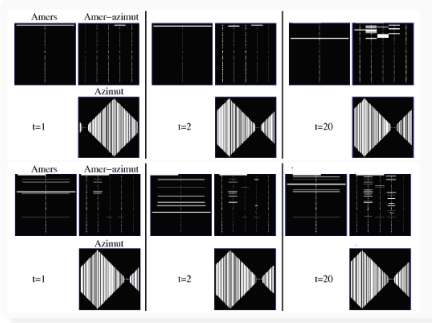


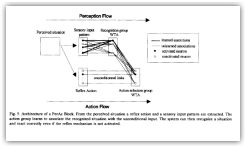
application



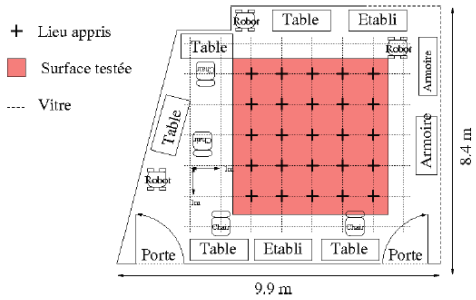


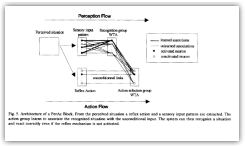
application



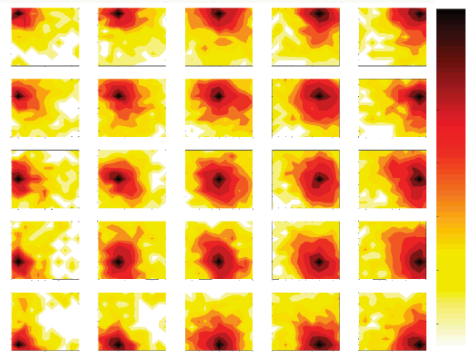


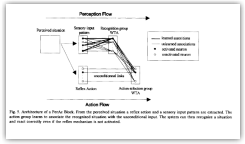
application





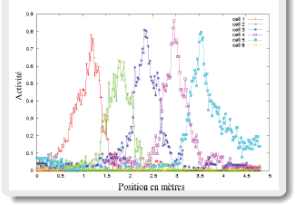
application



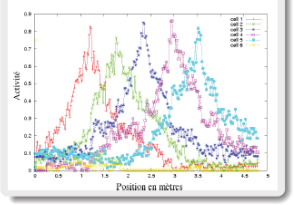


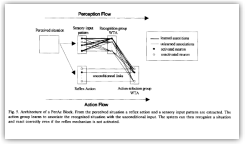
application

Compétition dure (1 gagnant)

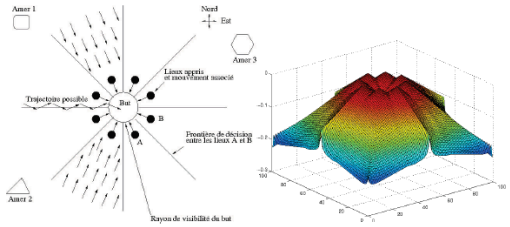


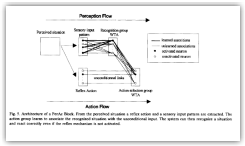
Compétition molle (4 gagnants)



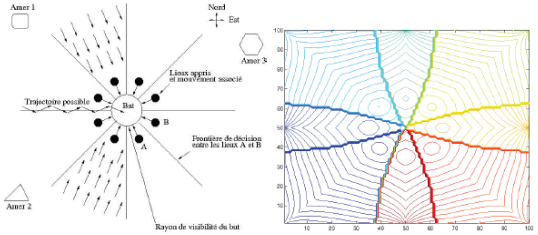


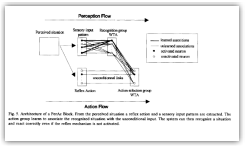
application



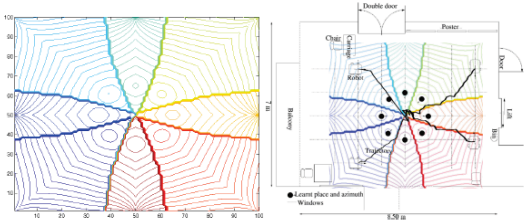


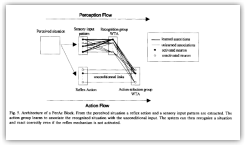
application





application





Learning

Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement

Apprentissage de séquences temporelles

Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement : du conditionnement au renforcement

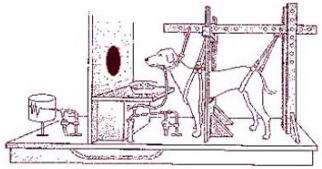
Apprentissage de séquences temporelles

Le conditionnement classique

dit conditionnement pavlovien [Pavlov27]

Un psychologue, rentrant d'une expérience, raconte à un ami : "Nous disposons maintenant d'une technique très sûre pour amener nos animaux à faire exactement ce que nous voulons; je viens justement de conditionner un rat à appuyer sur un levier." Le rat en question a été remis dans sa cage habituelle avec un de ses congénères; il raconte à celui-ci : "Nous disposons maintenant d'une technique très sûre pour amener l'homme du laboratoire à faire ce que nous désirons : quand j'ai faim, il suffit que j'appuie sur le levier pour qu'il obéisse immédiatement et me donne à manger; il est littéralement conditionné."

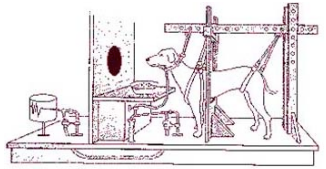
D'après Le Ny [lN92]



Le conditionnement classique

dit conditionnement pavlovien [Pavlov27]

Le conditionnement classique a pu être formalisé dès 1927 grâce aux expériences du psychologue russe Pavlov [Pav27]. L'expérience la plus connue de Pavlov est celle du chien et de la clochette. Le processus expérimental consiste à présenter de la nourriture à un chien qui n'a pas mangé depuis 24 heures. Sa réaction "*réflexe*" à la vue de la nourriture est de saliver. Le protocole du conditionnement consiste alors à faire tinter une clochette à chaque fois que l'on présente la nourriture. Après plusieurs présentations concomitantes de la nourriture et de la clochette, le chien se met à saliver dès qu'il entend la clochette, même si la nourriture n'est plus présentée. On dit alors que le chien a été *conditionné* à saliver lorsqu'il entend la clochette.*



** doit pouvoir s'appliquer a un étudiant face aux crêpes du BDE*

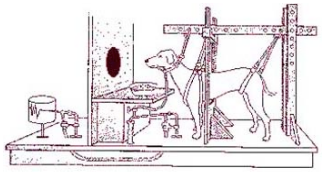
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

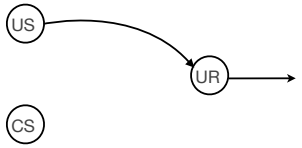
Le conditionnement classique

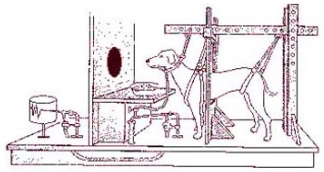
terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-





** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

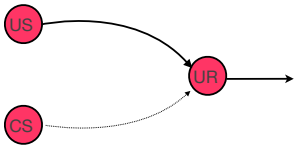
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

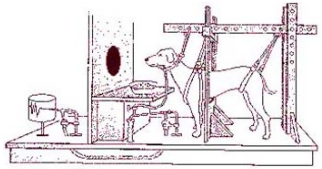
Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



$$dW/dt = -a + eps.(x_{CS}.x_{UR})$$



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

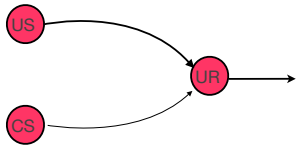
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

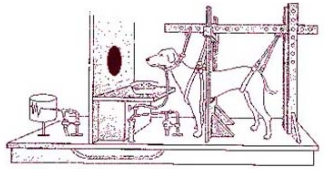
Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



$$dW/dt = -a + eps.(x_{CS}.x_{UR})$$



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

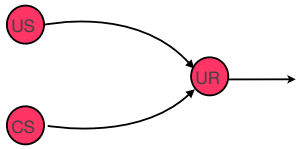
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

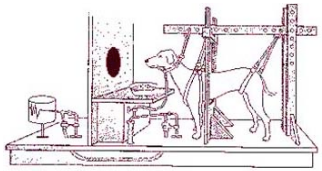
Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



$dW/dt = -a + eps.(x_{CS}.x_{UR})$



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

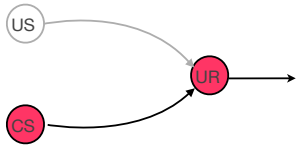
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

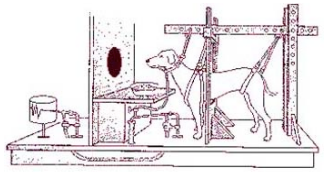
Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



$dW/dt = -a.x_{CS}.x_{UR}$



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

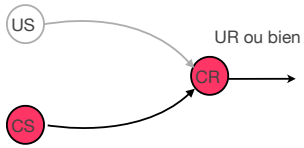
Le conditionnement classique

terminologie :
un lien stimulus-réponse existe déjà avant tout conditionnement : lorsque le chien est affamé il salive *forcement.* C'est le lien *inconditionnel*.

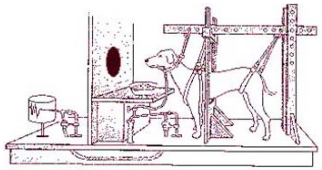
Vue de la nourriture en situation de faim : US -unconditionnal Stimulus-
Réaction de salivation : UR -unconditional response-

Clochette avant conditionnement : CS -condtionnal stimulus-

Réaction de salivation après conditionnement : CR -conditionnal response-



$dW/dt = -a + eps.(x_{CS}.x_{UR})$



** doit pouvoir s'appliquer a un étudiant face aux crèpes du BDE*

Le conditionnement classique

Rescorla et Wagner, deux psychophysiologistes, ont proposé un modèle du conditionnement pavlovien qui essaye de palier les problèmes de saturation et explique le phénomène d'extinction [Res72] (nous verrons de quoi il s'agit plus loin, et nous en envisagerons les conséquences). Leur idée est que la variation du poids qui associe le CS à la UR dépend de la différence entre la réponse obtenue et celle désirée. Ce principe peut être formalisé selon l'expression suivante :

$$W_{ij}[t+1] = W_{ij}[t] + \alpha_i \cdot I_i \cdot \beta \cdot (O_{Max} - O_j) \quad (1)$$

Avec $\alpha_i \cdot I_i$ l'intensité du CS (α_i est la constante d'apprentissage); β le coefficient d'apprentissage; O_{Max}, la réponse souhaitée; O_j, la réponse effective.

Le conditionnement classique

Rescorla et Wagner, deux psychophysiologistes, ont proposé un modèle du conditionnement pavlovien qui essaye de palier les problèmes de saturation et explique le phénomène d'extinction [Res72] (nous verrons de quoi il s'agit plus loin, et nous en envisagerons les conséquences). Leur idée est que la variation du poids qui associe le CS à la UR dépend de la différence entre la réponse obtenue et celle désirée. Ce principe peut être formalisé selon l'expression suivante :

$$W_{ij}[t+1] = W_{ij}[t] + \alpha_i \cdot I_i \cdot \beta \cdot (O_{Max} - O_j) \quad (1)$$

Avec $\alpha_i \cdot I_i$ l'intensité du CS (α_i est la constante d'apprentissage); β le coefficient d'apprentissage; O_{Max}, la réponse souhaitée; O_j, la réponse effective.

WIDROW - HOFF

Explique l'extinction du conditionnement s'il n'est plus entretenu

Le conditionnement instrumental

De nombreux chercheurs se sont lancés à la suite de Pavlov, dans l'étude du conditionnement. Et c'est lors d'une expérience de Miller et Konorski [Kon48], dès 1928, que naquit l'idée d'une autre forme possible de conditionnement. Dans cette étude, l'expérimentateur prenait la patte d'un chien et la fléchissait, en même temps qu'il lui donnait à manger. Après plusieurs répétitions, comme ils s'y attendaient d'après les théories du conditionnement pavlovien, le fait de fléchir la patte du chien (CS) le faisait saliver (UR puis RC). Cependant, une nouvelle réaction non-attendue apparut. Le chien se mit à fléchir la patte de lui-même[3]. Cette réaction gagna en vitesse et en force comme on continuait à la récompenser. Le chien avait appris à associer le mouvement qu'il avait effectué avec l'apparition de la récompense. Il espérait donc qu'en fléchissant la patte il pourrait obtenir de la nourriture. On poursuivit l'expérience en administrant de l'acide après flexion de

Le conditionnement instrumental

De nombreux chercheurs se sont lancés à la suite de Pavlov, dans l'étude du conditionnement. Et c'est lors d'une expérience de Miller et Konorski [Kon48], dès 1928, que naquit l'idée d'une autre forme possible de conditionnement. Dans cette étude, l'expérimentateur prenait la patte d'un chien et la fléchissait, en même temps qu'il lui donnait à manger. Après plusieurs répétitions, comme ils s'y attendaient d'après les théories du conditionnement pavlovien, le fait de fléchir la patte du chien (CS) le faisait saliver (UR puis RC). Cependant, une nouvelle réaction non-attendue apparut. Le chien se mit à fléchir la patte de lui-même[3]. Cette réaction gagna en vitesse et en force comme on continuait à la récompenser. Le chien avait appris à associer le mouvement qu'il avait effectué avec l'apparition de la récompense. Il espérait donc qu'en fléchissant la patte il pourrait obtenir
de la nourriture. On poursuivit l'expérience en administrant de l'acide après flexion de la patte. Le chien réagit alors par un mouvement d'extension de la patte, de manière à résister à la flexion. Dans ce cas, le chien avait appris à associer le mouvement à la douleur. En tendant la patte, il espérait ainsi échapper à cette sensation.

Le conditionnement instrumental

Skinner [Ski53]

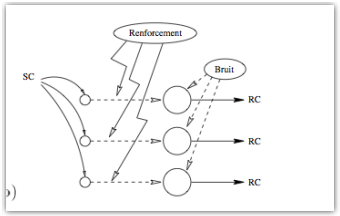
Dans ses expériences, un rat est mis dans une boite disposant d'un abreuvoir et d'un distributeur de nourriture muni d'un levier. Au début, le rat fait des mouvements erratiques jusqu'à ce qu'il appuie par inadvertance sur le levier, lui permettant ainsi d'obtenir une boulette de nourriture. On s'aperçoit alors qu'au cours du temps, la fréquence des actions de pression sur le levier augmente très rapidement.

Dans d'autres expériences, appelées *apprentissage d'échappement*, le rat est mis dans une cage et l'expérimentateur fait passer un courant électrique dans le grillage, à moins que le rat n'appuie sur un levier particulier. On constate alors que le rat apprend petit à petit à faire la séquence de mouvements lui permettant d'appuyer sur le levier. Un dernier type d'expérience (*apprentissage d'évitement*), consiste à faire entendre un son avant de créer des décharges électriques dans la cage. Le résultat est que le rat apprend à appuyer sur le levier dès qu'il entend ce son.

A la différence de l'expérience de Konorski, on voit que dans le cas des expériences de Skinner, il n'y a aucun US qui fasse réagir le rat. Skinner parle alors d'activité "*émise*" plutôt que " *déclenchée*" et nomme le phénomène qu'il a observé, *conditionnement opérant* (synonyme de conditionnement instrumental).

Le conditionnement instrumental

- Présence d'un **SC**
- Obtention d'une **RC** de l'animal
- Renforcement négatif ou positif



Le conditionnement instrumental

Le principal modèle du conditionnement opérant, est celui proposé par Sutton et Barto [Bar81b]. La fonction d'activation des neurones utilisés dans leur modèle est la suivante (voir figure 17) :

$$O_j = H(\sum_i W_{ij} \cdot I_i + bruit) \qquad (2)$$

Avec $H(x)$ la fonction de Heaviside définie par :

$$H(x) = \begin{cases} 1 & si \;\; x > 0 \\ 0 & sinon \end{cases}$$

$$W_{ij}[t+1] = W_{ij}[t] + \epsilon \cdot [R[t] - R[t-1]] \cdot [O_j[t-1] - O_j[t-2]] \cdot I_i[t-1]$$

Où $R[t]$ est la récompense donnée à l'instant t.
On peut aussi écrire cette équation sous une forme différentielle :

$$\frac{\partial W_{ij}(t+1)}{\partial t} = \epsilon \cdot \frac{\partial R(t)}{\partial t} \cdot \frac{\partial O_j(t-1)}{\partial t} \cdot I_i(t-1)$$

Le conditionnement instrumental

Le principal modèle du conditionnement opérant, est celui proposé par Sutton et Barto [Bar81b]. La fonction d'activation des neurones utilisés dans leur modèle est la suivante (voir figure 17) :

$$O_j = H(\sum_i W_{ij} \cdot I_i + bruit) \qquad (2)$$

Avec $H(x)$ la fonction de Heaviside définie par :

$$H(x) = \begin{cases} 1 & si \;\; x > 0 \\ 0 & sinon \end{cases}$$

$$W_{ij}[t+1] = W_{ij}[t] + \epsilon \cdot [R[t] - R[t-1]] \cdot [O_j[t-1] - O_j[t-2]] \cdot I_i[t-1]$$

Où $R[t]$ est la récompense donnée à l'instant t.
On peut aussi écrire cette équation sous une forme différentielle :

$$\frac{\partial W_{ij}(t+1)}{\partial t} = \epsilon \cdot \frac{\partial R(t)}{\partial t} \cdot \frac{\partial O_j(t-1)}{\partial t} \cdot I_i(t-1)$$

Le conditionnement instrumental

Le principal modèle du conditionnement opérant, est celui proposé par Sutton et Barto [Bar81b]. La fonction d'activation des neurones utilisés dans leur modèle est la suivante (voir figure 17) :

$$O_j = H(\sum_i W_{ij} \cdot I_i + bruit) \qquad (2)$$

Avec $H(x)$ la fonction de Heaviside définie par :

$$H(x) = \begin{cases} 1 & si \;\; x > 0 \\ 0 & sinon \end{cases}$$

$$W_{ij}[t+1] = W_{ij}[t] + \epsilon \cdot [R[t] - R[t-1]] \cdot [O_j[t-1] - O_j[t-2]] \cdot I_i[t-1]$$

Où $R[t]$ est la récompense donnée à l'instant t.
On peut aussi écrire cette équation sous une forme différentielle :

$$\frac{\partial W_{ij}(t+1)}{\partial t} = \epsilon \cdot \frac{\partial R(t)}{\partial t} \cdot \frac{\partial O_j(t-1)}{\partial t} \cdot I_i(t-1)$$

"Bruit sur les actions"
Permet de proposer aléatoirement une sortie même si I=0
(pas d'entrée)

Assure une exploration aléatoire de l'environnement

Le conditionnement instrumental

Le principal modèle du conditionnement opérant, est celui proposé par Sutton et Barto [Bar81b]. La fonction d'activation des neurones utilisés dans leur modèle est la suivante (voir figure 17) :

$$O_j = H(\sum_i W_{ij} \cdot I_i + bruit) \qquad (2)$$

Avec $H(x)$ la fonction de Heaviside définie par :

$$H(x) = \begin{cases} 1 & si \ \ x > 0 \\ 0 & sinon \end{cases}$$

$$W_{ij}[t+1] = W_{ij}[t] + \epsilon \cdot [R[t] - R[t-1]] \cdot [O_j[t-1] - O_j[t-2]] \cdot I_i[t-1]$$

Variation de la récompense

Variation de la réponse

Où $R[t]$ est la récompense donnée à l'instant t.
On peut aussi écrire cette équation sous une forme différentielle :

$$\frac{\partial W_{ij}(t+1)}{\partial t} = \epsilon \cdot \frac{\partial R(t)}{\partial t} \cdot \frac{\partial O_j(t-1)}{\partial t} \cdot I_i(t-1)$$

Le conditionnement instrumental

Variation du renforcement	Variation de la sortie	Entrée	dW
↗	↗	Active	↗
↘	↗	Active	↘
0	↘ ou ↗	Active	0
↘ ou ↗	0	Active	0
↘ ou ↗	↘ ou ↗	Inactive	0

Tableau 1. *Récapitulation du fonctionnement du modèle de Sutton et Barto. dW : variation du poids. Entrée : Active≡ activité≠ 0; Inactive≡ activité= 0*

Le conditionnement instrumental

points forts :
- modèle simple pour une combinaison exploration/exploitation

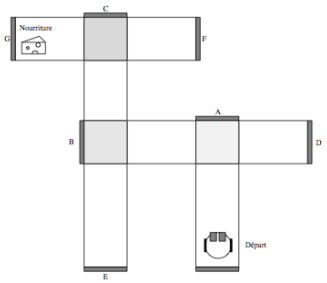
- l'exploration est assurée par un niveau de bruit sur les sorties, présent quand il n'y a pas d'entrées.

-rend compte des expériences de conditionnement opérant. association par obtention d'une récompense en l'absence de US.

Mais :

- difficile de gérer efficacement le niveau de bruit une fois les associations apprises (bruit plus fort que l'apprentissage ? NON. mais alors comment ré-explorer ?).

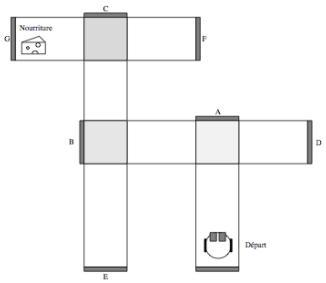
- inefficace si la récompense est **retardée ou différée**.



Apprentissage par renforcement

la récompense est **retardée ou différée**.

comment rétro-propager la récompense sur toutes les actions qui ont permis d'obtenir r ?



Apprentissage par renforcement

Journal of Artificial Intelligence Research 4 (1996) 237-285 Submitted 9/95; published 5/96

Reinforcement Learning: A Survey

Leslie Pack Kaelbling LPK@CS.BROWN.EDU
Michael L. Littman MLITTMAN@CS.BROWN.EDU
Computer Science Department, Box 1910, Brown University
Providence, RI 02912-1910 USA

Andrew W. Moore AWM@CS.CMU.EDU
Smith Hall 221, Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213 USA

Abstract

This paper surveys the field of reinforcement learning from a computer-science perspective. It is written to be accessible to researchers familiar with machine learning. Both the historical basis of the field and a broad selection of current work are summarized. Reinforcement learning is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment. The work described here has a resemblance to work in psychology, but differs considerably in the details and in the use of the word "reinforcement." The paper discusses central issues of reinforcement learning,

Apprentissage par renforcement

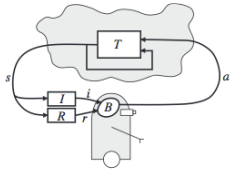


Figure 1: The standard reinforcement-learning model.

In the standard reinforcement-learning model, an agent is connected to its environment via perception and action, as depicted in Figure 1. On each step of interaction the agent receives as input, i, some indication of the current state, s, of the environment; the agent then chooses an action, a, to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar *reinforcement signal*, r. The agent's behavior, B, should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms that are the subject of later sections of this paper.

Apprentissage par renforcement

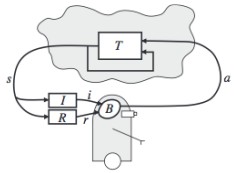


Figure 1: The standard reinforcement-learning model.

Formally, the model consists of

- a discrete set of environment states, $\mathcal{S}$;
- a discrete set of agent actions, $\mathcal{A}$; and
- a set of scalar reinforcement signals; typically $\{0, 1\}$, or the real numbers.

Apprentissage par renforcement

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.

The agent's job is to find a policy π, mapping states to actions, that maximizes some long-run measure of reinforcement. We expect, in general, that the environment will be non-deterministic; that is, that taking the same action in the same state on two different occasions may result in different next states and/or different reinforcement values. This happens in our example above: from state 65, applying action 2 produces differing reinforcements and differing states on two occasions. However, we assume the environment is stationary; that is, that the *probabilities* of making state transitions or receiving specific reinforcement signals do not change over time.[1]

comportement optimal

The *finite-horizon* model is the easiest to think about; at a given moment in time, the agent should optimize its expected reward for the next h steps:

$$E(\sum_{t=0}^{h} r_t) \ ;$$

récompense reçue à l'instant t

The infinite-horizon discounted model takes the long-run reward of the agent into account, but rewards that are received in the future are geometrically discounted according to discount factor γ, (where $0 \leq \gamma < 1$):

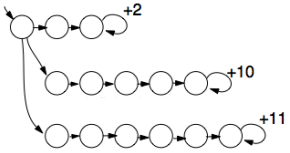
$$E(\sum_{t=0}^{\infty} \gamma^t r_t) \ .$$

plus on regarde dans le futur (plus t est grand) et plus la récompense estimée est décotée

Another optimality criterion is the *average-reward model*, in which the agent is supposed to take actions that optimize its long-run average reward:

$$\lim_{h \to \infty} E(\frac{1}{h} \sum_{t=0}^{h} r_t) \ .$$

comportement optimal



Soit les trois modèles :

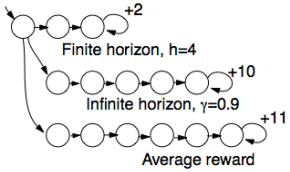
Finite horizon, h=4

Infinite horizon, gamma = 0.9

Average reward (h très grand)

quels choix vont ils faire ? si on change h à 1000 et gamma à 0.2 ?

comportement optimal



Soit les trois modèles :

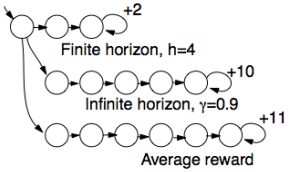
Finite horizon, h=4

Infinite horizon, gamma = 0.9

Average reward (h très grand)

quels choix vont ils faire ? si on change h à 1000 et gamma à 0.2 ?

comportement optimal



r = 0 + 2 = 2

r = 0

r=0

Soit les trois modèles :

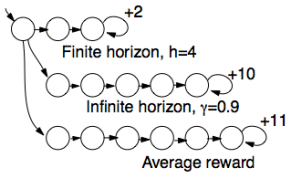
Finite horizon, h=4

Infinite horizon, gamma = 0.9

Average reward (h très grand)

quels choix vont ils faire ? si on change h à 1000 et gamma à 0.2 ?

comportement optimal



r = 0.9 x0 + 0.81 X 2 = 1.62

r = 0 ..... + 0.59 x10 = 5.9

r=0 .....+0.53 x11 = 5.85

Soit les trois modèles :

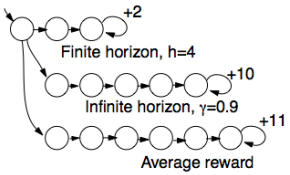
Finite horizon, h=4

Infinite horizon, gamma = 0.9

Average reward (h très grand)

quels choix vont ils faire ? si on change h à 1000 et gamma à 0.2 ?

comportement optimal



r = 2/h

r = 10/h

r= 11/h

Soit les trois modèles :

Finite horizon, h=4

Infinite horizon, gamma = 0.9

Average reward (h très grand)

quels choix vont ils faire ? si on change h à 1000 et gamma à 0.2 ?

comportement optimal

One major difference between reinforcement learning and supervised learning is that a reinforcement-learner must explicitly explore its environment. In order to highlight the problems of exploration, we treat a very simple case in this section. The fundamental issues and approaches described here will, in many cases, transfer to the more complex instances of reinforcement learning discussed later in the paper.

The simplest possible reinforcement-learning problem is known as the k-armed bandit problem, which has been the subject of a great deal of study in the statistics and applied mathematics literature (Berry & Fristedt, 1985). The agent is in a room with a collection of k gambling machines (each called a "one-armed bandit" in colloquial English). The agent is permitted a fixed number of pulls, h. Any arm may be pulled on each turn. The machines do not require a deposit to play; the only cost is in wasting a pull playing a suboptimal machine. When arm i is pulled, machine i pays off 1 or 0, according to some underlying probability parameter p_i, where payoffs are independent events and the p_is are unknown. What should the agent's strategy be?

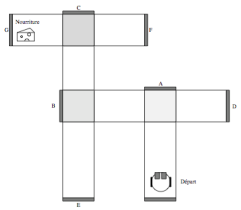
This problem illustrates the fundamental tradeoff between exploitation and exploration.

Delayed reward

Problems with delayed reinforcement are well modeled as *Markov decision processes* (MDPs). An MDP consists of

- a set of states $\mathcal{S}$,
- a set of actions $\mathcal{A}$.
- a reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \Re$, and
- a state transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$, where a member of $\Pi(\mathcal{S})$ is a probability distribution over the set $\mathcal{S}$ (i.e. it maps states to probabilities). We write $T(s, a, s')$ for the probability of making a transition from state s to state s' using action a.

Delayed reward



TD(0)
s l'état courrant
s' l'état dans lequel on se trouve après avoir effectué l'action a
r le renforcement instantané
gamma : discount factor
alpha : rapport présent/futur (learning rate, doit diminuer progressivement au cours de l'exploration)

$$V(s) := V(s) + \alpha(r + \gamma V(s') - V(s)) \ .$$

la mise a jour suite au passage de s dans s' (alpha)

la valeur du passé (1-alpha)

Delayed reward

TD(0)
s l'état courrant
s' l'état dans lequel on se trouve après avoir effectué l'action a
r le renforcement instantané
gamma : discount factor
alpha : rapport présent/futur (learning rate, doit diminuer progressivement au cours de l'exploration)

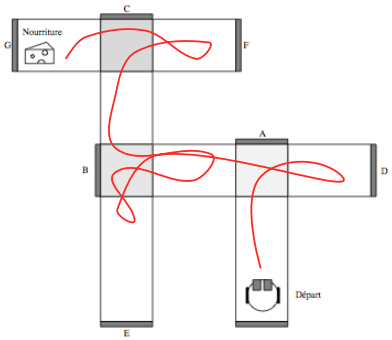
$$V(s) := V(s) + \alpha(r + \gamma V(s') - V(s)) \ .$$

la mise a jour suite au passage de s dan s' (alpha)

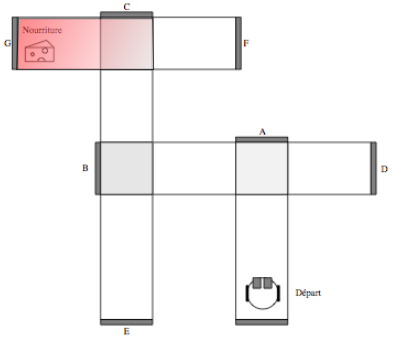
la valeur du passé (1-alpha)

notez que a n'intervient pas dans cette équation

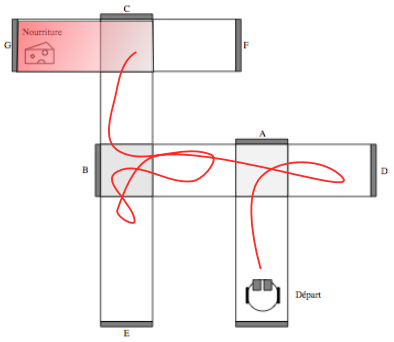
Delayed reward



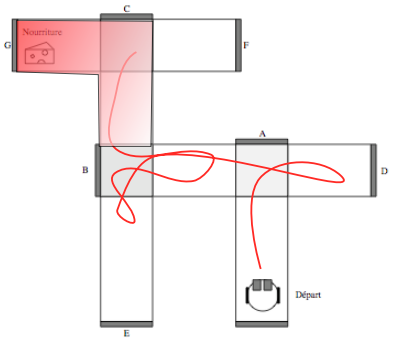
Delayed reward



Delayed reward



Delayed reward



$$V(s) := V(s) + \alpha(r + \gamma V(s') - V(s)) \ .$$

exploration après exploration : construction progressif d'un gradient de récompense pour chaque état u

Delayed reward

décote temporelle de k

est elligible si s était l'état visité à l'instant k

TD(lambda), avec lambda [0,1]

The $TD(0)$ rule as presented above is really an instance of a more general class of algorithms called $TD(\lambda)$, with $\lambda = 0$. $TD(0)$ looks only one step ahead when adjusting value estimates; although it will eventually arrive at the correct answer, it can take quite a while to do so. The general $TD(\lambda)$ rule is similar to the $TD(0)$ rule given above,

$$V(u) := V(u) + \alpha(r + \gamma V(s') - V(s))e(u) \ ,$$

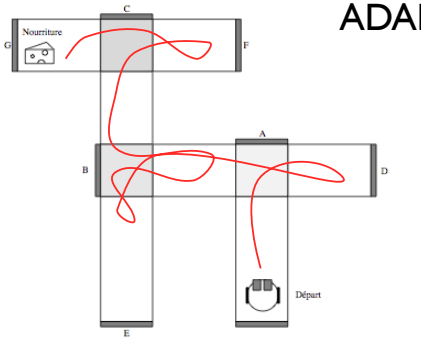
but it is applied to *every state* according to its eligibility $e(u)$, rather than just to the immediately previous state, s. One version of the eligibility trace is defined to be

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s,s_k} \ , \text{ where } \delta_{s,s_k} = \begin{cases} 1 \text{ if } s = s_k \\ 0 \text{ otherwise} \end{cases} \ .$$

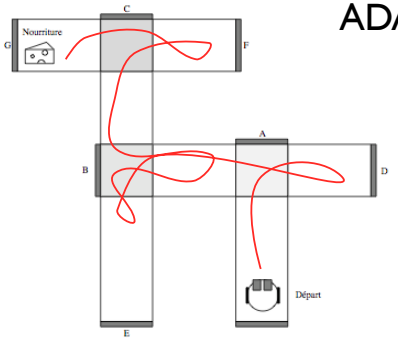
The eligibility of a state s is the degree to which it has been visited in the recent past; when a reinforcement is received, it is used to update all the states that have been recently visited, according to their eligibility. When $\lambda = 0$ this is equivalent to $TD(0)$. When $\lambda = 1$, it is roughly equivalent to updating all the states according to the number of times they were visited by the end of a run.

Delayed reward

ADABEBCFCG

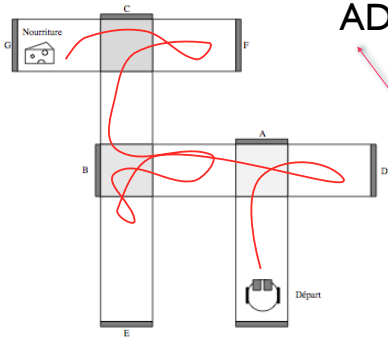


Delayed reward



ADABEBCFCG

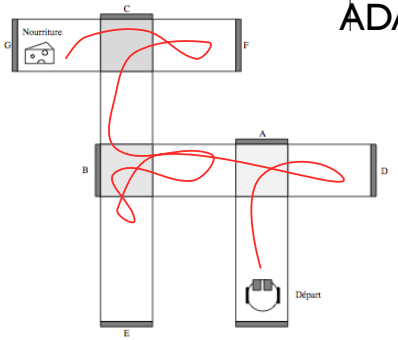
Delayed reward

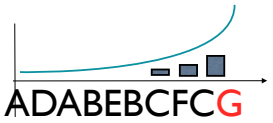


ADABEBCFCG

$$V(u) := V(u) + \alpha(r + \gamma V(s') - V(s))e(u)$$

Delayed reward





ADABEBCFCG

Delayed reward

Optimal

mais quand lambda ->1 : très couteux en calculs (maintenir une mémoire la liste des états elligibles à chaque action + calculs).

Delayed reward

TD(0)
s l'état courrant
s' l'état dans lequel on se trouve après avoir effectué l'action a
r le renforcement instantané
gamma : discount factor
alpha : rapport présent/futur (learning rate, doit diminuer progressivement au cours de l'exploration)

$$V(s) := V(s) + \alpha(r + \gamma V(s') - V(s)) \ .$$

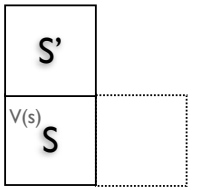
la mise a jour suite au passage de s dan s' (alpha)

la valeur du passé (1-alpha)

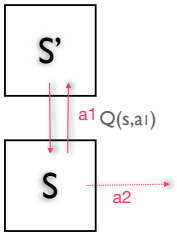
notez que a n'intervient pas dans cette équation

Qlearning

TD(lambda)



Qlearning



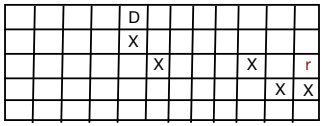
Qlearning

The Q learning rule is

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \ ,$$

Qlearning

Soit l'environnement suivant :



où X est un mur, S la case de départ, r la case à atteindre et D une case dangereuse.
Un robot peut exécuter les actions haut, bas, droite, gauche sur chaque case (correspondant à des états) et arrive sur la case correspondante si possible, sinon, il reste sur la même case.
Toutes les valeurs des arcs sont à 1 initialement.
La récompense sur l'état but r est de 1000.
La récompense sur la zone de danger D est de -5.

Qlearning

Implémentez l'algo Q-Learning à l'aide du langage de votre choix.
Rappel de l'algo :
1. Initialiser la table des poids des arcs :
2.$\forall s \forall a Q(s,a) \leftarrow 0$
3. pour n boucles d'apprentissage
4...$\lambda \leftarrow 1; \epsilon \leftarrow 1;$
5...etatCourant←etatinitial
6...Pour n' itérations (n' = nombre supposé max d'états possibles)
7.....s←etatCourant
8.....nb←random(0,1)
9..... Si nb<ε Alors
10.....Choisir aléatoirement une action a
11.....Sinon
12.....Choisir action la plus intéressante :
13.....a←argMax$_{a}$($Q(s,a')$)
14..... Fin Si
15..... Effectuer action a :
16.....s'←nouvelEtatCourant
17..... calculer Q(s,a)
18.....$Q(s,a) \leftarrow \lambda \times (r + \gamma \times max_{a'}(Q(s',a')) + (1-\lambda) \times Q(s,a)$

19..... **decrémenter** les coefficients d'apprentissage et d'exploration :
20.....$\lambda \leftarrow 0.99 \times \lambda$
21.....$\epsilon \leftarrow 0.99 \times \epsilon$
22..... Si but atteint, casser la boucle n' (uniquement celle là)
22...Fin pour n' itérations
23. Fin pour n boucles d'apprentissage
.
A la fin d'une boucle, afficher le chemin trouvé (liste des actions les plus intéressantes à partir de l'état initial).

Qlearning

Implémentez l'algo Q-Learning à l'aide du langage de votre choix.
Rappel de l'algo :
1. Initialiser la table des poids des arcs :
2.$\forall s \forall a Q(s,a) \leftarrow 0$
3. pour n boucles d'apprentissage
4...$\lambda \leftarrow 1; \epsilon \leftarrow 1;$
5...etatCourant←etatinitial
6...Pour n' itérations (n' = nombre supposé max d'états possibles)
7.....s←etatCourant
8.....nb←random(0,1)
9..... Si nb<ε Alors
10.....Choisir aléatoirement une action a
11.....Sinon
12.....Choisir action la plus intéressante :
13.....a←argMax$_{a'}$($Q(s,a')$)
14..... Fin Si
15..... Effectuer action a :
16.....s'←nouvelEtatCourant
17..... calculer Q(s,a)
18.....$Q(s,a) \leftarrow \lambda \times (r + \gamma \times max_{a'}(Q(s',a')) + (1-\lambda) \times Q(s,a)$

exploration

exploitation (greedy)

19..... **decrémenter** les coefficients d'apprentissage et d'exploration :
20.....$\lambda \leftarrow 0.99 \times \lambda$
21.....$\epsilon \leftarrow 0.99 \times \epsilon$
22..... Si but atteint, casser la boucle n' (uniquement celle là)
22...Fin pour n' itérations
23. Fin pour n boucles d'apprentissage
.
A la fin d'une boucle, afficher le chemin trouvé (liste des actions les plus intéressantes à partir de l'état initial).

Qlearning

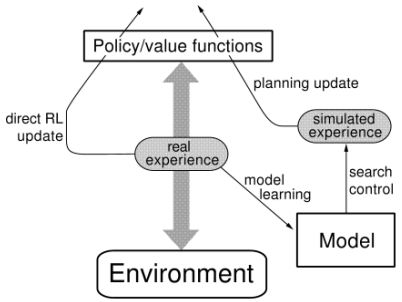
Limitations ?

- nécessite beaucoup d'explorations avant de converger sur l'ensemble de l'environnement (pensez a un robot dont chaque action de déplacement prends du temps)

- couteux en temps de calcul et en temps réel (pensez a un robot donc chaque action de déplacement prends du temps)

- ne s'applique bien que si l'environnement peut être modélisé en couples (états,actions)

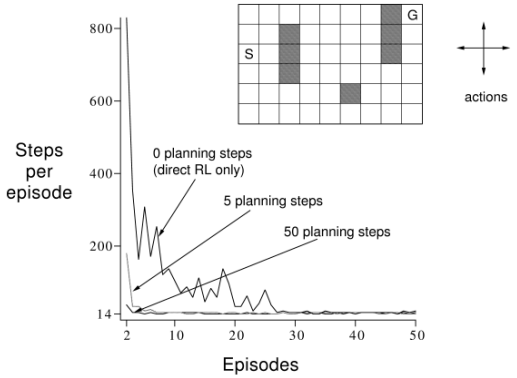
Dyna-Q



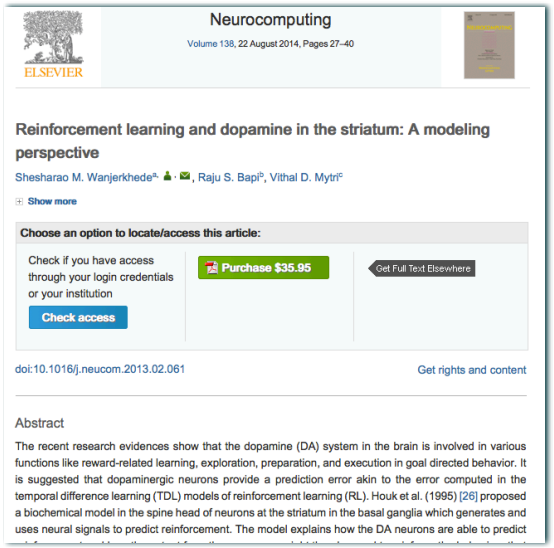
Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
- (a) $s \leftarrow$ current (nonterminal) state
- (b) $a \leftarrow \varepsilon$-greedy(s,Q)
- (c) Execute action a; observe resultant state, s', and reward, r
- (d) $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
- (e) $Model(s,a) \leftarrow s', r$ (assuming deterministic environment)
- (f) Repeat N times:
  - $s \leftarrow$ random previously observed state
  - $a \leftarrow$ random action previously taken in s
  - $s', r \leftarrow Model(s,a)$
  - $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$

Dyna-Q



Dyna-Q



Neurocomputing

Volume 138, 22 August 2014, Pages 27–40

ELSEVIER

Reinforcement learning and dopamine in the striatum: A modeling perspective

Shesharao M. Wanjerkhede[a], Raju S. Bapi[b], Vithal D. Mytri[c]

Show more

Choose an option to locate/access this article:

Check if you have access through your login credentials or your institution

Check access

Purchase $35.95

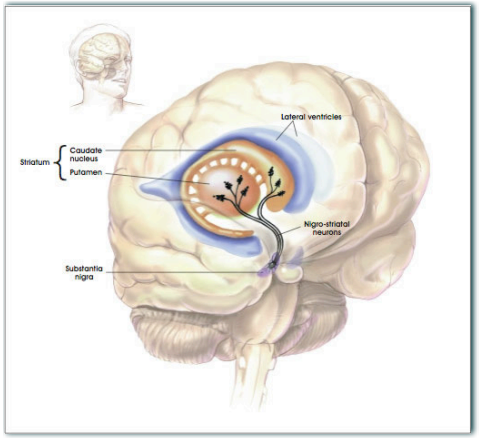
Get Full Text Elsewhere

doi:10.1016/j.neucom.2013.02.061

Get rights and content

Abstract

The recent research evidences show that the dopamine (DA) system in the brain is involved in various functions like reward-related learning, exploration, preparation, and execution in goal directed behavior. It is suggested that dopaminergic neurons provide a prediction error akin to the error computed in the temporal difference learning (TDL) models of reinforcement learning (RL). Houk et al. (1995) [26] proposed a biochemical model in the spine head of neurons at the striatum in the basal ganglia which generates and uses neural signals to predict reinforcement. The model explains how the DA neurons are able to predict

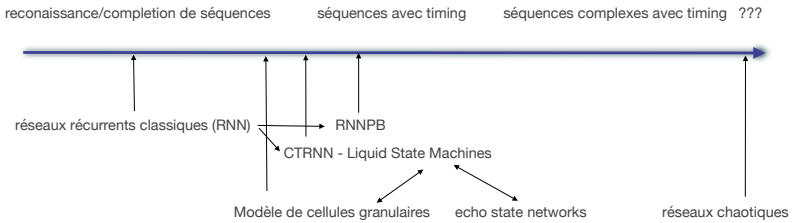


Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement

Apprentissage de séquences temporelles

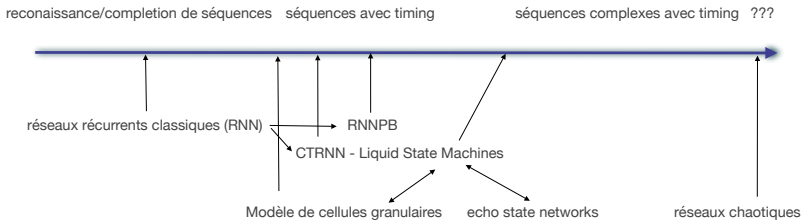


Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement

Apprentissage de séquences temporelles



Recurrent Neural Networks

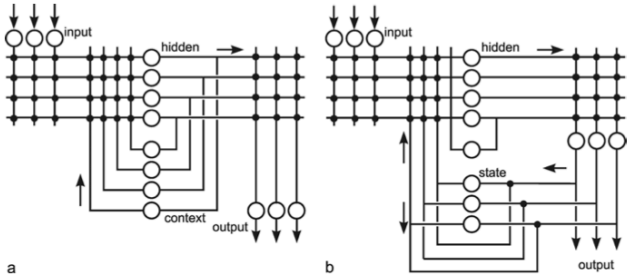
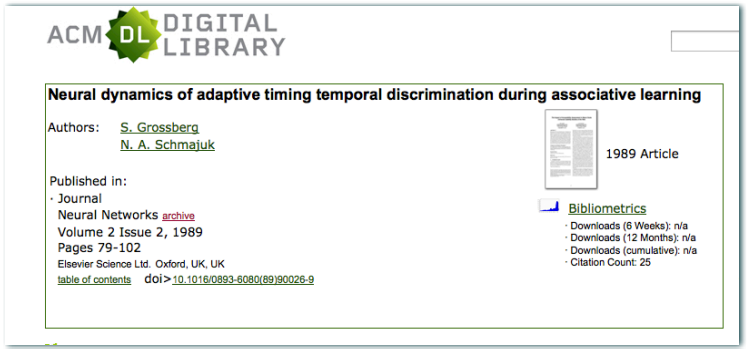


Fig. 14.1 **Two networks with partial recurrent connections.** (a) Elman net (b) Jordan net. Only the weights in the feedforward channels can be modified. The arrows show the direction of the information flow. These networks can be used to learn temporal sequences, i. e., to provide an output vector which depends on the temporal order of the input vectors. Conversely, for a given input vector, a series of output vectors can be produced

timing & neural network



ACM DL DIGITAL LIBRARY

Neural dynamics of adaptive timing temporal discrimination during associative learning

Authors: S. Grossberg
N. A. Schmajuk

1989 Article

Published in:
· Journal
Neural Networks archive
Volume 2 Issue 2, 1989
Pages 79-102
Elsevier Science Ltd. Oxford, UK, UK
table of contents doi>10.1016/0893-6080(89)90026-9

Bibliometrics
· Downloads (6 Weeks): n/a
· Downloads (12 Months): n/a
· Downloads (cumulative): n/a
· Citation Count: 25

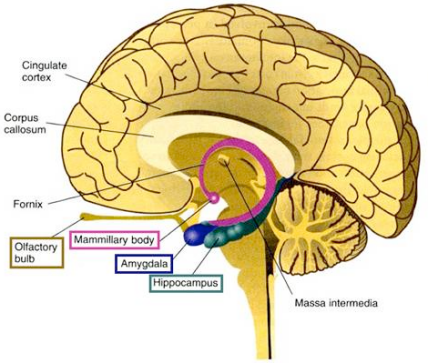
timing & neural network

Learning and communication via imitation:
an autonomous robot perspective

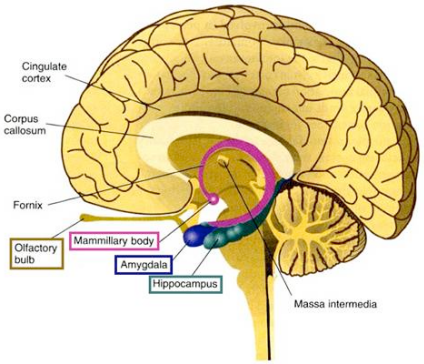
P. Andry, P. Gaussier, S. Moga, J. P. Banquet[1], J. Nadel[2]
Neurocybernetic team, ETIS Lab, UPRES A 8051 UCP-ENSEA
ENSEA, 6 avenue du Ponceau, 95014 Cergy, France
andry@ensea.fr, gaussier@ensea.fr
[1]Neurosciences et modélisation INSERM U483 Paris.
[2]Equipe Développement et Psychopathologie UMR CNRS 7593
Hopital de la Salpétrière, Pavillon Clérambault
54 bd de l'Hopital, 75005 Paris, France

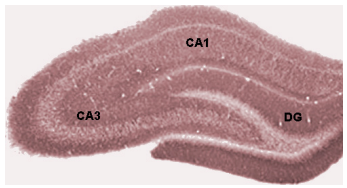
Abstract— **This paper proposes a neural network architecture designed to exhibit learning and communication capabilities via imitation. Our architecture allows a "proto imitation" behavior using the "perception ambiguity" inherent to real environments. In the perspective of turn-taking and gestural communication between two agents, new experiments on movement synchronization in an interaction game are presented. Synchronization is obtained as a global at-**

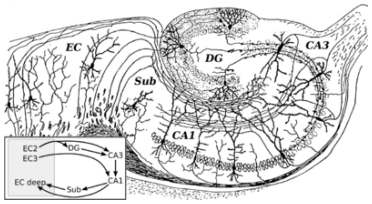
timing & neural network



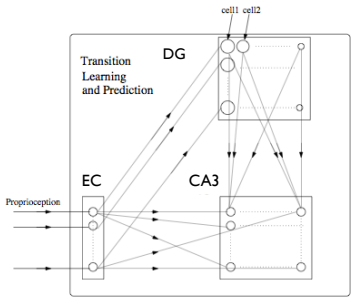
timing & neural network



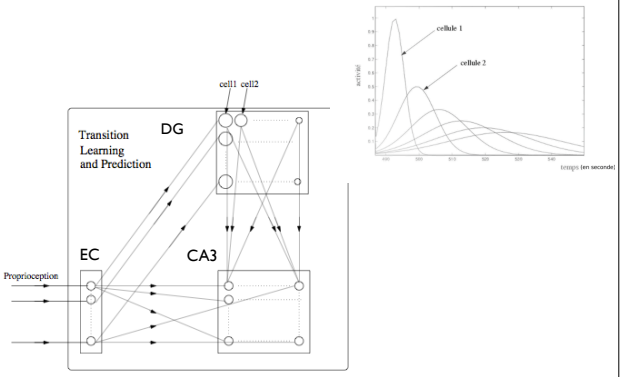




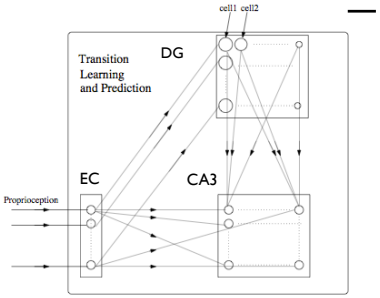
timing & neural network



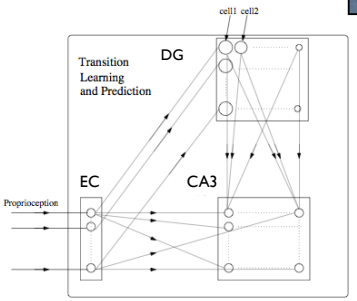
timing & neural network



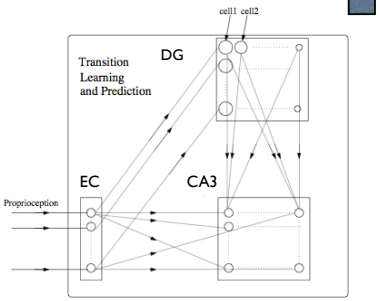
timing & neural network



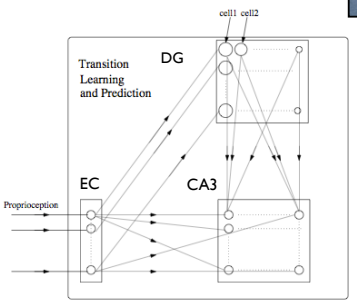
timing & neural network



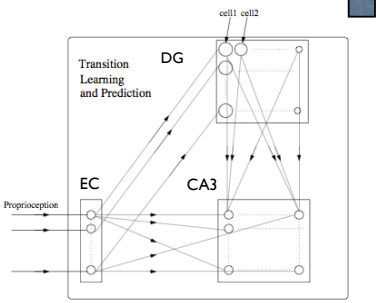
timing & neural network



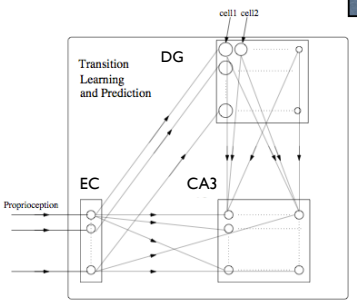
timing & neural network



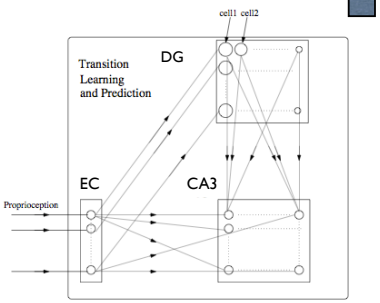
timing & neural network



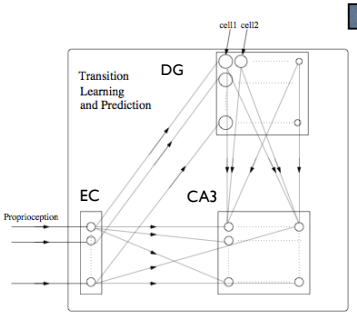
timing & neural network



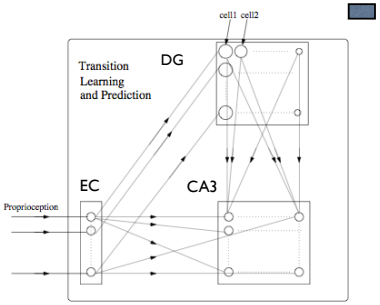
timing & neural network



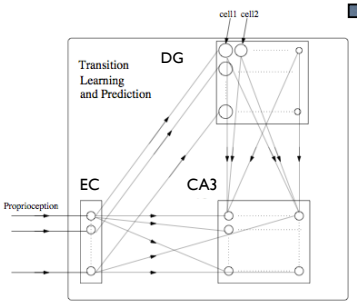
timing & neural network



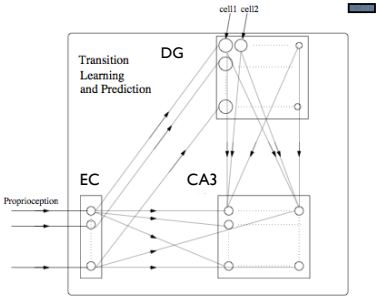
timing & neural network



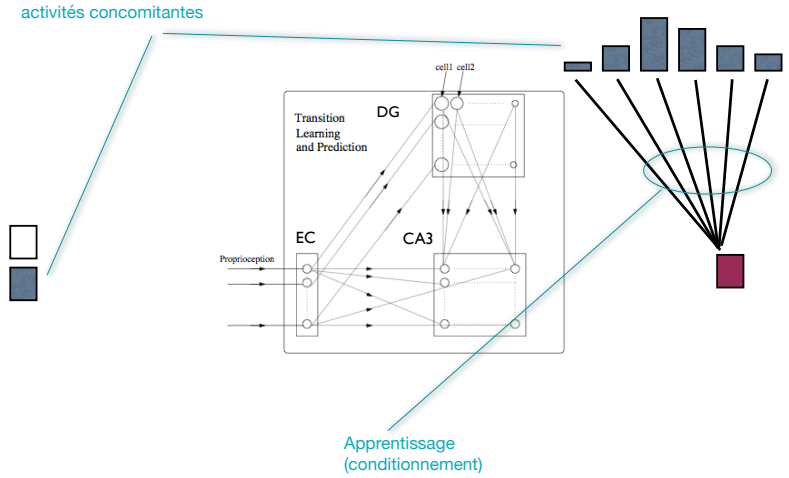
timing & neural network



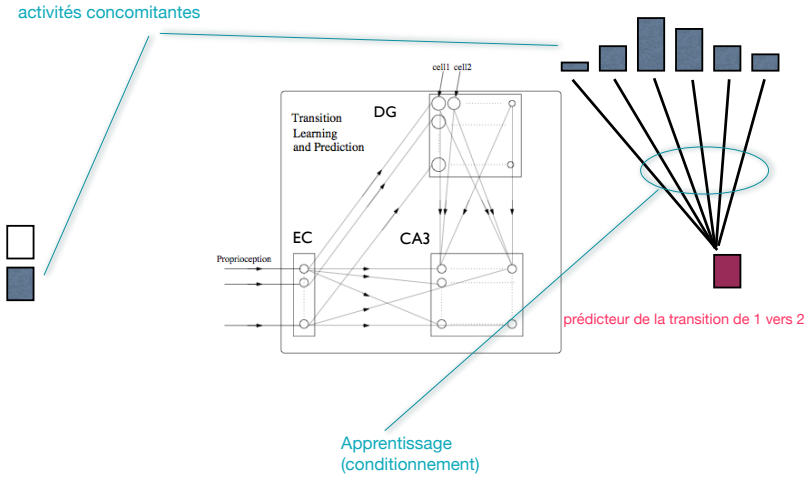
timing & neural network



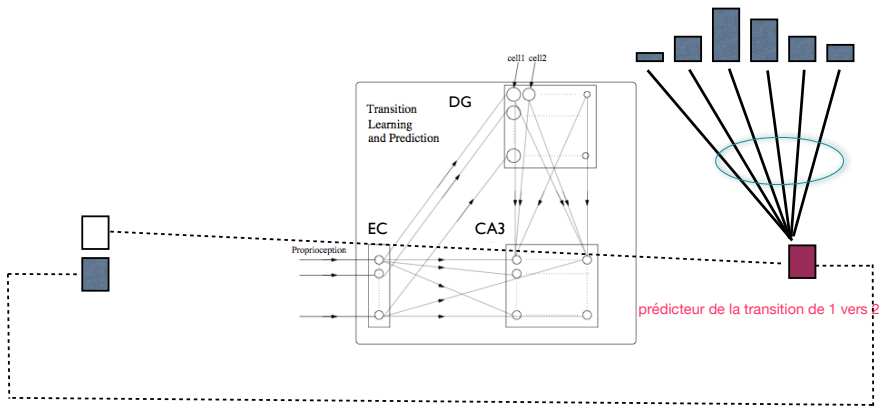
timing & neural network



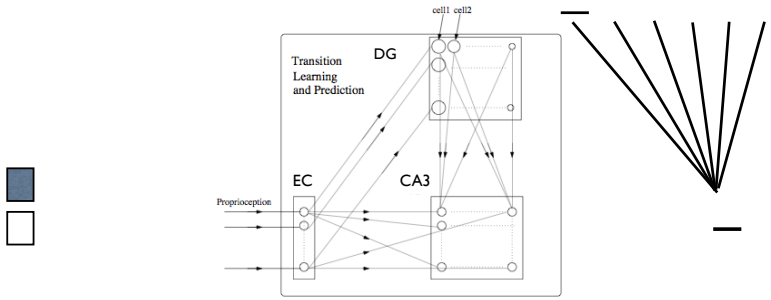
timing & neural network



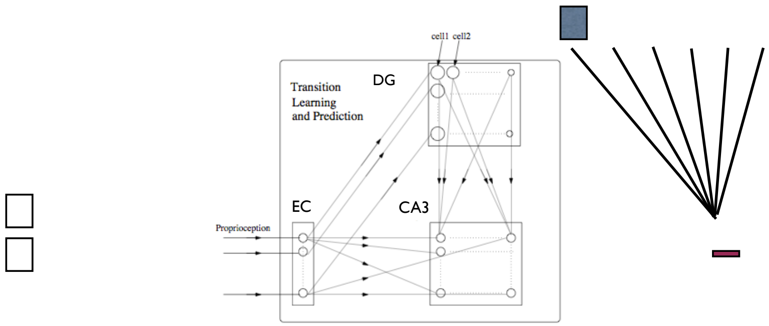
timing & neural network



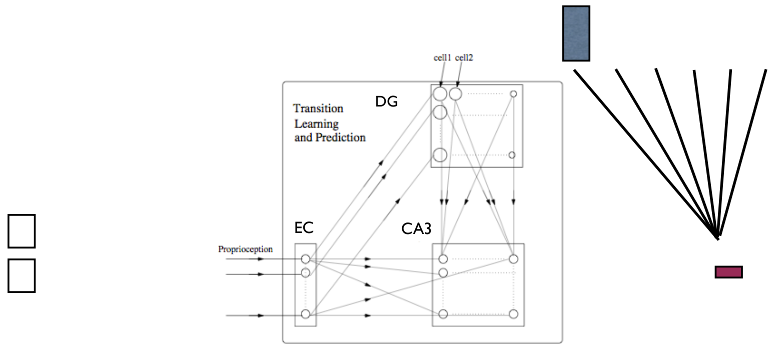
timing & neural network



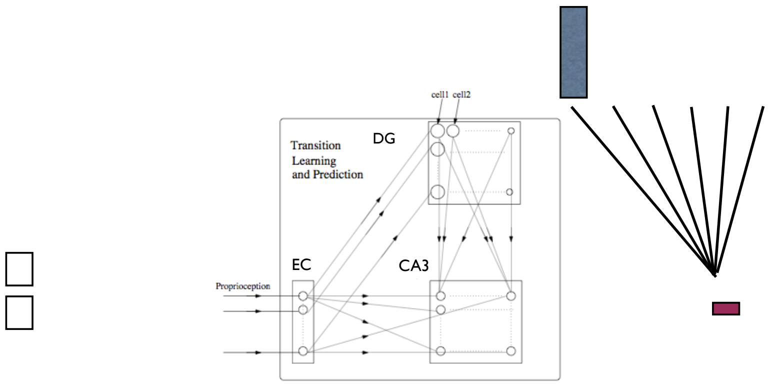
timing & neural network



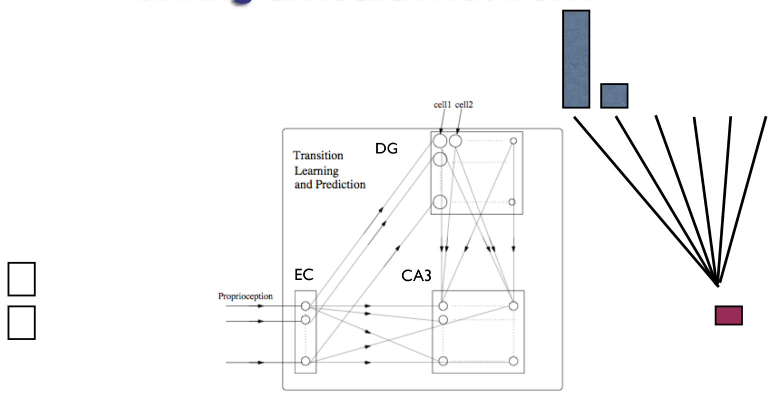
timing & neural network



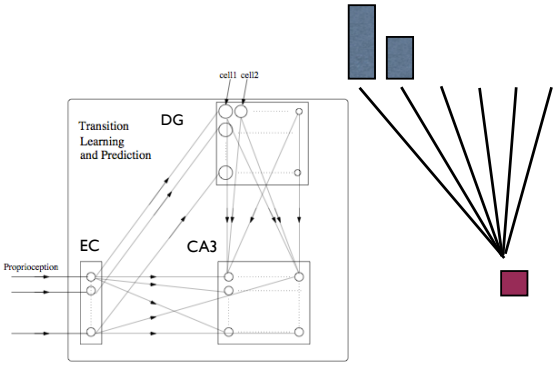
timing & neural network



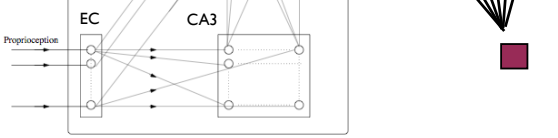
timing & neural network

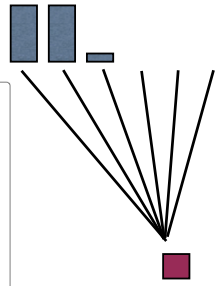


timing & neural network

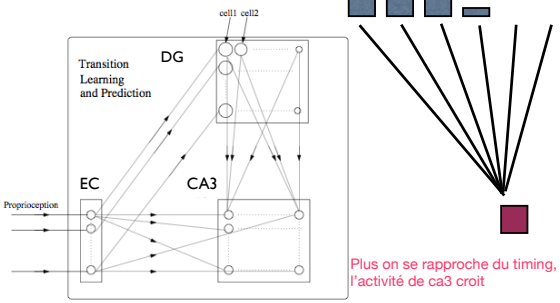


timing & neural network



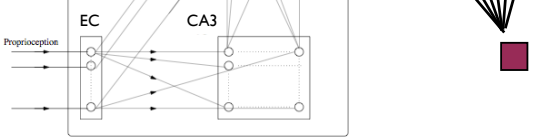


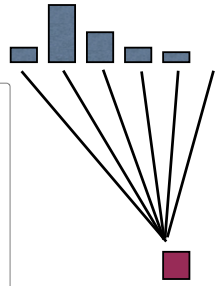
timing & neural network



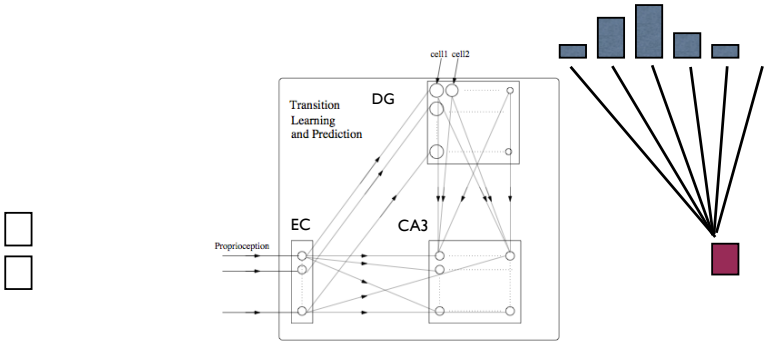
Plus on se rapproche du timing, plus l'activité de ca3 croit

timing & neural network

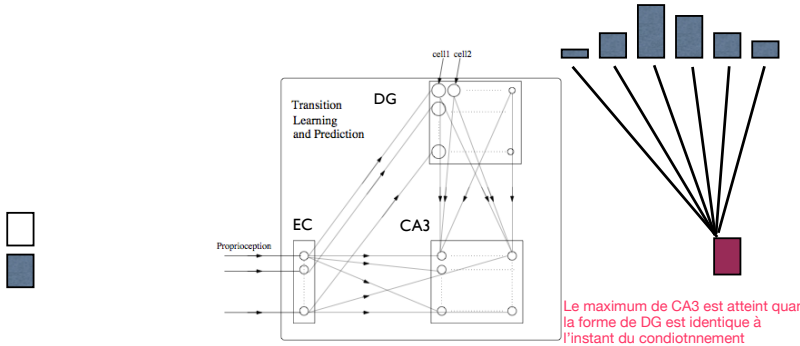




timing & neural network

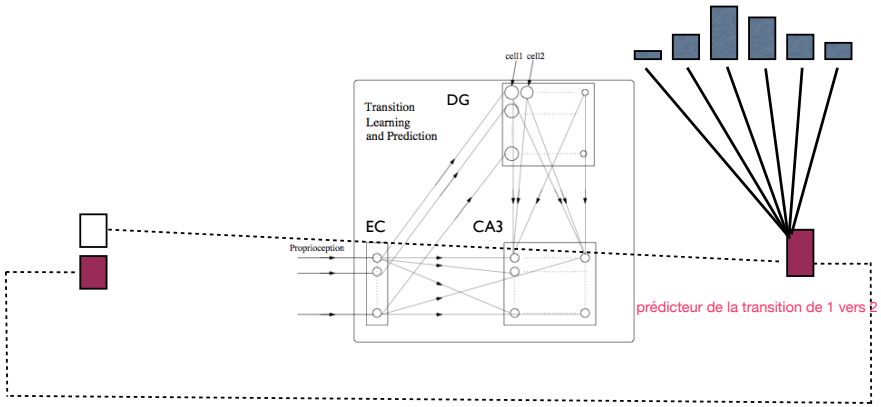


timing & neural network



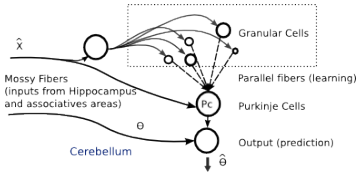
Le maximum de CA3 est atteint quand la forme de DG est identique à l'instant du condiotnnement

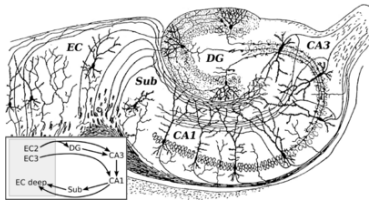
timing & neural network



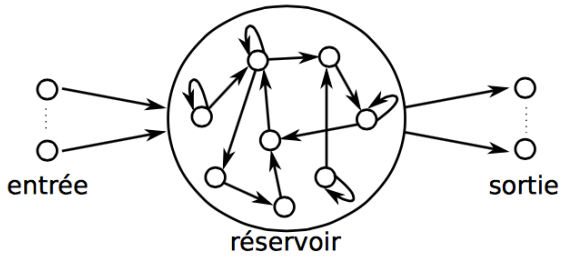
prédicteur de la transition de 1 vers 2

timing & neural network

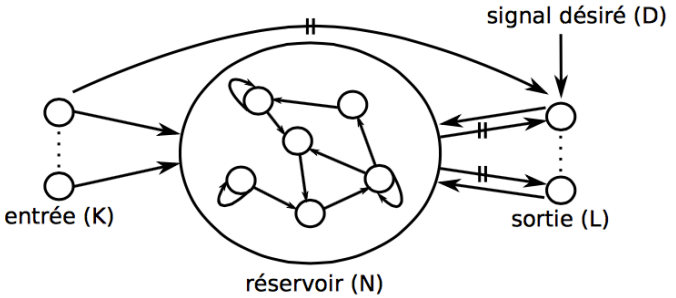




Echo State Networks



Echo State Networks



Echo State Networks

Les activités des neurones du réservoir de dynamiques sont mises à jour suivant l'équation

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + W^{fb}y(n))$$

avec $x(n)$ l'état du réservoir de dimension N au temps n. $u(n)$ est l'état de la couche d'entrée de dimension K. $y(n)$ est l'état de la couche de sortie de dimension L. f une fonction sigmoïde. W est la matrice NxN des poids des connexions entre les neurones du réservoir. W^{in} est la matrice KxN des poids des connexions entre la couche d'entrée et le réservoir. W^{fb} est la matrice NxL des poids des connexions entre la couche de sortie est le réservoir.

Echo State Networks

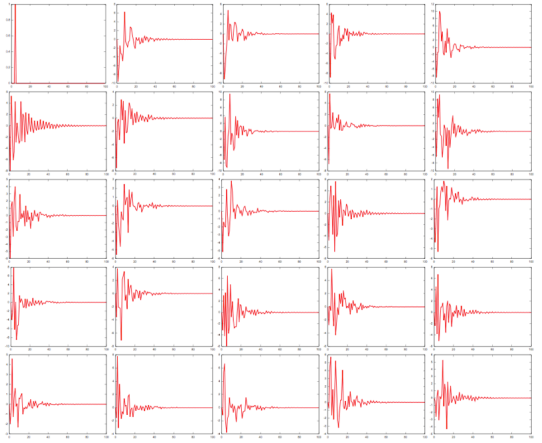
L'apprentissage est réalisé sur les connexions entre le réservoir et la couche de sortie. Toutes les autres connexions ont des poids non modifiables et choisis aléatoirement. L'apprentissage est réalisé hors ligne.

Echo State Networks

La valeur des connexions internes du réservoir ont été reprises du rapport technique [Jaeger, 2001]. Il est nécessaire que ces poids vérifient des propriétés très précises pour que la dynamique du réseau soit suffisament riche. Le réservoir est composé de 400 neurones. La matrice des poids comporte donc $400 * 400 = 1600$ poids. 98.75% des connexions ont un poids à 0, 0.625% des connexions ont un poids de 0.4 et 0.625% des connexions ont un poids de -0.4. La distribution de ces poids est faites aléatoirement et reste la même pour les phases d'apprentissages et de reproduction de séquences. La séquence étant un signal "analogique" - dont les états sont des activités variant entre 0 et 1 - la couche de sortie ne contient qu'un seul neurone. Par conséquent, entre la couche de sortie et le réservoir il y a 1×400 poids qui sont assignés aléatoirement avec des poids allant de -2 à 2.

Echo State Networks

Dans les ESNs, le réservoir de dynamiques joue le rôle de mémoire à court terme. On se rend compte de cette propriété en observant les activités des neurones du réservoir lorsque l'on applique un signal impulsionnel en entrée (figure 3.17).



Echo State Networks

Lors de la phase d'apprentissage, il est indispensable de répéter un nombre suffisant de fois la séquence que l'on souhaite faire apprendre. Ceci pour deux raisons : la première, pour que les dynamiques du réservoir aient le temps de converger vers un état stable. La seconde pour que l'apprentissage soit stable. Par exemple, pour une séquence simple de deux états comme "0 1" il faut environ 50 itérations avant que les dynamiques du réservoir se stabilisent sur un état stable

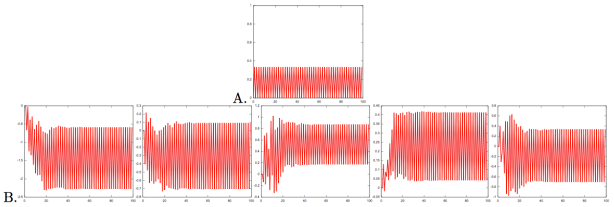


FIG. 3.18: A) Activité du signal donné à l'architecture. B) Activités de 5 neurones du réservoir de dynamiques choisis arbitrairement. On observe que les dynamiques mettent environ 50 itérations avant de converger vers un état stable.

Echo State Networks

Lors de la phase d'apprentissage, il est indispensable de répéter un nombre suffisant de fois la séquence que l'on souhaite faire apprendre. Ceci pour deux raisons : la première, pour que les dynamiques du réservoir aient le temps de converger vers un état stable. La seconde pour que l'apprentissage soit stable. Par exemple, pour une séquence simple de deux états comme "0 1" il faut environ 50 itérations avant que les dynamiques du réservoir se stabilisent sur un état stable

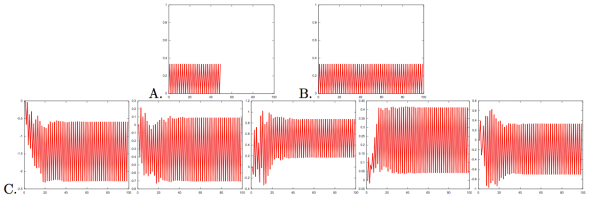


FIG. 3.19: A) Activité du signal donné à l'architecture durant les 50 premières itérations. B) Activité de sortie. Durant les 50 premières itérations, ce signal est forcé par le signal en A. Ensuite le signal correspond aux activités que l'architecture calcule. On observe que la séquence est correctement reproduite. C) Activités de 5 neurones du réservoir de dynamiques choisis arbitrairement. On observe que les dynamiques sont identiques à la phase d'apprentissage.

Echo State Networks

Si les ESNs permettent d'apprendre le timing des transitions, cette propriété reposant sur le temps de convergence vers un état stable est limitée. En effet, plus ce temps est long, plus les dynamiques du réservoir se seront dissipées jusqu'à la présentation d'un nouvel état. Comme vu

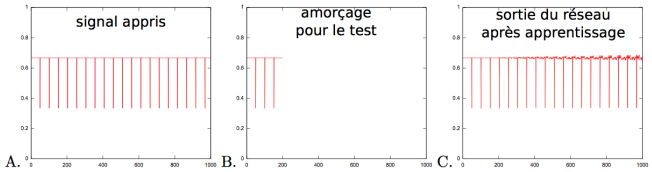


FIG. 3.21: A) Activité du signal fourni à l'architecture durant la phase d'apprentissage. Ce signal représente une séquence "2 - 1" avec un intervalle de 50 itérations pendant lesquelles l'état "2" est maintenu avant de présenter l'état "1" au réseau. B) Activité du signal fourni à l'architecture durant la phase de reproduction. Le signal est fourni durant les 200 premières itérations. C) Activité de sortie du réseau. Durant les 200 premières itérations, ce signal est forcé par le signal en B. Ensuite le signal correspond aux activités que le réseau calcule. On observe que la séquence n'est pas reproduite correctement ; elle se déforme au fil du temps.

Echo State Networks

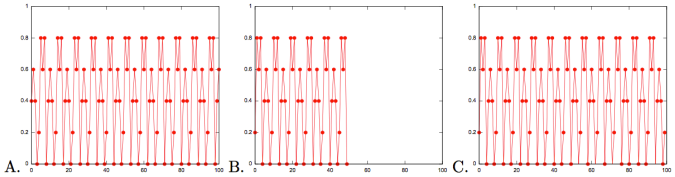


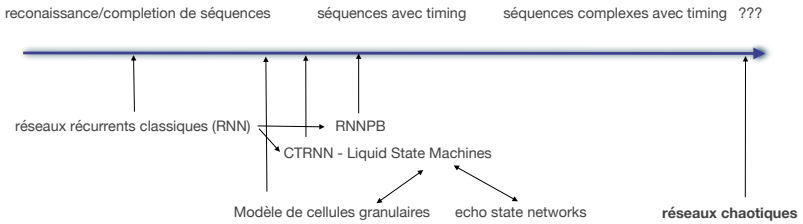
FIG. 3.22: A) Activité du signal fourni à l'architecture durant la phase d'apprentissage. Ce signal représente une séquence composée de neuf états "2 3 2 0 1 4 3 4 0". B) Activité du signal fourni à l'architecture durant la phase de reproduction. Le signal est fourni durant les 50 premières itérations. On remarque que le signal fourni est temporellement décalé par rapport à celui donné lors de la phase d'apprentissage en A. C) Activité de sortie de l'architecture. Durant les 50 premières itérations, ce signal est forcé par le signal en B. Ensuite le signal correspond aux activités que l'architecture calcul. On observe que la séquence est correctement reproduite.

Vue d'ensemble

Apprentissage non (ou peu) supervisé

Apprentissage avec un renforcement

Apprentissage de séquences temporelles



Vue d'ensemble