



INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

APPRENTISSAGE SUPERVISE

Introduction et un peu d'histoire ...

- **But : appréhender les techniques utilisées pour résoudre des problèmes liés à l'analyse, le traitement et l'apprentissage des connaissances**
- **Connaissance = « ce que l'on a acquis par l'étude ou par la pratique »**
- **Cadre général des sciences cognitives**
- **➔ étude de l' « intelligence »**
- **Multiples tentatives concrétisées dans l'histoire :**
 - ➔ pour construire des machines capables de se substituer à l'homme afin d'effectuer à sa place certaines tâches « intellectuelles » répétitives ou fastidieuses

Le boulier

La machine de Pascal : la Pascaline

Les automates de Vaucanson





La roue pascaline comptait dans le système décimal le nombre d'unités d'un ensemble de mesures non-décimales de l'ancien régime. Pour cela il fallait des machines avec des roues avec un nombre de dents différentes suivant l'unité considérée

- **Pascal fut le premier et le seul à avoir une machine à calculer opérationnelle au XVII^e siècle**
- **La Pascaline, la première machine à calculer :**
 - dont la description ait été rendue publique en son temps,
 - à posséder un reporteur qui permettait la progression effective des retenues en cascade.
 - utilisée dans un bureau (celui de son père, notaire),
 - commercialisée (avec vingt exemplaires construits),
 - brevetée (*privilège royal* de 1649)

- **au carrefour de plusieurs disciplines**

Philosophie :

1596-1650 : Descartes → animaux sont des sortes de machines vivantes, les hommes sont dotés d'une âme échappant à la matière

1646-1716: Leibniz → le matérialisme : tout est régi par des lois physiques

1561-1626: Bacon → l'empirisme : compréhension par l'intermédiaire des sens

1711-1776: Hume → Principe d'induction : les règles générales sont acquises par exposition répétée à des associations de leurs éléments

1872-1970 : Russell → Positivisme logique: le savoir est caractérisé par des liens logiques reliés aux sens

Mathématiques :

9^{ème} siècle: al-Khowarazmi → introduction de l'algorithmique, l'algèbre et la notation arabe

1815-1864: Boole → algèbre binaire et logique formelle

1848-1925: Frege → logique du premier ordre

1906-1978: Gödel → théorème d'incomplétude et d'indécidabilité = « *une théorie suffisante pour faire de l'arithmétique est nécessairement incomplète, au sens où il existe dans cette théorie des énoncés qui ne sont pas démontrables et dont la négation n'est pas non plus démontrable* »

1912-1954: Turing → Toute fonction calculable l'est par une machine de Turing (« ancêtre » de l'ordinateur) : *définition précise du concept d'algorithme ou « procédure mécanique »*



Psychologie :

1821-1894: Helmholtz, 1832-1920: Wundt: Origine de la psychologie scientifique

1878-1958: Watson, 1874-1949 Thorndike: Introduction du « **béhaviorisme** » ou « **comportementalisme** » : on peut décrire les grandes lois de l'apprentissage comme des principes gouvernant l'association entre stimulus et réponse. Ce type d'apprentissage par essai-erreur et association progressive entre une action et son résultat est à la base du « comportementalisme »

1886-1959: Tolman: → Le comportement ne peut être réduit au schéma « stimulus-réponse », le cerveau contient des représentations sous formes de « carte cognitive ».

Automates :

1592-1635: Schickard: crée la première machine à calculer à l'aide d'engrenages (addition, soustraction, multiplications, mémorisation de résultats et dépassement de capacité)

1623-1662: Pascal crée la Pascaline (1642),

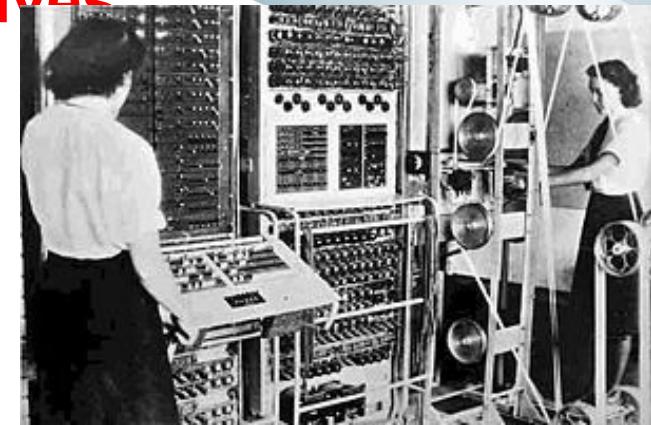
1574-1660: William Oughtred invente la règle à calcul qui utilise le principe des logarithmes en 1622

1709-1782: Vaucanson construit des automates (canard, joueur de flûte traversière) qui donnent l'illusion d'être vivants

1785-1870: Thomas de Colmar ➔ construction de l'arithmomètre considéré comme le premier calculateur mécanique de « bureau », à avoir connu un réel succès commercial, ancêtre des calculatrices

1792-1871: Babbage: ➔ « machine analytique » programmable grâce à des cartes de variables et des cartes d'opérations : ➔ sur le modèle des cartes du métier Jacquard, dont la lecture séquentielle donnait des instructions et des données à sa machine, ➔ l'ancêtre mécanique des ordinateurs d'aujourd'hui

Les sciences cognitives



Informatique :

- 1940: Equipe de Turing crée la « Heath Robinson » pour décoder les messages allemands (technologie à base de relais)
- 1943: Même équipe crée Colossus (technologie à base de lampes)
- 1940-82:
 - Z3: ordinateur programmable doté du premier langage évolué
 - EDVAC: Programme mémorisable (Von Neumann)
 - IBM701: 1952 → premier ordinateur scientifique commercial (mémoire = 2048 mots de 36 bits chacun!)
 - 1965-1980: mise au point d'algorithmes efficaces
 - 1982: ordinateurs de la 5^{ème} génération au « Japon » → machine parallèle capable de raisonner (1992)



Intelligence Artificielle :

1943: McCulloch et Pitts : création du modèle du neurone formel

1948: Création de la cybernétique par Norbert Wiener

1949: Hebb établit la première règle d'apprentissage neuronal

1950: Shannon, 1952: Samuel, 1953: **Turing**: machine à jouer aux échecs

1956: Première apparition du terme « Intelligence Artificielle » lors d'un workshop

1960: John McCarty, Allen Bewell et Herbert Simon déclarent que « l'ordinateur peut être utilisé pour autre chose que des calculs, comme par exemple « manipuler des symboles » » (idée proposée par Ada Lovelace en 1842, amie de Babbage!)

1969: arrêt des réseaux de neurones (limitations des perceptrons décrites par Minsky et Papert)

1969-1979: systèmes experts

Depuis 1986: retour des réseaux de neurones

- **Applications de l'intelligence artificielle:**

- Les systèmes d'aide au diagnostic
- la traduction automatique (initialisée pendant la guerre)
- les jeux
- l'analyse des grandes bases de données: data mining
- la reconnaissance de formes
- la commande des systèmes
- le traitement du signal, compression de données (télécommunications), suppression du bruit
- Synthétiseur vocal
- Finance: prévision du coût de la vie
- Secteur médical: analyse EEC et ECG

....



- **BIG DATA :**

Explosion des données de tout genre disponibles

Quelques chiffres

Bibliothèque d'Alexandrie: considérée comme renfermant la somme de toute la connaissance humaine (300 AV) → Aujourd'hui : estimation que l'information disponible représenterait 320x Nbre habitants dans le monde → 1200 exabytes (10^{18})

Si toutes les informations étaient mises sur CDs puis empilés → 5 colonnes atteignant toutes la Lune...

Projection : 40 zettabytes (10^{21}) d'ici 2020

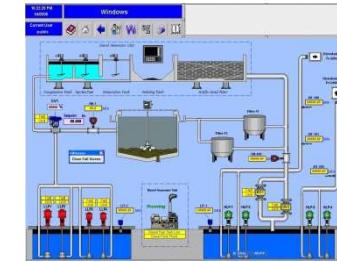
Année 2000 : $\frac{1}{4}$ des données conservées de manière numérique →
Aujourd'hui moins de 2% non numériques

**The Rise of Big Data
How It's Changing the Way We
Think About the World**
(K. Neil Cukier and V. Mayer-
Schoenberger
May/June 2013 Issue)

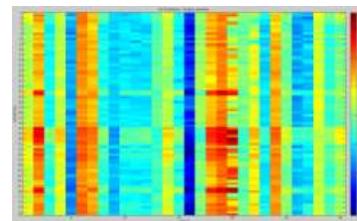
- **Idée** : on pourrait apprendre des choses d'une grande masse d'informations que l'on ne pourrait comprendre en utilisant beaucoup moins.
→ Au lieu d'essayer de comprendre précisément pourquoi un système tombe en panne dans certaines conditions → récupérer le plus de données en relation avec cette faute → en déduire des motifs (regroupements) qui permettraient de prédire l'arrivée de ce type de faute.
- **Espoir** = trouver des schémas/(associations) insoupçonnés auparavant
- Rechercher des relations de cause à effet dans un grand volume de données même si elles sont incomplètes, imprécises et parfois non pertinentes → l'idée étant de jouer sur le *volume* de données et la *variété*
- **Big Data : règle des 3V : volume, vélocité et variété**

BIG DATA : Pour faire quoi ?

- Secteurs : économie, marché, process (automobile, aéronautique, chimique, biologique, météorologique, écologique,),



médecine,



- Explosion des données Internet mais pas seulement, utilisation d'objets « connectés »,



Airinov

- Monitoring permettant la mesure en temps réel de nombreux capteurs, plus informations, plus rapides,



- Systèmes décisionnels : ex OAD (Outil d'Aide à la Décision),
→ Traiter efficacement les données pour en extraire une vraie valeur ajoutée pour l'utilisateur : table sur des modèles (aide à la décision), règles, ... ou utilisation de statistique descriptive, sur des données à forte densité en information afin de mesurer des phénomènes, détecter des tendances...
- Fouille de données : inférer des lois (régressions) sur des données à faible densité en information qui pourraient être utilisées pour leur capacité prédictive
→ *apprendre des données*

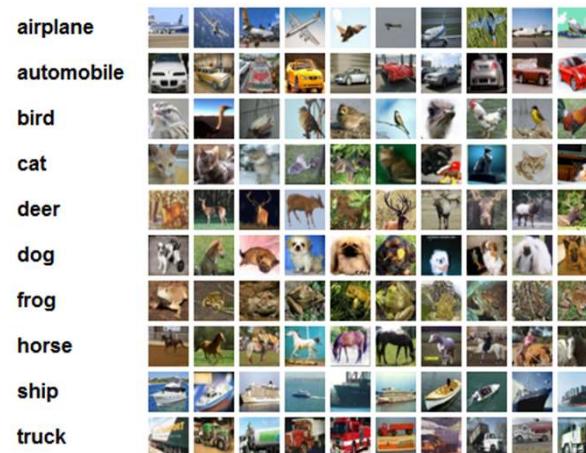
- **Machine Learning : Apprentissage**

➔ Steve Ballmer, le précédent Chief Executive Officer de Microsoft, prédit que le “ machine learning” sera le principal centre d’intérêt de l’informatique dans les toutes prochaines années (Nov 13, 2014), Computer Sciences Dept. , Harward.

- *Apprentissage : Apprendre des observations pour en déduire un modèle simple*

➔ *Appel aux sciences cognitives* : appréhender les techniques utilisées pour résoudre des problèmes liés à l’analyse, le traitement et l’apprentissage des connaissances

- **Apprentissage supervisé** : on dispose d'un ensemble de données dont on connaît la classe (classification) dites annotées de leur sorties ou le résultat de la fonction (régression) : ces valeurs sont appelées des cibles = « targets », données étiquetées → le but avoir un algorithme qui permette de prédire à partir de nouvelles données une fois que celui-ci aura été « **entraîné** »
 $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots$ et on espère **prédirer** la sortie sur de nouvelles observations : $X^* \rightarrow y^*$
- **Ex: les données d'entrée = images, et la cible = la catégorie de photo**

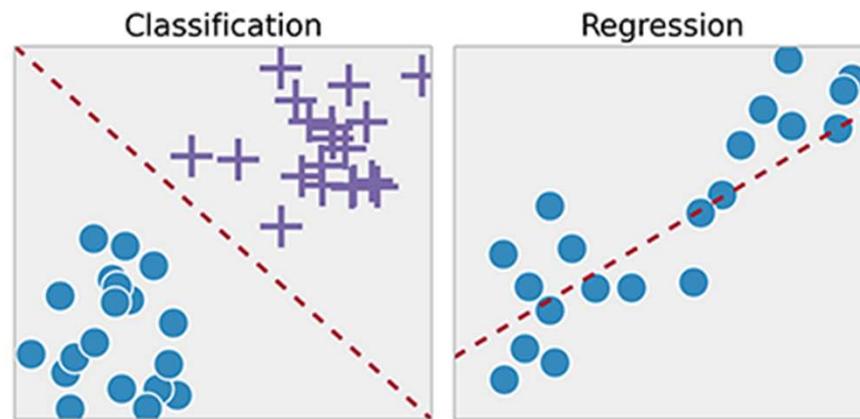


- **Apprentissage non supervisé** : on ne connaît pas les cibles → on regroupe les données par **similarité** :

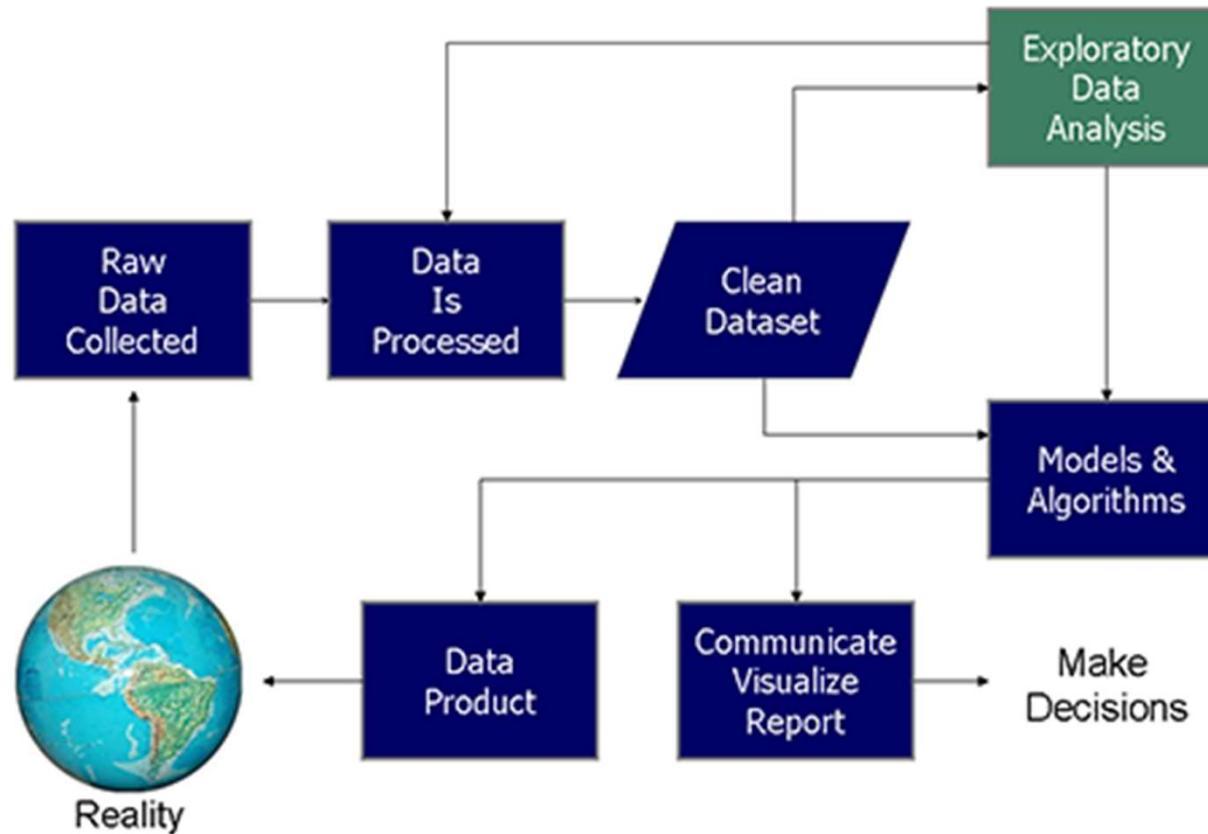
X_1, X_2, X_3, \dots et on espère découvrir la relation avec des variables latentes structurelles : $X_i \rightarrow y_i$

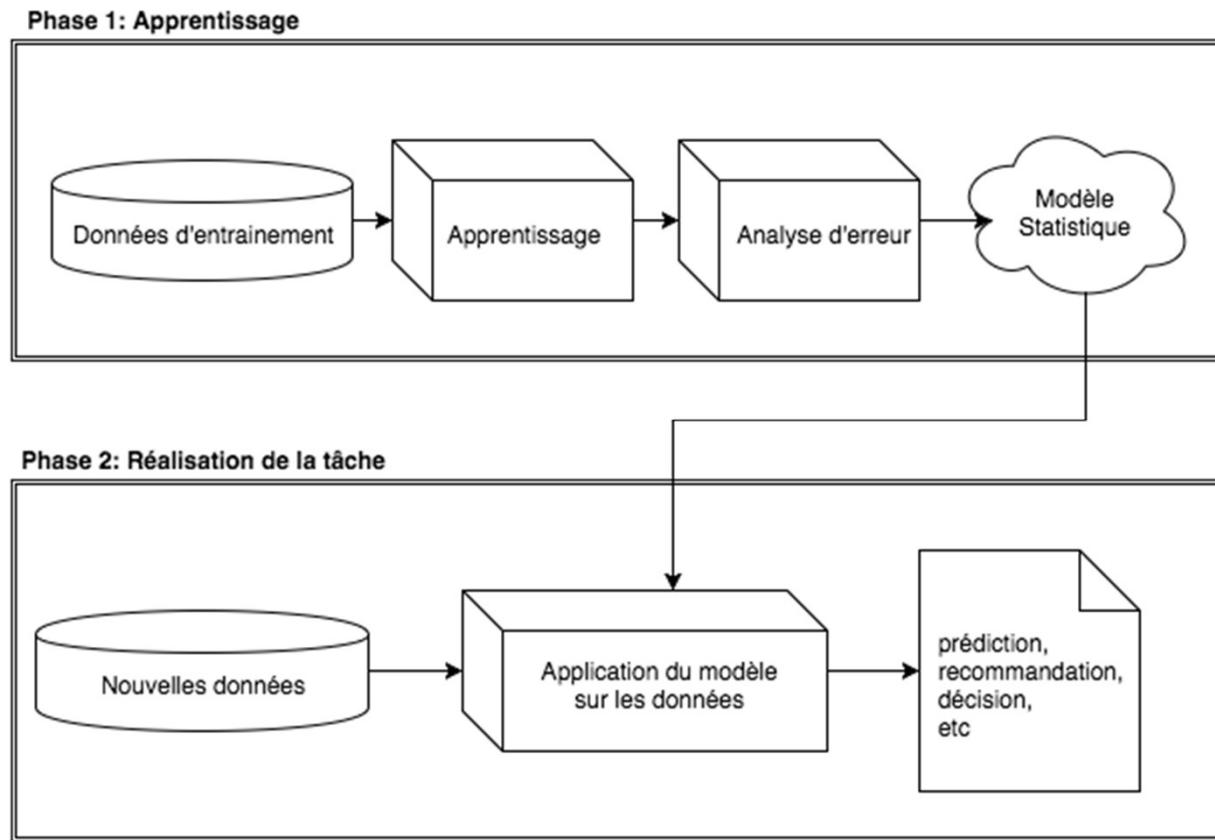
- Le **semi-supervised learning** qui prend en entrée certaines données annotées et d'autres non. Ce sont des méthodes très intéressantes qui tirent parti des deux mondes (supervised et unsupervised), mais bien sûr apportent leur lot de difficultés.
- Le **reinforcement learning (apprentissage par renforcement)** qui se base sur un cycle d'expérience / récompense et améliore les performances à chaque itération. Une analogie souvent citée est celle du cycle de dopamine : une "bonne" expérience augmente la dopamine et donc augmente la probabilité que l'agent répète l'expérience.

- Deux types de problèmes



Difference entre régression linéaire et classification linéaire





Un problème de machine learning comporte différents éléments spécifiques :

- **Les données** (les données d'entraînement mais aussi les nouvelles données)
- **La tâche spécifique** à accomplir (prédire, recommander, décider quelque chose, etc.)
- **L'algorithme d'apprentissage** en lui-même
- **L'analyse d'erreur** (ou mesure des performances du modèle)



- Comment juger que l'apprentissage est réussi?
- → Validation du modèle : estimation de la fiabilité d'un modèle

La validation croisée¹ (« cross-validation ») est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage.

- 3 méthodes:
 1. « testset validation » (test validation) ou « holdout method »,
 2. « k-fold cross-validation »
 3. « leave-one-out cross-validation » (LOOCV).

On définit aussi :

- **VP (vrais positifs)** représente le nombre d'individus malades avec un test positif,
 - **FP (faux positifs)** représente le nombre d'individus non malades avec un test positif,
 - **FN (faux négatifs)** représente le nombre d'individus malades avec un test négatif,
 - **VN (vrais négatifs)** représente le nombre d'individus non malades avec un test négatif.
1. **SENSIBILITE** : **true positive rate, the recall, or probability of detection** → La sensibilité, ou la probabilité que le test soit positif si la maladie est présente, se mesure chez les malades seulement ($VP+FN$) : **proportion de positifs correctement identifiés** $VP/(VP+FN)$
 2. **SPECIFITE**: **true negative rate**: une mesure de la sensibilité s'accompagne toujours d'une mesure de la **spécificité**. Cette dernière se mesure chez les non-malades seulement. Ainsi, la spécificité, ou la probabilité d'obtenir un test négatif chez les non-malades, est donné par : $VN/(VN+FP)$

- root-mean-square deviation (RMSD) or root-mean-square error (RMSE)

- $$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

Avec \hat{y}_i valeur calculée par le modèle, n le nombre d'exemples dans la base d'apprentissage, y_i la valeur dans la base d'apprentissage.

-

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$$

- La **première méthode** est très simple, il suffit de diviser l'échantillon de taille n en deux sous échantillons, le premier d'apprentissage (communément supérieur à 60 % de l'échantillon) et le second de **test**. Le modèle est bâti sur l'échantillon d'apprentissage et validé sur l'échantillon de test. L'erreur est estimée en calculant un test, une mesure ou un score de performance du modèle sur l'échantillon de test : « **testset validation** » (test validation) ou « **holdout method** »,
- **Dans la seconde**, division de la base en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les $(k-1)$ autres échantillons constitueront l'ensemble d'apprentissage. On calcule comme dans la première méthode le score de performance. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les $(k-1)$ échantillons qui n'ont pas encore été utilisés pour la validation du modèle. ➔ répétition de l'opération ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des k erreurs quadratiques moyennes est enfin calculée pour estimer l'erreur de prédiction: « **k-fold cross-validation** »
- **La troisième méthode** est un cas particulier de la deuxième méthode où $k=n$, c'est-à-dire que l'on apprend sur $(n-1)$ observations puis on valide le modèle sur la n ième observation et l'on répète cette opération n fois².

La **régression linéaire** s'appuie sur l'hypothèse qu'il existe une relation linéaire entre l'entrée (les observations) et la sortie (les prédictions).

$$\hat{y}_i = \theta_0 + \theta_1 \times x_i$$

soit : $\hat{y} = \theta^T x$

avec $x = (1, x_1, x_2, \dots, x_N)$, $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N)$

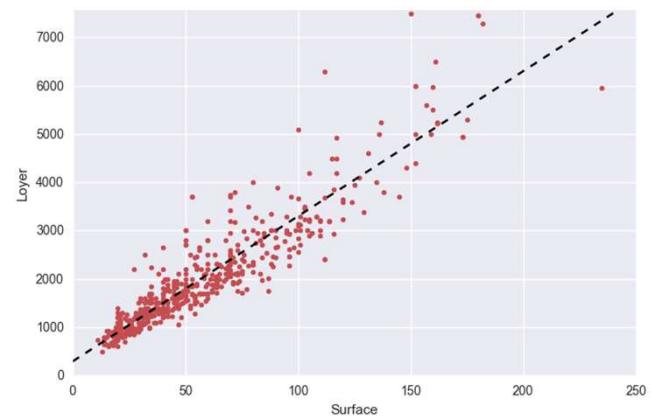
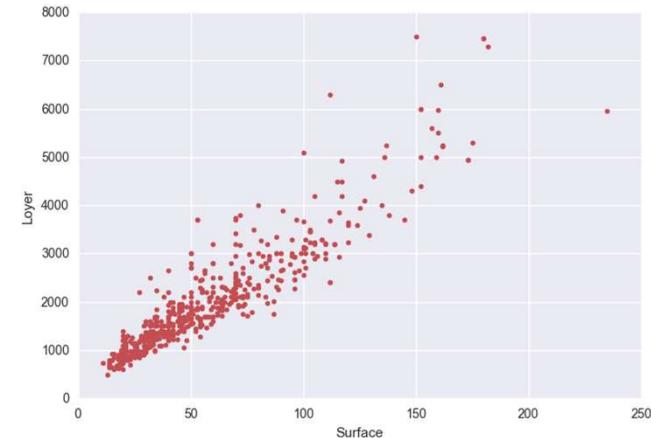
et $\theta = (\theta_0, \theta_1)$

Moindres carrés:

$$E = \sum_{i=1}^n (\theta^T x_i - y_i)^2$$

Une solution analytique:

$$\theta^T = (X^T X)^{-1} X^T y$$



→ loyer = 30 × surface + 294.3

Pb de grande dimension → nécessite l'inversion d'une matrice → pb convexe (somme des carrés) → utiliser un algorithme d'optimisation **descente de gradient** ou autre (**Lewenberg- Marquard**)



- **Apprentissage supervisé**
- - k-NN :
- Réseaux de neurones
- SVM (Machines à Vecteurs de Support)
- *Random Forests*

Méthode k-NN Méthode de k plus proches voisins

- La méthode k-NN où KNN (K Nearest Neighbors) ou k-plus proches voisins est sans doute la méthode la plus simple des méthodes d'apprentissage supervisé (dès les années 1970).
- Cette méthode est dédiée à la classification
- Cette méthode consiste à partir d'une base d'apprentissage D avec des individus y (exemples) dont la classe a été pré-affectée (apprentissage supervisé).
- Quand un nouvel élément x doit être classé : on regarde autour de celui-ci quels sont ses k-plus proches voisins (par mesure de similarité) dans la base d'apprentissage et on lui affecte la classe qui est ressort le plus souvent.

➔ Il faut donc : définir une mesure de similarité et le nombre k de voisins

- Notion de similarité :

→ On va utiliser un distance: $d(x,y)$

→ Plusieurs cas possibles :

- La distance de Manhattan: qui calcule la somme des valeur absolue des différences entre les coordonnées de deux points

$$d(x,y) = \sum_{i=1}^n |x_i - y_i|$$

- La distance euclidienne:

$$d(x,y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- La distance de Minkowski:

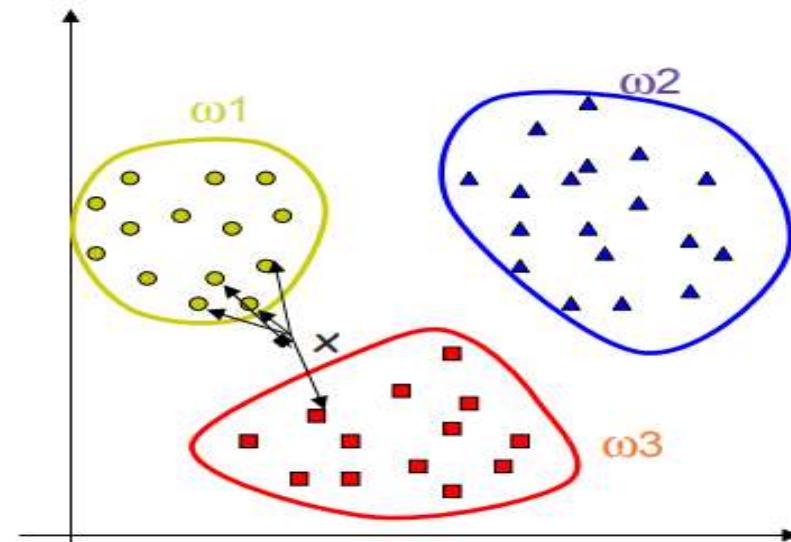
$$d(x,y) = \sqrt[q]{\sum_{i=1}^n |x_i - y_i|^q}$$



- Pour un nouvel exemple non étiqueté x (**de classe inconnue**) , trouver les k plus proches exemples étiquetés (voisins de classes connues) de la base d'apprentissage.
- La classe associée à x est la classe majoritaire qui apparaît.

Dans l'exemple suivant, nous avons **3** classes (ω_1 , ω_2 , ω_3) et le but est de trouver la valeur de la classe de l'exemple inconnu x .

Nous prenons la distance Euclidienne comme métrique de proximité et $k=5$ voisins.



- Parmi les 5 plus proches voisins, 4 appartiennent à ω_1 et 1 appartient à ω_3 , donc x est affecté à ω_1 , la classe majoritaire.



- Algorithme
- Soit $D = \{(y, c), c \in C\}$ l'ensemble des individus de la base d'apprentissage
- Soit x l'élément (exemple) à classer

. Début

- Pour chaque $((y, c), \in D)$ faire:

Calculer la distance $d(x, y)$

- Fin

- Pour chaque $\{y \in knn(x)\}$:

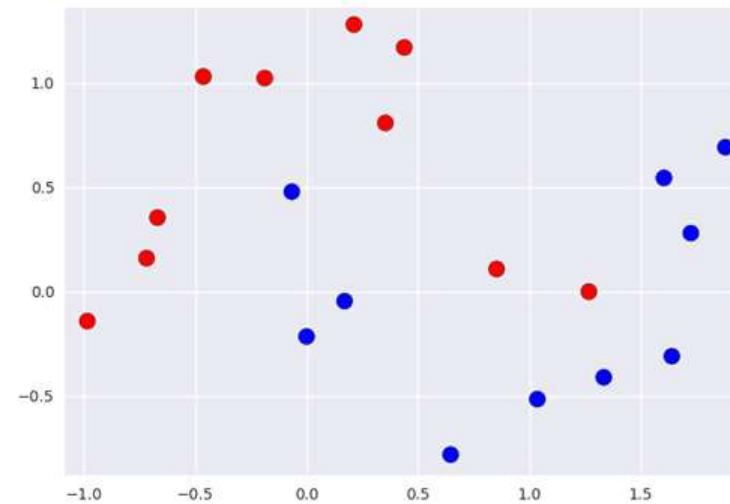
Compter le nombre d'occurrence de chaque classe

- Fin

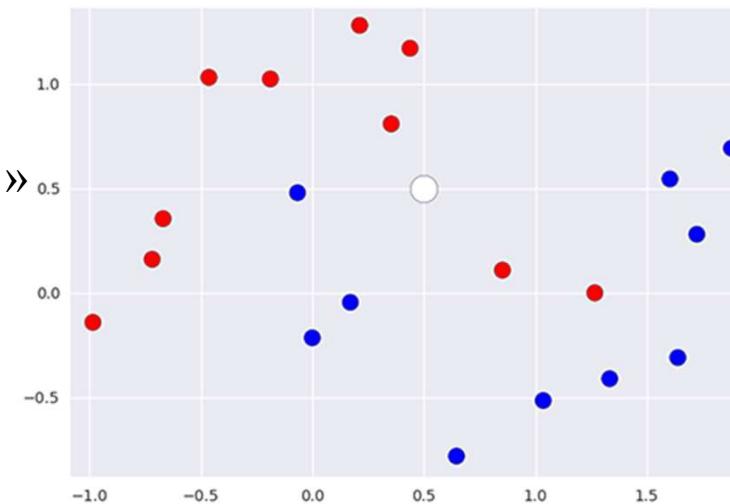
Attribuer à x la classe la plus fréquente

. Fin

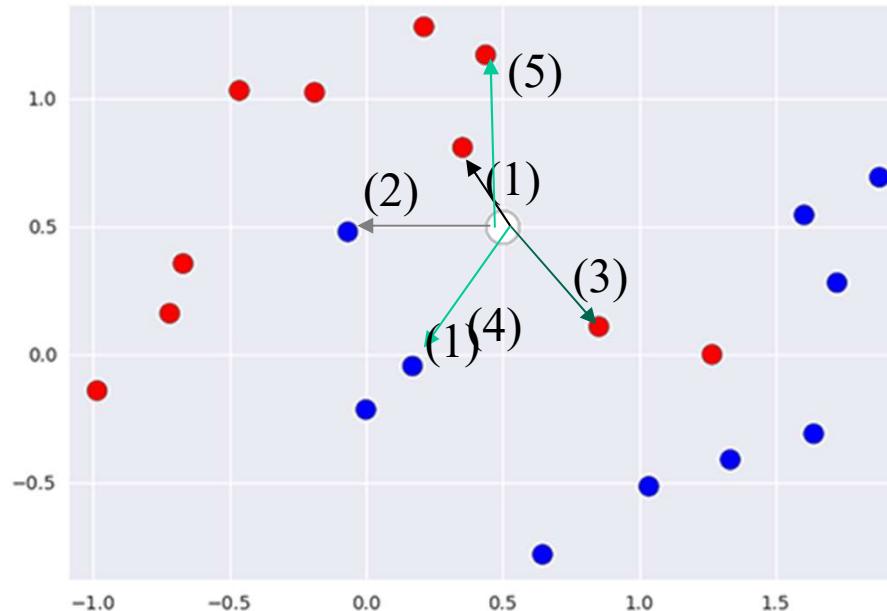
Base d'apprentissage



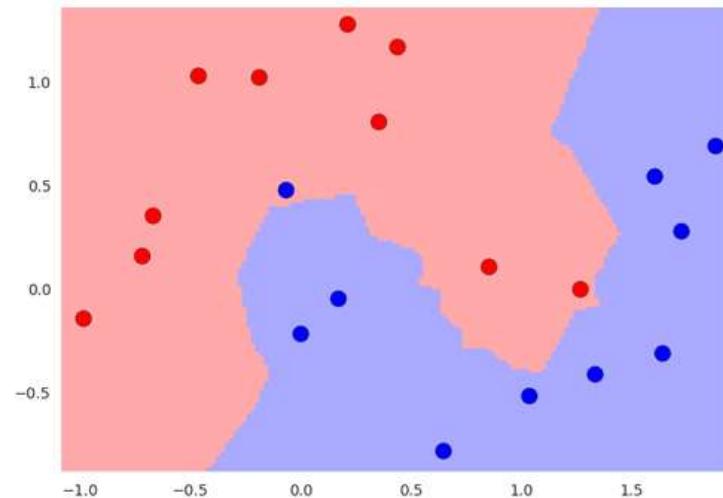
Classer le point « blanc »



- En utilisant :



- K=1 : le plus proche voisin → classe=« rouge »
- K=2 : classe=indéterminée
- K=3: classe=« rouge »
- K=4: classe indéterminée
- K=5: classe « rouge »



- **Résultat :**
- **dépend de la distance choisie**
- **Dépend de la valeur de k**
 - $k=1$ pour des frontières des classes très complexes:
 - ➔ très sensible aux fluctuations des données (variance élevée),
 - ➔ risque de sur-ajustement (sur-apprentissage = bon pour l'apprentissage, mauvais pour le test)
 - ➔ ne convient pas aux données bruitées
 - $k=n$ (nombre d'individus dans la base) frontière rigide
 - ➔ moins sensible au bruit
 - ➔ Plus k est grand plus l'affectation est correctement réalisée

- **Avantages:**
 - ➔ algorithme simple
 - ➔ adapté aux classes où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières (ex: reconnaissance de chiffres manuscrits ou d'images satellites)
 - ➔ peut être utilisé aussi pour effectuer une prédiction (moyenne des k plus proches voisins)
- **Inconvénients:**
 - ➔ pas d'apprentissage en tant que tel sur la base : « lazy algorithm »
 - ➔ prédiction lente car il faut calculer la distance à tous les points de la base d'apprentissage à chaque fois
 - ➔ nécessite de stocker toute la base d'apprentissage en mémoire (memory-based)
 - ➔ sensible aux attributs (features) non pertinents et corrélés
 - ➔ peu adapté au problème de grande dimensionnalité (big data)

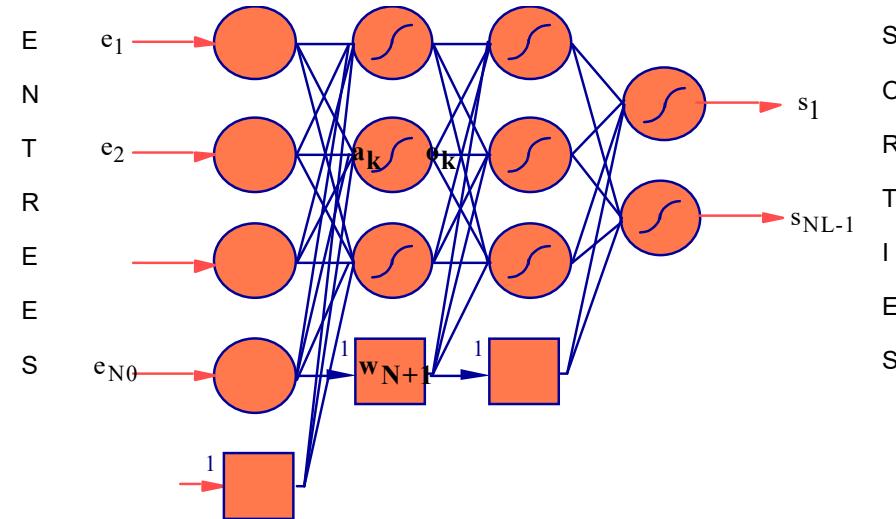


INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

LES RESEAUX DE NEURONES

LES RESEAUX DE NEURONES

→ une représentation entrées-sorties particulière



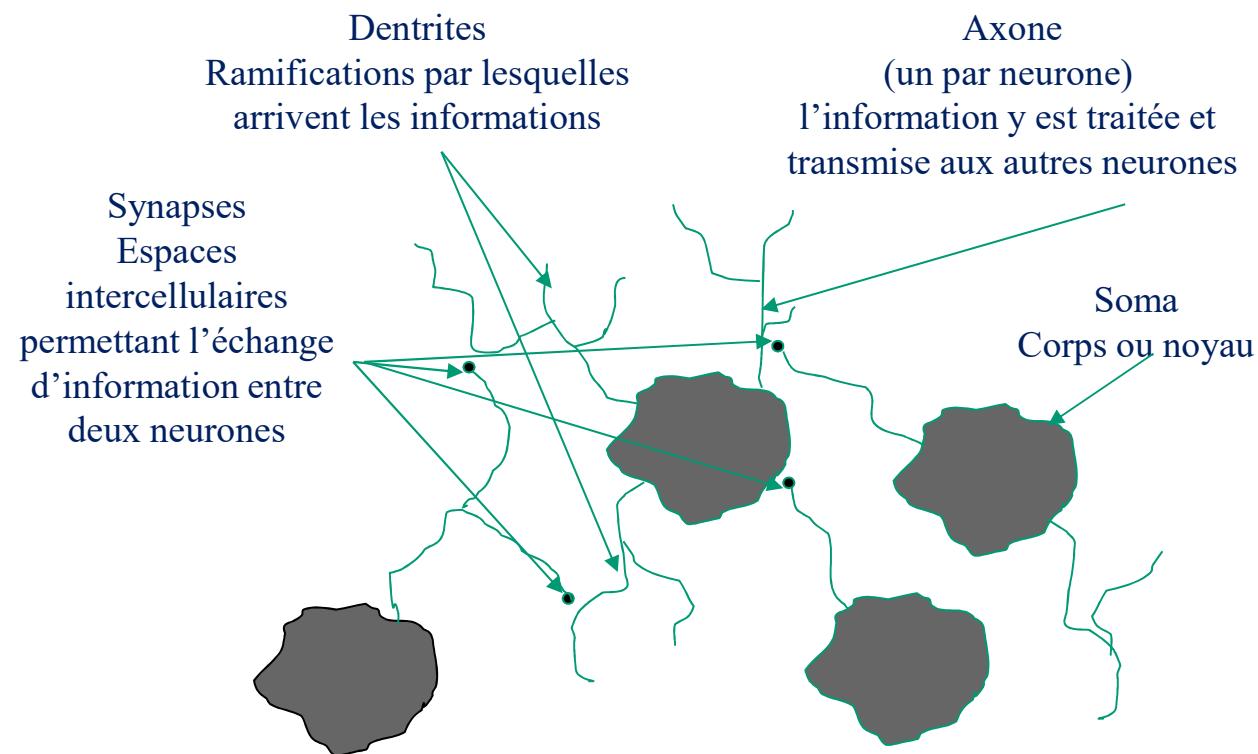
1890 : W. James, célèbre psychologue américain, introduit le concept de mémoire associative et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones

1943 : Début des recherches dans le domaine du connexionnisme : par J. McCulloch et W. Pitts, les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (études théoriques) puis par Wiener et Von Neumann (1946 -->1953)

==> Automate booléen

1949 : **Hebb**, psychologue et neuropsychologue canadien: travaux sur l'apprentissage par des réseaux de neurones artificiels qui ont eu une influence décisive sur les neurosciences cognitives et l'intelligence artificielle, une des sources de la révolution cognitive aux États-Unis

Le modèle neurophysiologique

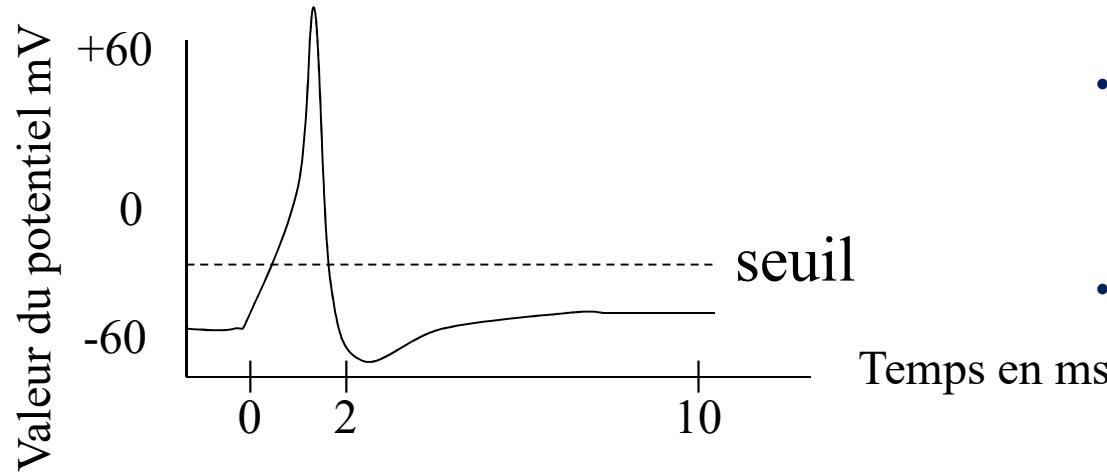


Quelques repères

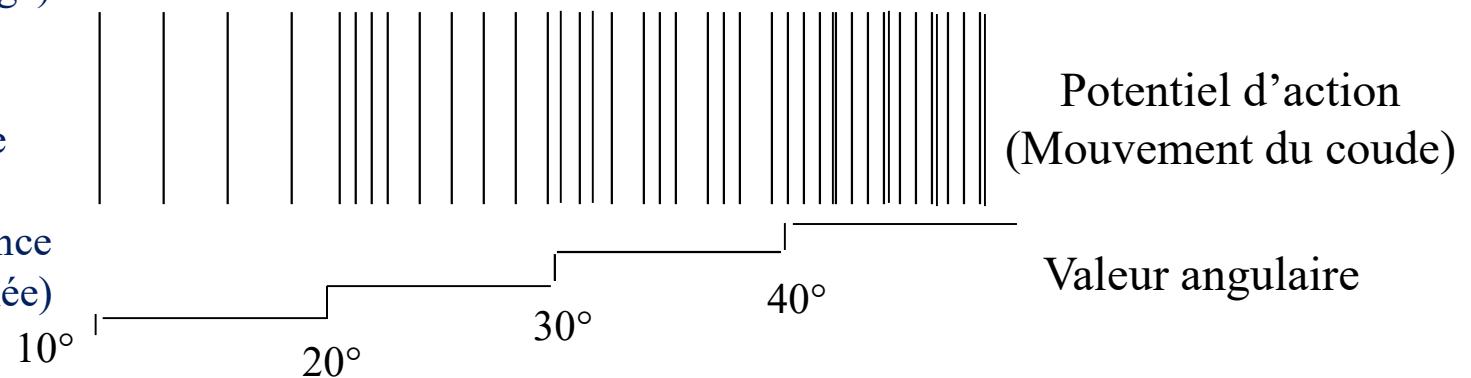
- Le cerveau se compose d'environ 10^{12} (mille milliards) de neurones
- Espace intercellulaire (la synapse) : quelques dizaines d'Angstroms
- Longueur de l'axone : de quelques microns à 1,50 mètres
- Le corps cellulaire du neurone est le centre de contrôle : c'est là que les informations reçues sont interprétées
- Le passage d'information entre neurones repose sur la création d'un potentiel d'action



Notions de potentiel d'action



- Information (message) codé en fréquence : nombre de potentiel d'action par seconde (fréquence) et les variations en fréquence (fréquence instantanée)



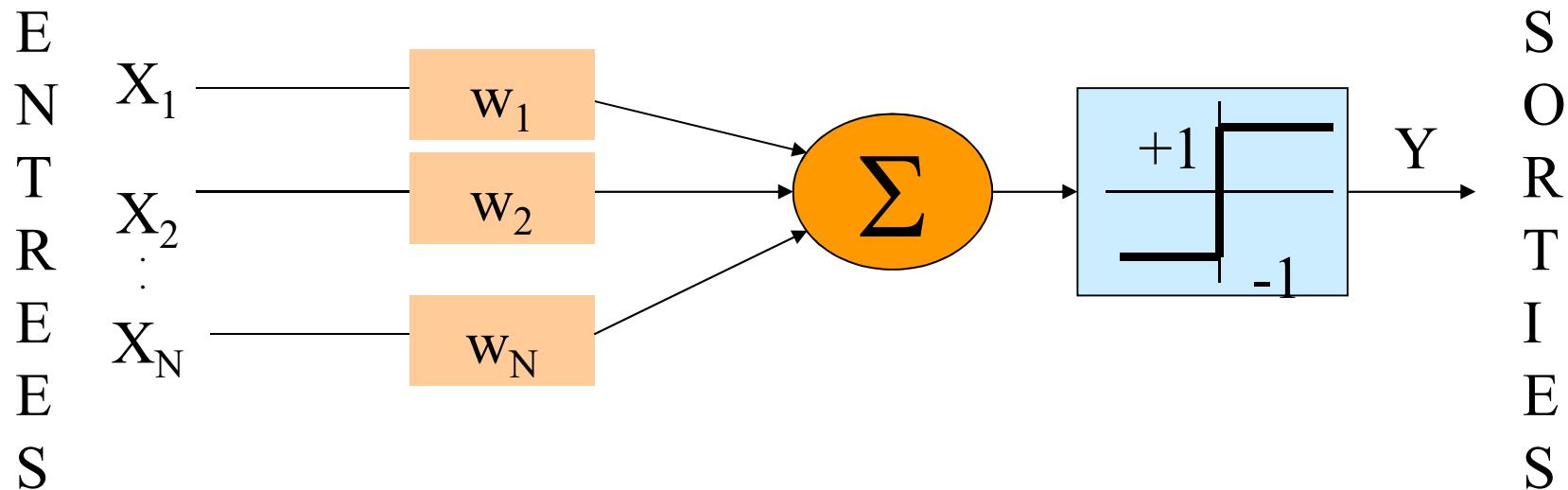
- Différence de potentiel entre milieu intérieur et milieu extérieur à la cellule (au repos) = -60 mV
- Si la différence de potentiel devient supérieur au seuil, alors il y a libération d'un neuromédiateur dans l'espace synaptique

Définitions

- Les réseaux de neurones artificiels sont des réseaux fortement connectés de **processeurs élémentaires** fonctionnant en parallèle
- Chaque processeur élémentaire calcule **une sortie unique** sur la base des informations qu'il reçoit.
- Toute structure hiérarchique de réseaux est évidemment un réseau.
- L'**apprentissage** est une phase de développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré : modification des **poids des connections**

Quelques dates clés

- 1890 : W. James, célèbre psychologue américain, introduit le concept de mémoire associative et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones
- 1943 : Début des recherches dans le domaine du connexionnisme : par J. McCulloch et W. Pitts, les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (études théoriques) puis par Wiener et Von Neumann (1946 -->1953)
==> Automate booléen
- 1949 : Hebb, psychologue et neuropsychologue canadien: travaux sur l'apprentissage par des réseaux de neurones artificiels qui ont eu une influence décisive sur les neurosciences cognitives et l'intelligence artificielle, une des sources de la révolution cognitive aux États-Unis



- X_k est la k ième entrée binaire du neurone
- w_{jk} est le “poids” de la connexion entre la k ième entrée et le neurone
- Y est la sortie du neurone

Fonctionnement :

- Si la somme > seuil ==> activation du neurone (réponse en sortie)
- Sinon aucune réponse
- Les poids étaient fixés a priori.



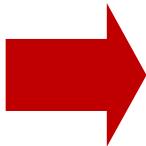
Notion d'apprentissage



1949 : Travaux de D. Hebb , physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes (conditionnement de type pavlovien) ==>introduction de la notion d'**apprentissage**

modification des poids des connexions :

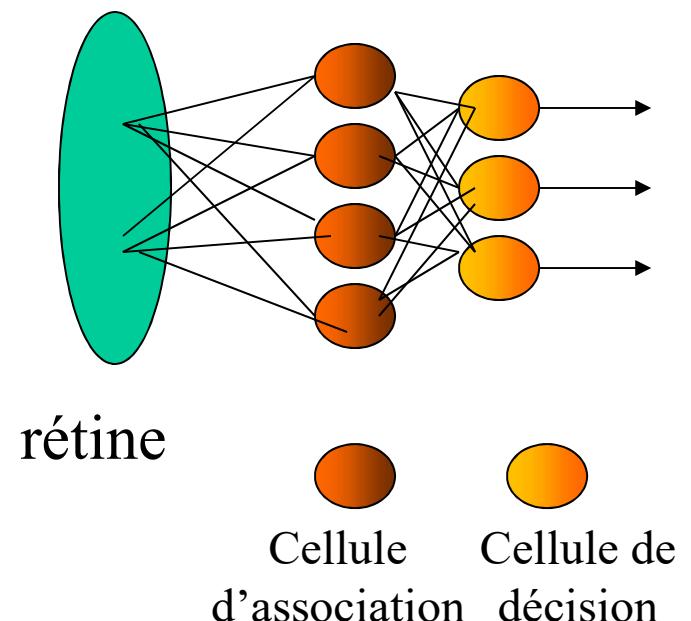
$$w_{ij}(k+1) = w_{ij}(k) + \mu a_i a_j$$



1958 : Travaux de

F. Rosenblatt ==> **PERCEPTRON**

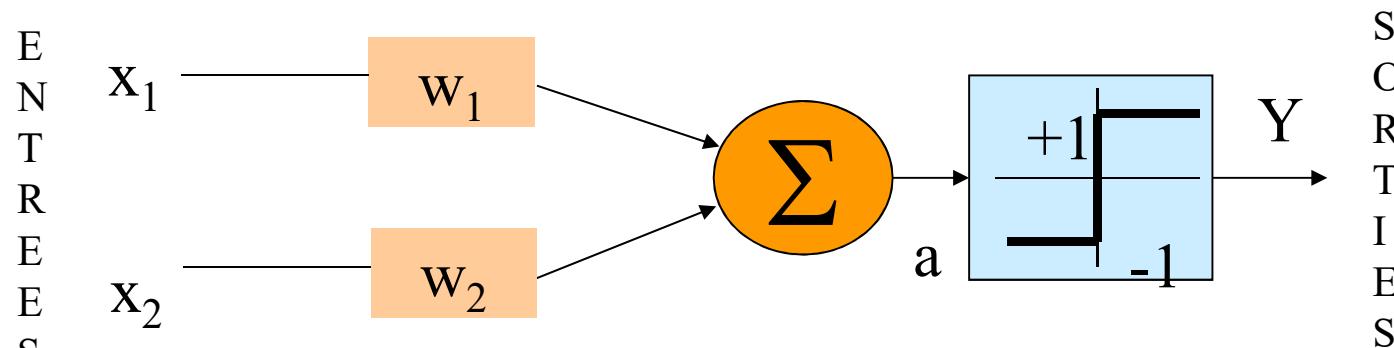
(Traitement d'images)



Notion d'apprentissage

- 1982 : J.J. Hopfield prone l'utilisation des réseaux de neurones et sa notoriété en tant que physicien va faire redémarrer les travaux sur les réseaux de neurones
- 1986 : Travaux de Rumelhart et Mc Clelland
==> méthode d'apprentissage du perceptron multicouche ou réseau de neurones multicouche
→ algorithme de rétropopagation
- PMC ==> permet d'approximer toute fonction continue (Hornick et al. 1989)

Exemple d'apprentissage non supervisé

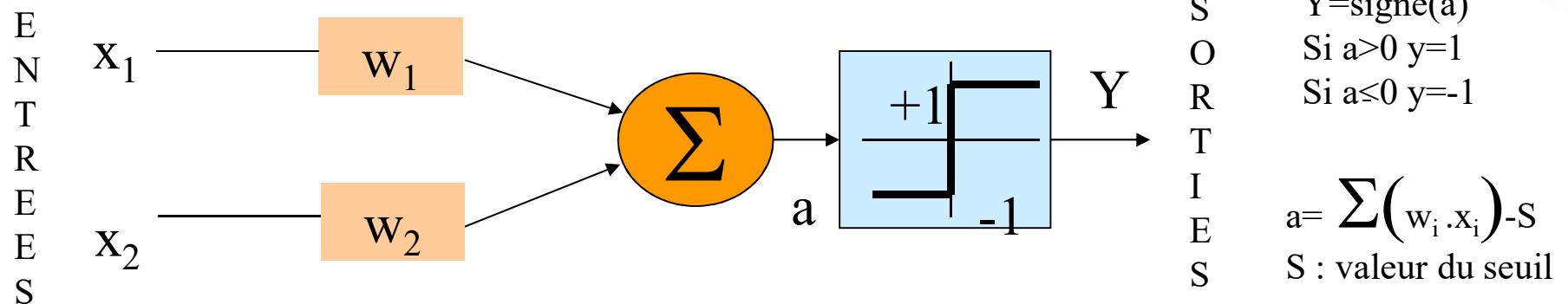


- **Base d'apprentissage = 4 exemples**

X_1	X_2	Y	(1)
1	1	1	(2)
1	-1	1	(3)
-1	1	-1	(4)
-1	-1	-1	(5)

- **Conditions initiales $\mu=1$, les poids et le seuil sont nuls**

Le neurone élémentaire



La loi de Hebb : un exemple d'apprentissage non supervisé

- 1) Initialisation des poids et du seuil S à des valeurs petites
 - 2) Présentation d'une entrée $E_k = (x_1, \dots, x_n)$ de la base d'apprentissage
 - 3) Calcul de la sortie Y obtenue
 - 4) Si Y est différente de la sortie désirée d_1 , modification des poids w_{ij} :
- $$w_i(t+1) = w_i(t) + \mu \cdot (x_i \cdot d_k)$$
- 5) Tant que tous les exemples de la base d'apprentissage ne sont pas traités
Correctement, retour à l'étape 2

Exemple



1. Conditions initiales $\mu=1$, les poids et le seuil sont nuls
2. Présentation d'une valeur de la base d'apprentissage

3. Calcul de la valeur de y pour l'exemple (1)

$$a = w_1 x_1 + w_2 x_2 - S = 0.0 \times 1 + 0.0 \times 1 - 0.0 = 0$$

$$a \leq 0 \Rightarrow y = -1$$

4. La sortie y est fausse, donc il faut modifier les poids en appliquant :

$$w_1 = w_1 + x_1 d = 0.0 + 1 \times 1 = 1$$

$$w_2 = w_2 + x_2 d = 0.0 + 1 \times 1 = 1$$

2. On passe à l'exemple (2)

$$a = w_1 x_1 + w_2 x_2 - S = 1 \times 1 + 1 \times -1 - 0.0 = 0$$

3. Calcul de la valeur de y pour l'exemple (2)

$$a \leq 0 \Rightarrow y = -1$$

2. La sortie y est fausse, donc il faut modifier les poids en appliquant :

$$w_1 = w_1 + x_1 d = 1 + 1 \times 1 = 2$$

$$w_2 = w_2 + x_2 d = 1 + 1 \times -1 = 0$$



2. On passe à l'exemple (3)

3. Calcul de la valeur de y pour l'exemple (3)

$$a = w_1x_1 + w_2x_2 - S = 2 \times -1 + 0 \times 1 - 0.0 = -2$$

$$a \leq 0 \Rightarrow y = -1$$

4. La sortie y est vraie, donc on ne modifie pas les poids

5. On passe à l'exemple (4)

6. Calcul de la valeur de y pour l'exemple (4)

$$a = w_1x_1 + w_2x_2 - S = 2 \times -1 + 0 \times -1 - 0.0 = -2$$

$$a \leq 0 \Rightarrow y = -1$$

7. La sortie y est vraie, donc on ne modifie pas les poids



2. On repasse à l'exemple (1)

3. Calcul de la valeur de y pour l'exemple (1)

$$a = w_1 x_1 + w_2 x_2 - S = -2 \times -1 + 0 \times 1 - 0.0 = 2$$

$$a \geq 0 \Rightarrow y = 1$$

4. La sortie y est vraie, donc on ne modifie pas les poids

On passe à l'exemple (2)

5. Calcul de la valeur de y pour l'exemple (2)

$$a = w_1 x_1 + w_2 x_2 - S = 2 \times 1 + 0 \times -1 - 0.0 = 2$$

$$a \geq 0 \Rightarrow y = 1$$

Tous les exemples sont bien traités. Donc les valeurs des poids sont : $w_1 = 2, w_2 = 0$



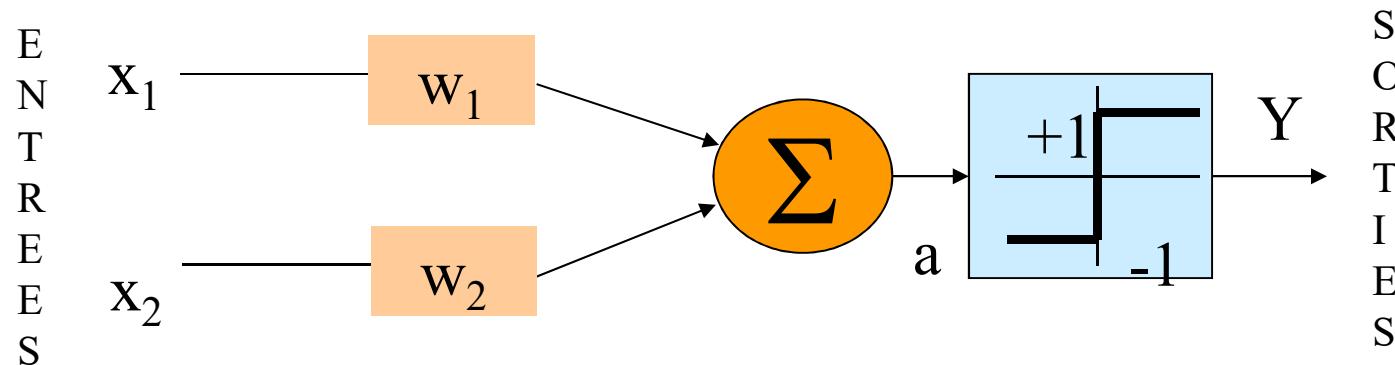
L'apprentissage fait appel à des exemples de comportement

Apprentissage non supervisé : auto-organisation du réseau au moyen de lois locales (renforcement-inhibition) qui contrôlent l'évolution des poids

Apprentissage supervisé : réalisé à partir d'une base d'apprentissage constituée d'exemples de type entrées-sorties

Apprentissage renforcé : utilisé lorsque les sorties désirées ne sont pas connues mais que l'on a un moyen pour mesurer algébriquement la qualité des sorties produites

La loi d'apprentissage du Perceptron : un exemple d'apprentissage supervisé



- 1) Initialisation des poids et du seuil S à des valeurs petites
- 2) Présentation d'une entrée $E_k=(x_1, \dots, x_n)$ de la base d'apprentissage
- 3) Calcul de la sortie Y_k obtenue :

$$a = \sum(w_i \cdot x_i) - S \text{ et } Y = \text{signe}(a) \quad (a > 0 \text{ } y = +1 \text{ sinon } y = -1)$$

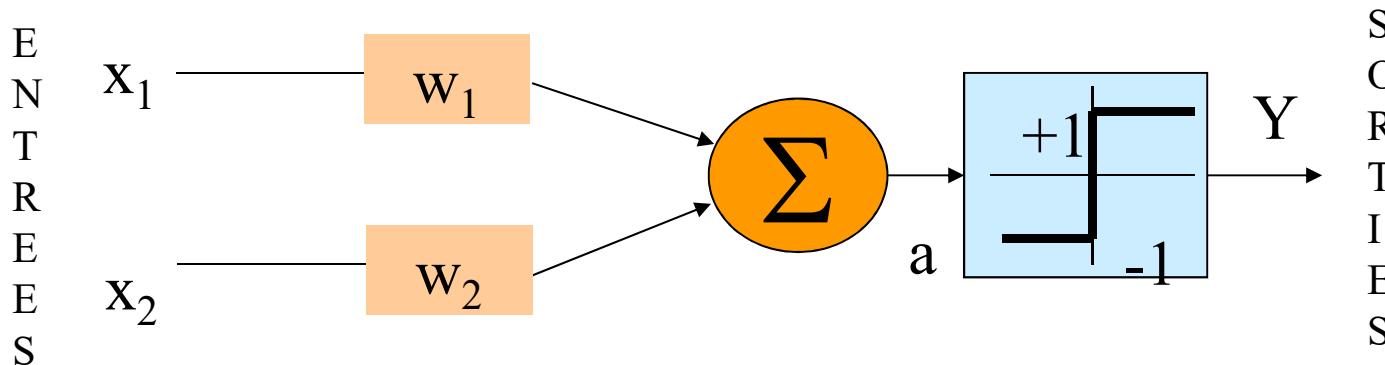
- 4) Si Y_k est différente de la sortie désirée d_k , modification des poids $w_{i,k}$:

$$w_{i,k}(t+1) = w_{i,k}(t) + \mu \cdot ((d_k - Y_k) \cdot x_i)$$

$(d_k - Y_k)$ est une estimation de l'erreur

- 5) Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement, retour à l'étape 2

Exemple d'apprentissage supervisé



- Base d'apprentissage = 4 exemples

X_1	X_2	Y	(1)
1	1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)
1	-1	-1	(5)

- Conditions initiales $\mu=0.1$, les poids et le seuil sont nuls



1. Conditions initiales $\mu=0.1$, les poids et le seuil sont nuls
2. Présentation d'une valeur de la base d'apprentissage
3. Calcul de la valeur de y pour l'exemple (1)

$$a = w_1 x_1 + w_2 x_2 - S = 0.0 \times 1 + 0.0 \times 1 - 0.0 = 0$$

$$a \leq 0 \Rightarrow y = -1$$

1. La sortie y est fausse, donc il faut modifier les poids en appliquant :

$$w_1 = w_1 + \mu x_1 (d - y) = 0.0 + 0.1(1 + 1).1 = 0.2$$

$$w_2 = w_2 + \mu x_2 (d - y) = 0.0 + 0.1(1 + 1).1 = 0.2$$



2. On passe à l'exemple (2)

3. Calcul de la valeur de y pour l'exemple (2)

$$a = w_1x_1 + w_2x_2 - S = 0.2 \times -1 + 0.2 \times 1 - 0.0 = 0$$

$$a \leq 0 \Rightarrow y = -1$$

4. La sortie y est bonne, donc on ne modifie pas les poids

2. On passe à l'exemple (3)

3. Calcul de la valeur de y pour l'exemple (3)

$$a = w_1x_1 + w_2x_2 - S = 0.2 \times -1 + 0.2 \times -1 - 0.0 = -0.4$$

$$a \leq 0 \Rightarrow y = -1$$

4. La sortie y est vraie, donc on ne modifie pas les poids



2. On passe à l'exemple (4)

3. Calcul de la valeur de y pour l'exemple (4)

$$a = w_1 x_1 + w_2 x_2 - S = 0.2 \times 1 + 0.2 \times -1 - 0.0 = 0$$

$$a \leq 0 \Rightarrow y = -1$$

4. La sortie y est vraie, donc on ne modifie pas les poids

2. On passe à l'exemple (1)

3. Calcul de la valeur de y pour l'exemple (1)

$$a = w_1 x_1 + w_2 x_2 - S = 0.2 \times 1 + 0.2 \times 1 - 0.0 = 0.4$$

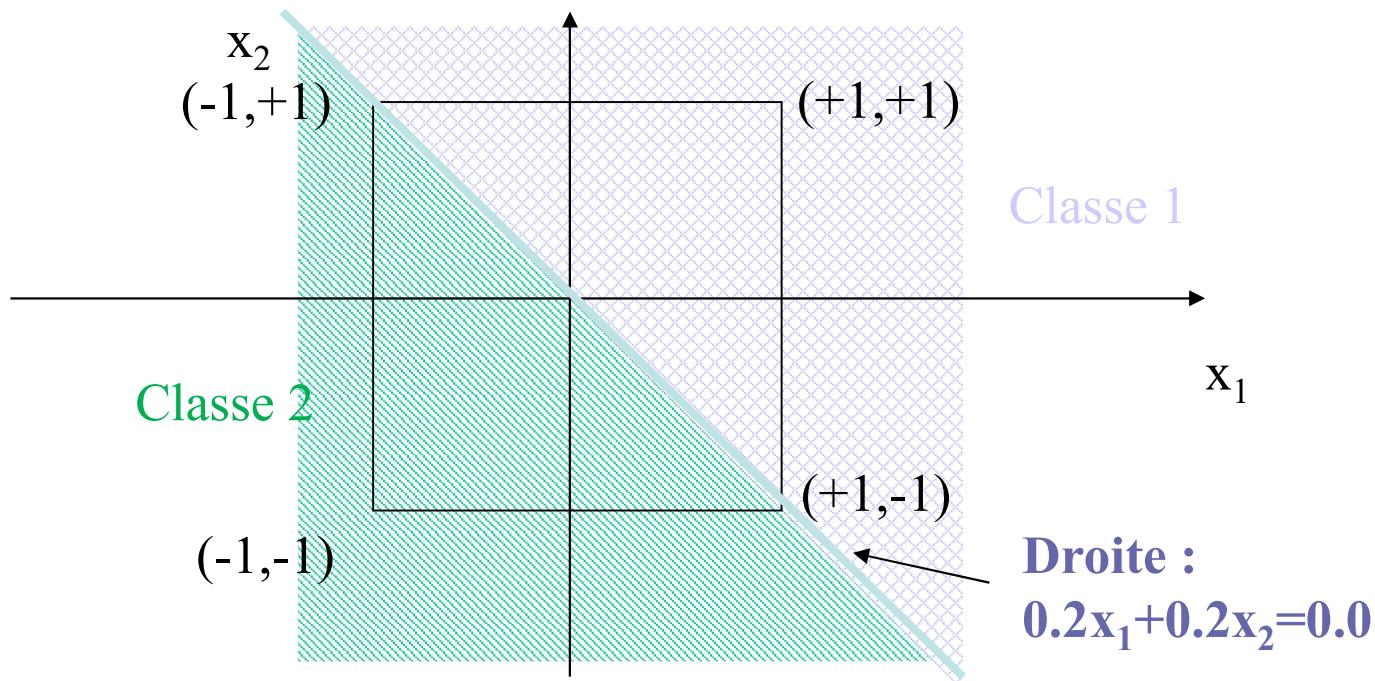
$$a > 0 \Rightarrow y = 1$$

4. La sortie y est vraie, donc on ne modifie pas les poids

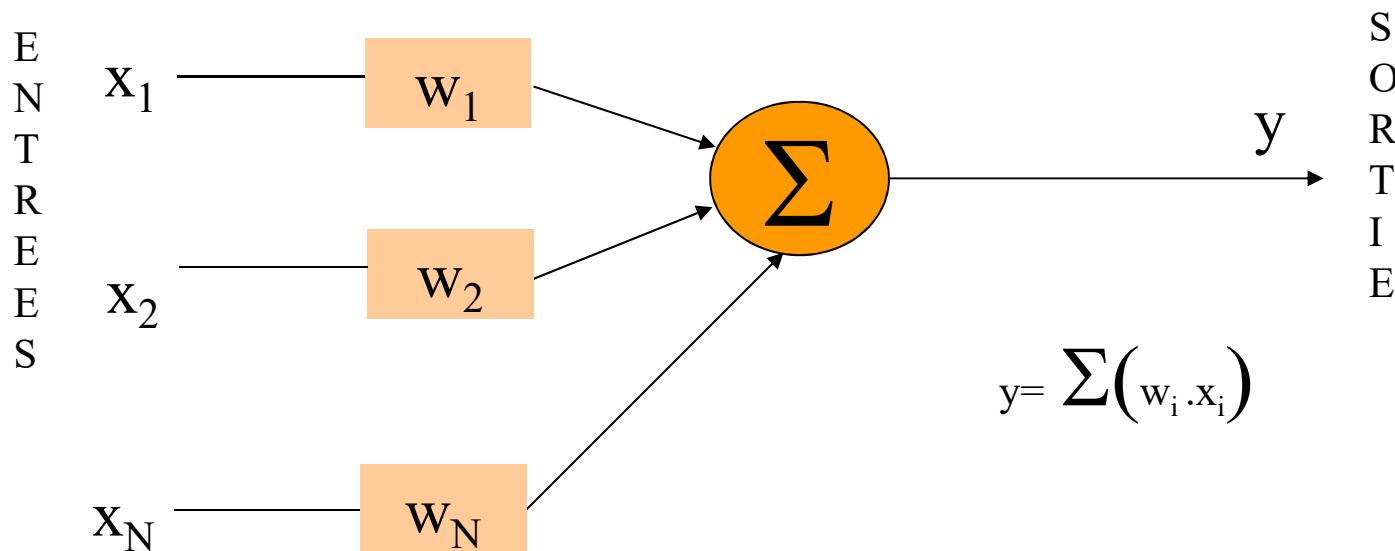
Interprétation géométrique

Le Perceptron réalise une partition de son espace d'entrée en 2 classes : (1 ou 2) selon la valeur de sa sortie (+1 ou -1).

La séparation de ces deux zones est effectuée par un hyperplan. L'équation de la droite séparatrice est : $w_1x_1+w_2x_2-S=0.0$



Apprentissage par la pseudo-inverse



Discriminant linéaire, Classification linéaire

Apprentissage supervisé : valeurs cibles d_k (target) ($k=1, P$) P = nombre d'exemples

S'il y avait autant de poids (N) que P : il y aurait une solution simple et directe, celle d'un système linéaire à $P=N$ variables

En fait $P > N$ et donc c'est un problème d'optimisation : modification adaptative des poids : ADaptive LINear Element : ADALINE



Adaline (suite)

- Minimisation de l'erreur de prédiction :

$$E = \sum_{k=1}^P (d_k - y_k)^2 = \sum_{k=1}^P (d_k - W^T X_k)^2 = \|D^T - W^T X\|^2$$

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,P} \\ x_{2,1} & \cdots & \cdots & x_{2,P} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N,1} & \cdots & \cdots & x_{N,P} \end{bmatrix} \quad \text{et} \quad D^T = [d_1 \quad \cdots \quad \cdots \quad d_P]$$

Où :

$$\left(\frac{\partial E}{\partial W} \right)^T = \left(\frac{\partial E}{\partial w_1} \frac{\partial E}{\partial w_2} \cdots \frac{\partial E}{\partial w_N} \right) = 0.0$$

- Minimisation d'un critère quadratique : annuler le gradient de l'erreur par rapport aux paramètres (les poids w_i)



Adaline (suite)

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \left(\|D^T - W^T X\|^2 \right) = \frac{\partial}{\partial w_i} \left((D^T - W^T X)(D^T - X^T W) \right)$$

$$\frac{\partial E}{\partial w_i} = 2(W^T X - D^T)X_i$$

Avec : $X_i = (x_{i,1} \cdots x_{i,P})$

$$\text{Minimum : } \left(\frac{\partial E}{\partial W} \right)^T = 0.0 \Rightarrow W = (X X^T)^{-1} D X$$

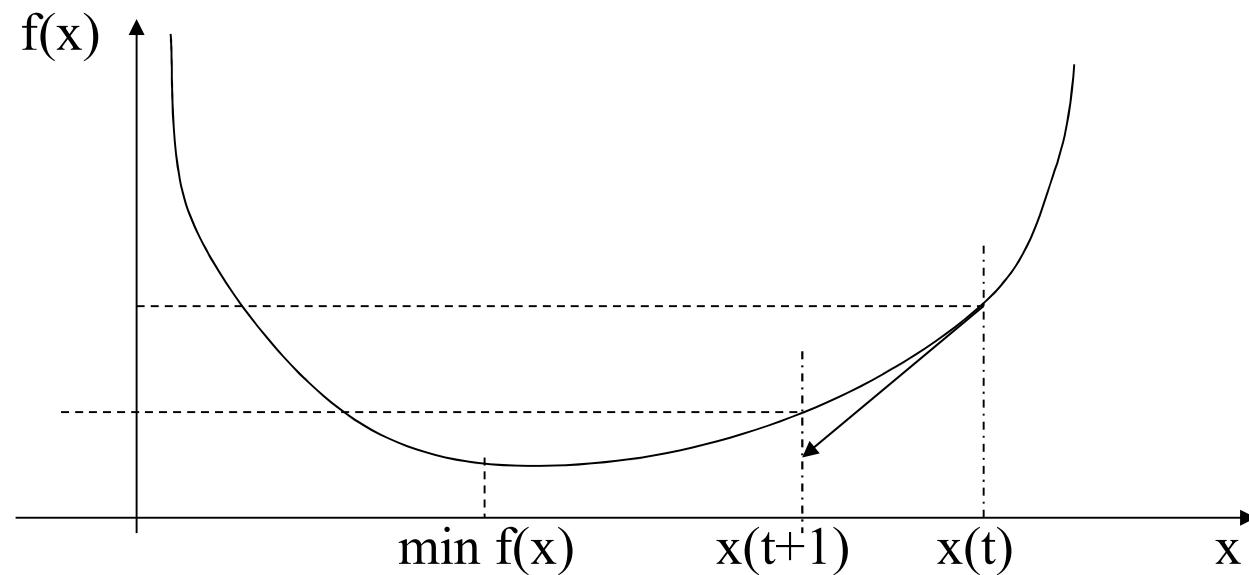
- Problème d'inversion d'une matrice souvent mal conditionnée
- Utilisation d'une méthode du gradient pour résoudre le problème

Méthode du gradient

- **Min f(x)**

$$x(t+1) = x(t) - \alpha \frac{\partial f}{\partial x} \Big|_{x(t)}$$

$t = n^{\circ}$ de l'itération



Méthode du gradient (exemple)

$$f(x) = (x + 2)^2 - 1$$

$$x(t+1) = x(t) - \alpha \frac{\partial f}{\partial x} \Big|_{x(t)}$$

$$\frac{\partial f}{\partial x} = 2(x + 2)$$

$$x(t+1) = x(t) - 2\alpha(x(t) + 2) = (1 - 2\alpha)x(t) - 4\alpha$$

Posons : $y(t) = x(t) + 2$

$$x(t+1) = -2 + (1 - 2\alpha)(y(t))$$

$$y(t+1) = (1 - 2\alpha)y(t)$$

$y(t)$ converge vers 0 ($x(t)$ vers -2) si $0 < \alpha < 0.5$ ($|1-2\alpha| < 1$)



Méthode du gradient (séquentielle)

$$E = \sum_{k=1}^P \left(d_k - W^T X_k \right)^2$$

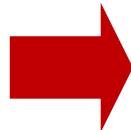
$$W(t+1) = W(t) - \alpha \frac{\partial E}{\partial W} \Big|_{W(t)}$$

$$W(t+1) = W(t) + 2\alpha X(D - X^T W)$$

Pseudo-inverse et méthode du gradient  même solution (même critère)

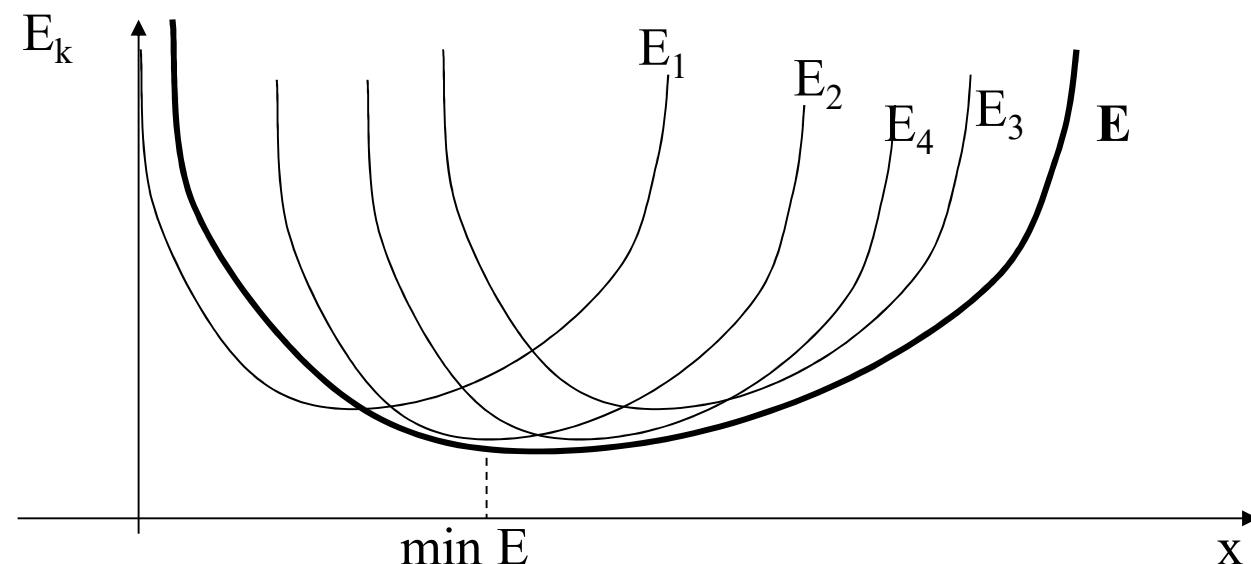
Remarque : E est une somme de terme positif

$$E = \sum_{k=1}^P \left(d_k - W^T X_k \right)^2 = \sum_{k=1}^P E_k$$



minimisation du critère E est équivalente à la minimisation successive des différents critères E_k

$$W(t+1) = W(t) + 2\alpha(d_k - W^T X_k)X_k$$

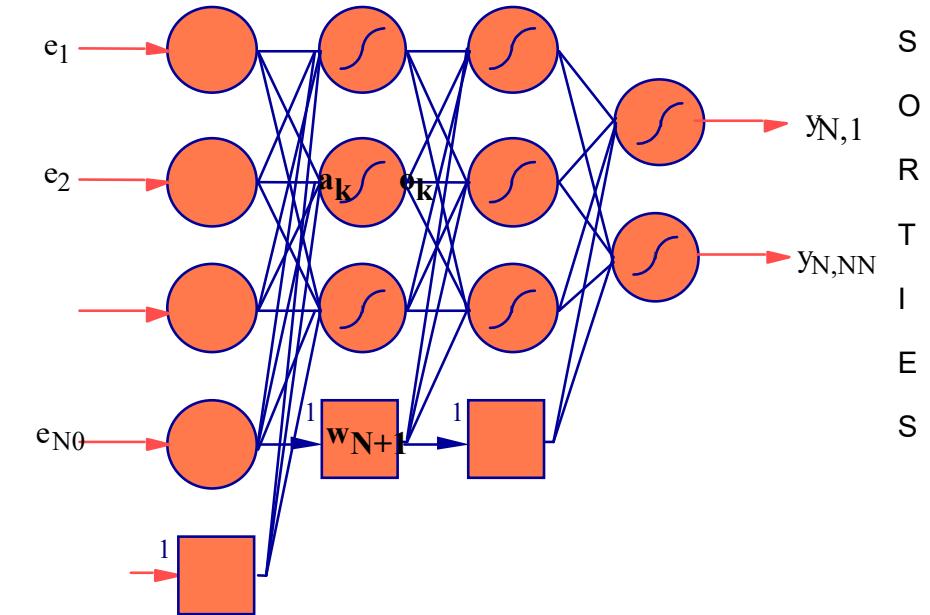


- Un réseau de neurones est une structure ordonnée de neurones connectés entre eux.
- Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle.
- Possibilités infinies
- Les structures les plus courantes

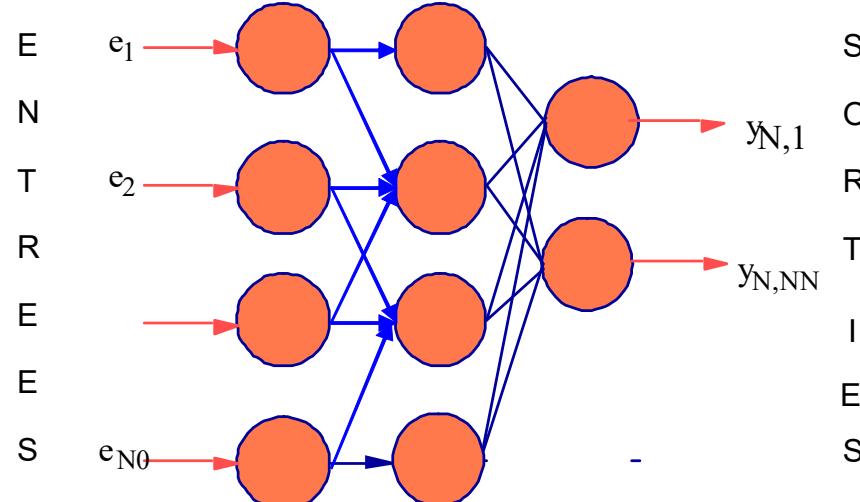


Le réseau multicouche

- Le réseau multicouche : les neurones sont arrangés par couche.
- Il n'y a pas de connexion entre neurones d'une même couche et les connexions se font avec les neurones des couches avales.
- Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci uniquement.
- Il y a donc un **sens de parcours** de l'information (de l'activation) au sein du réseau.
- **Définitions :**
 - Couche d'entrée : l'ensemble des neurones d'entrée
 - Couche de sortie : l'ensemble des neurones de sortie
 - Couches cachées : couches intermédiaires n'ayant aucun contact avec l'extérieur (entrée ou sortie)

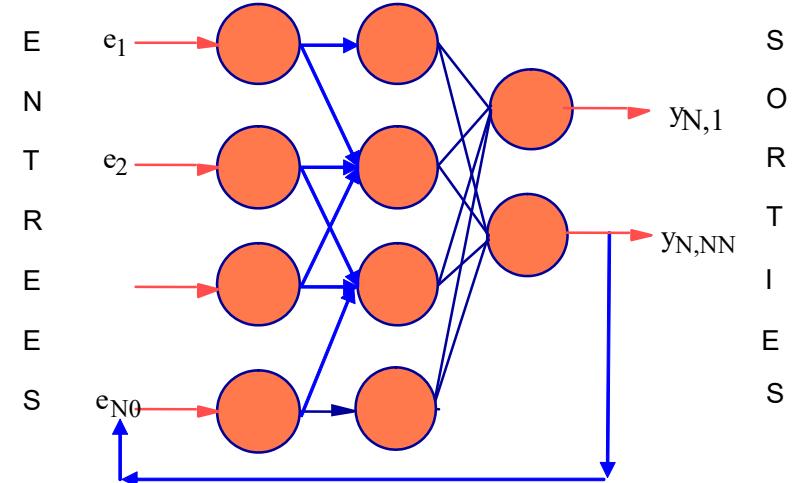


- C'est aussi un réseau multicouche mais qui possède une certaine topologie
- Chaque neurone entretient des relations avec un nombre limité et localisé de neurones de la couche avale.
- Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique



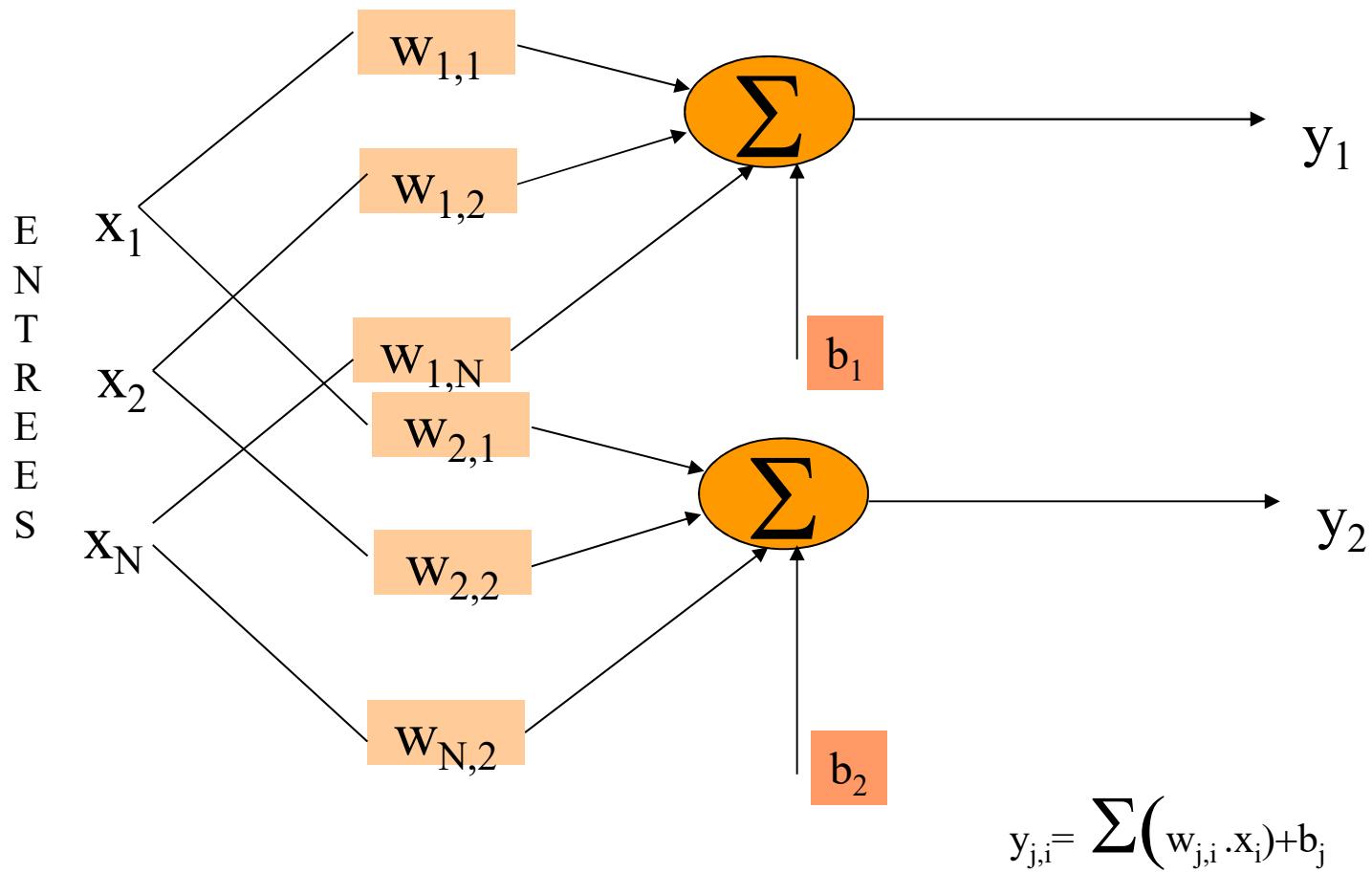
Le réseau à connexions récurrentes (réseaux récurrents)

- C'est aussi un réseau multicouche mais qui ramène l'information de sortie en arrière au sens de la propagation.
- Le plus souvent : la sortie du réseau devient une entrée du réseau
- Les connexions sont souvent locales
- Ces réseaux sont utilisés pour pouvoir modéliser des systèmes dynamiques (modélisation d'une relation récurrente entre la sortie et les sorties aux instants précédents)

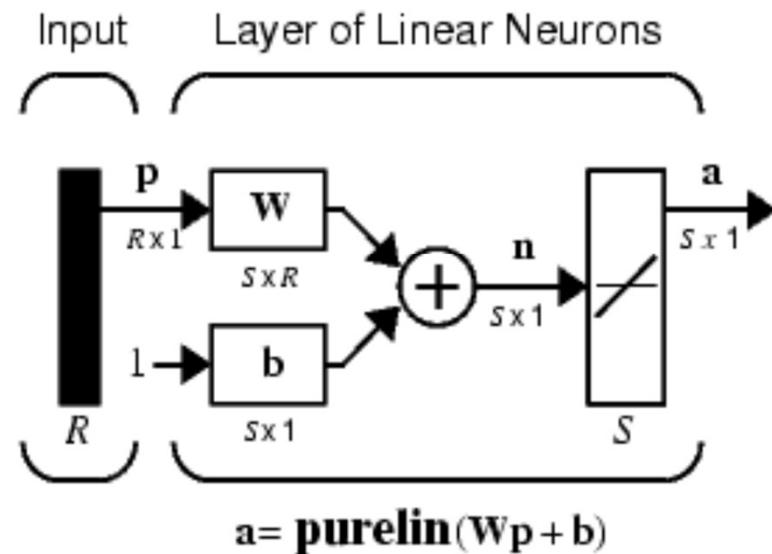
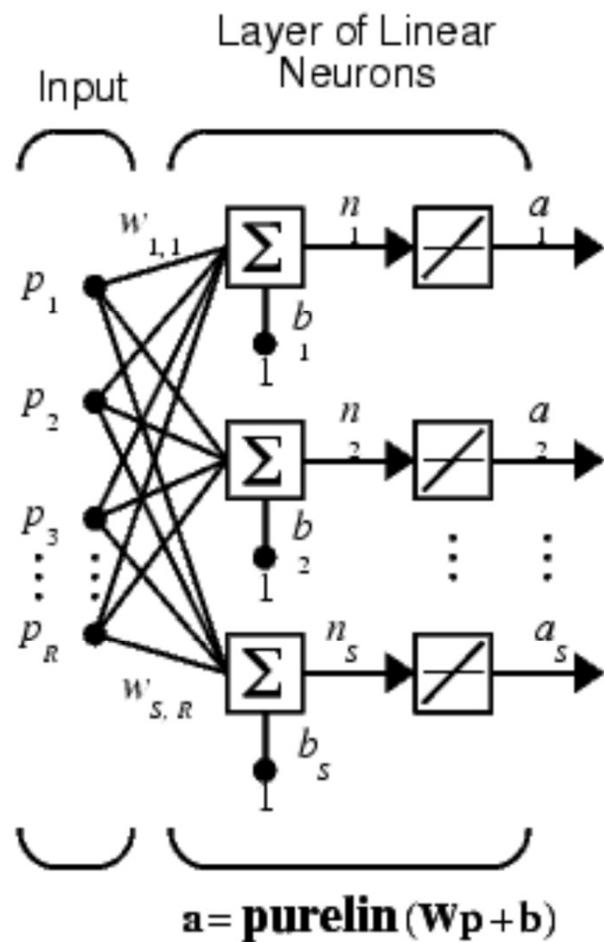


Le réseau Multiple ADaptive LINear Element (MADALINE)

Le réseau le plus simple constitué de neurones linéaires



Exemple Le réseau MADALINE (MATLAB)

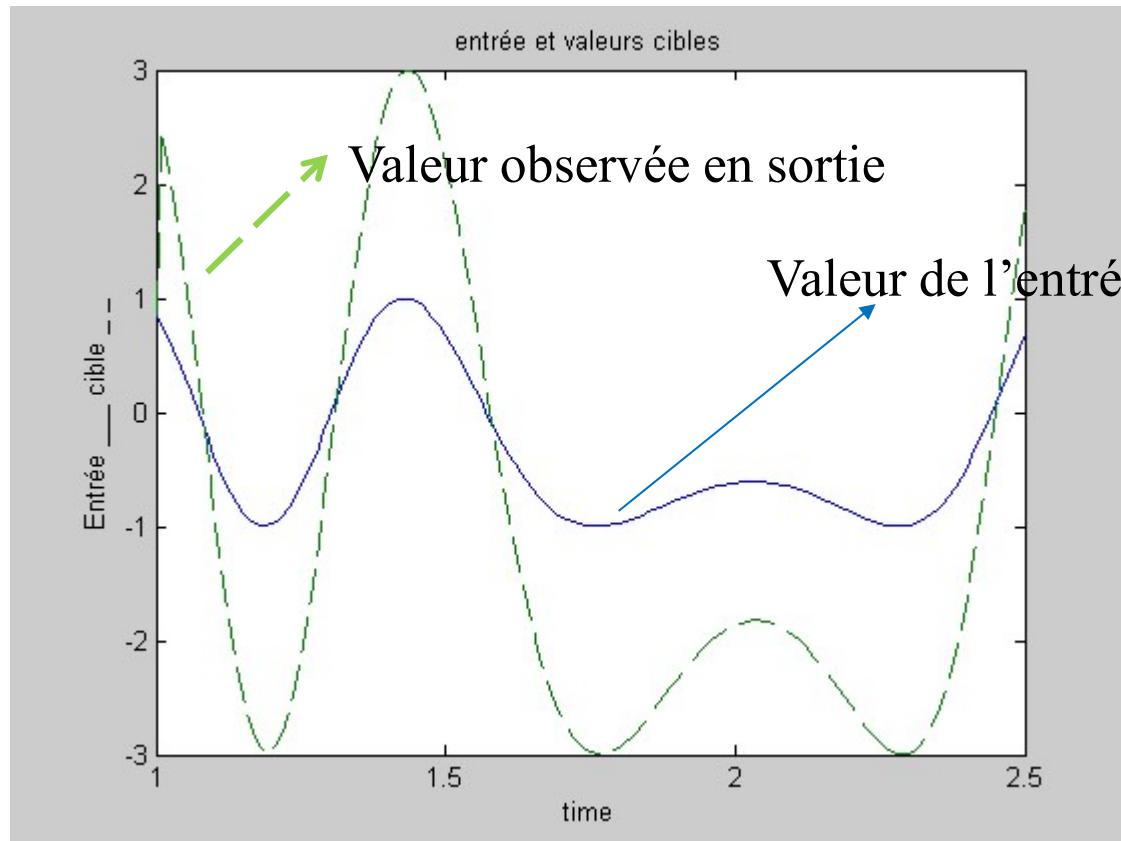


Where...

R = number of elements in input vector

S = number of neurons in layer

Exemple: prédiction



```
%élaboration du signal
time =1:0.01:2.5;
X=sin(sin(time).*time*10);
P=con2seq(X);
%décalage de 1
for i=1:150,
XX(i+1)=X(i);
end
XX(1)=0.0

% $T=2*P(i-1)+P$ 

T=con2seq(2*XX+X);
plot(time,
cat(2,P{:}),time,cat(2,T{:}),'--')
title('entrée et valeurs cibles')
xlabel('time')
ylabel('Entrée \_\_\_ cible \_\_')
pause;
```

Création d'un réseau 2 neurones d'entrée, 1 sortie

%création d'un réseau linéaire : [-3 3]= domaine de variation des entrées,
%second argument = 1 neurone, [0 1] : une entrée sans retard et une entrée
%avec retard de 1, alpha = 0.1

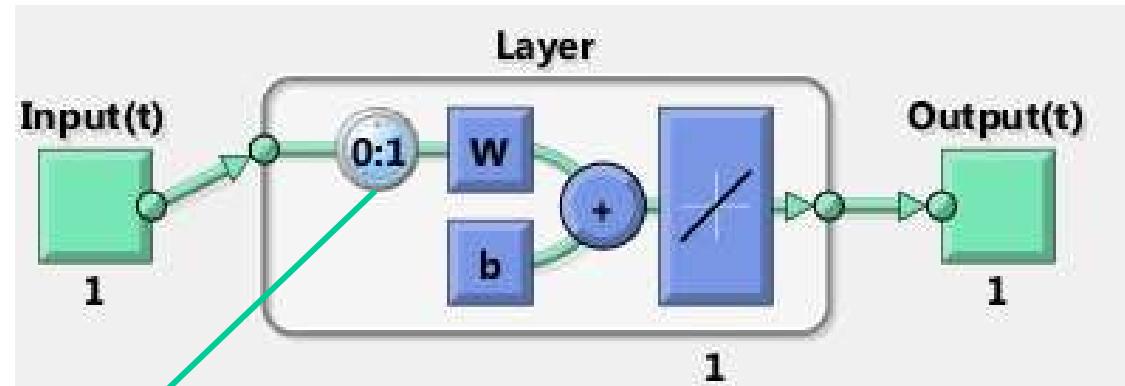
```
reseau=newlin([-3 3],1,[0 1],0.1)
pause;
[reseau,Y,E,Pf]=adapt(reseau,P,T)
pause;
plot(time,cat(2,Y{:}),'b',time,cat(2,T{:}),'r',time,cat(2,E{:}),'g',[1 2.5],[0 0],'k')
view(reseau)

poids=reseau.iw

biais=reseau.b
```

Création d'un réseau 2 neurones d'entrée, 1 sortie

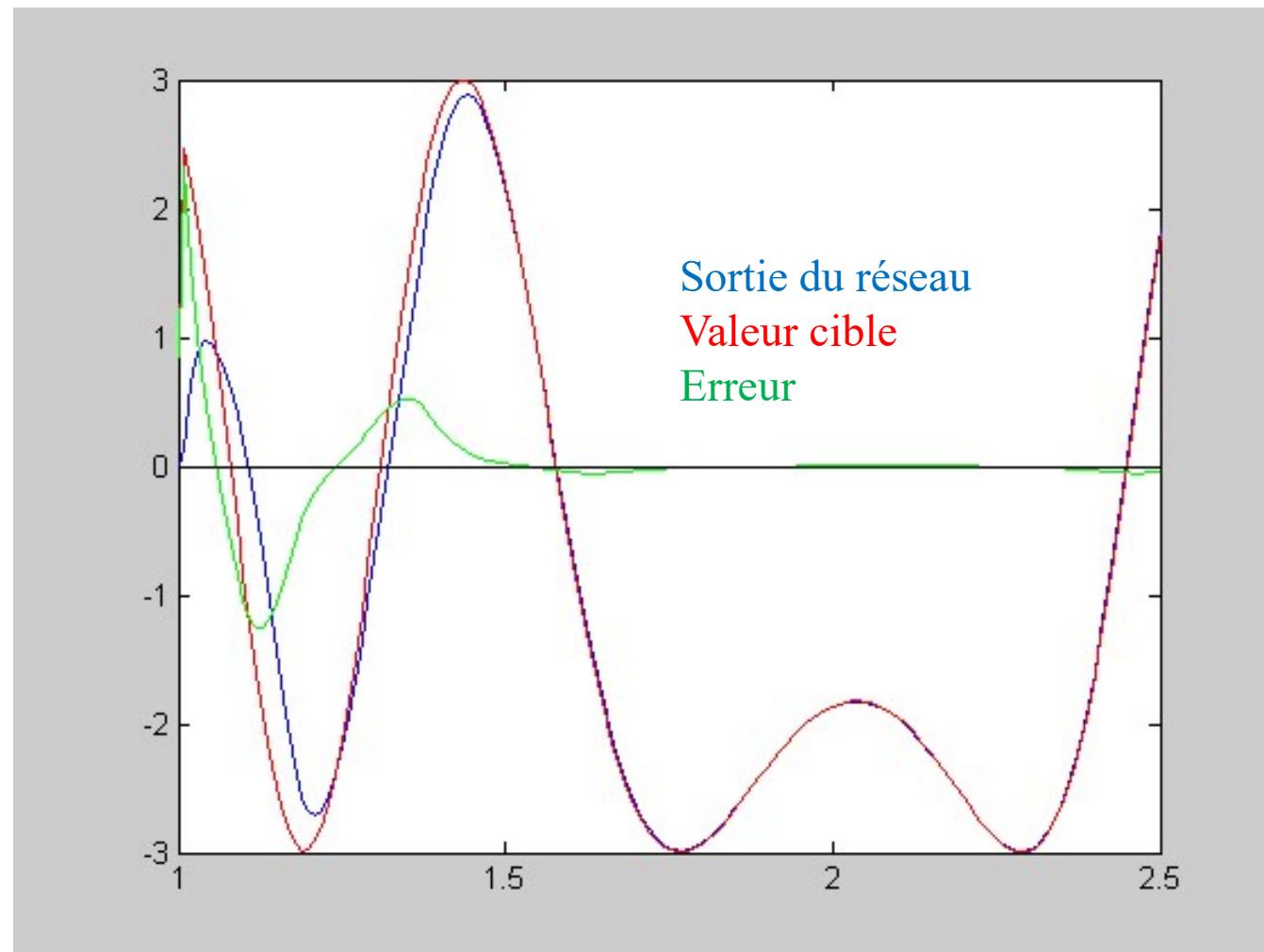
- Réseau construit



2 valeurs en entrée : 1 à l'instant t (0) et une à l'instant précédent (1)
 ➔ 2 poids
 ➔ w = poids=reseau.iw = [1,6061 1,4254]
 ➔ b=biais=reseau.b=-0,052



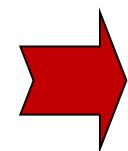
Résultats



Gaussienne : de loin la plus utilisée

$$\phi\{(e_j - c_j), \beta_j\} = \exp\left\{-\frac{1}{2} \frac{(e_i - c_j)^T (e_i - c_j)}{\sigma_j^2}\right\}$$

c_j est alors la moyenne et σ_j est la variance



Convient très bien à la classification : les centres représentent les centres des classes (centroïdes)

Pb : Les variances doivent être déterminées au même titre que les poids w_i

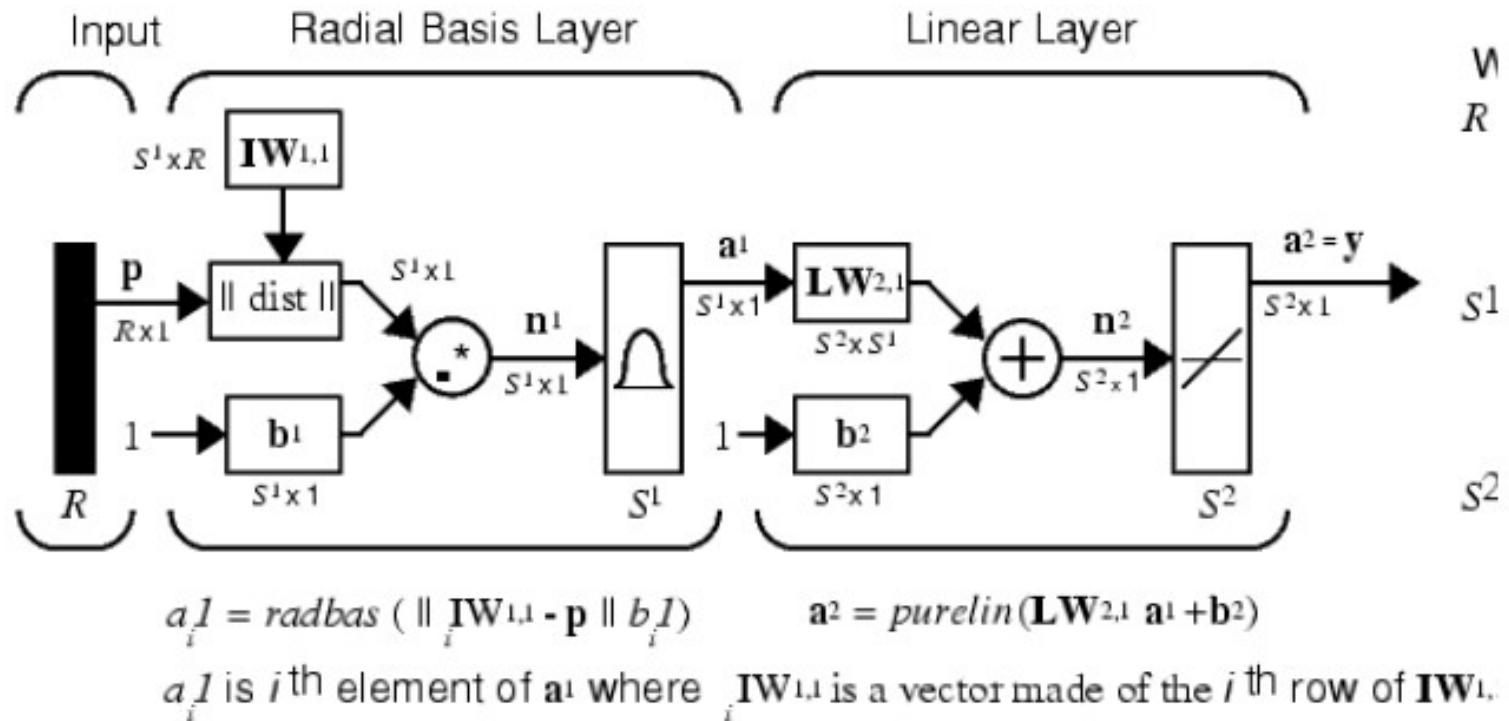
En pratique, il y a une procédure en deux temps :

- on fixe la variance sur les premiers points
- on détermine les poids (procédure linéaire) (voir ADALINE)
- on modifie suivant une relation de la forme :

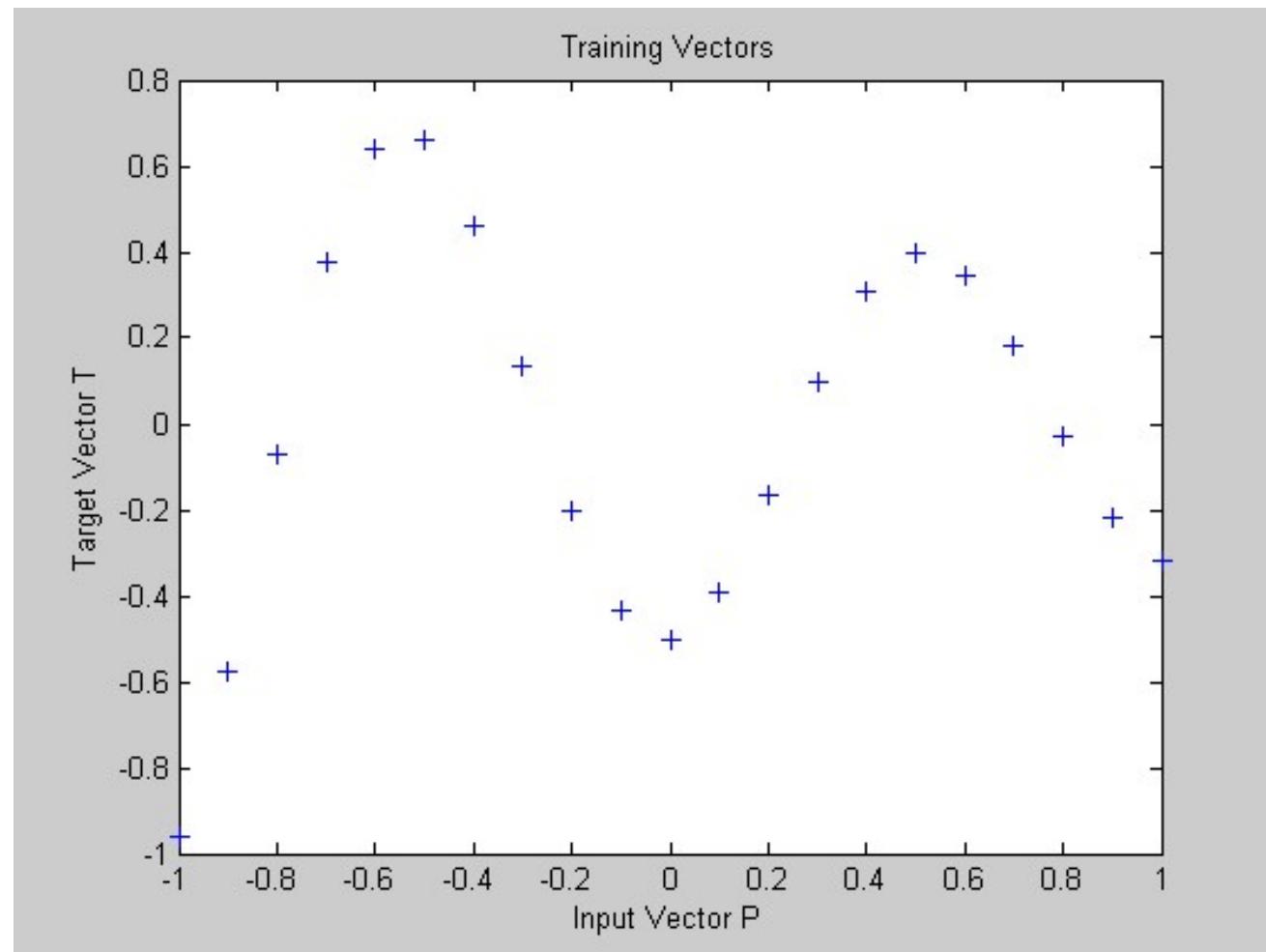
$$\sigma_j(k+1) = \sigma_j(k)(1 - g(k)) + g(k).2.\|e_j - c_i(k)\|$$



Exemple : les réseaux RBF dans Matlab



Exemple : modélisation



Fonction de base radiale

```
x = -3:.1:3;
```

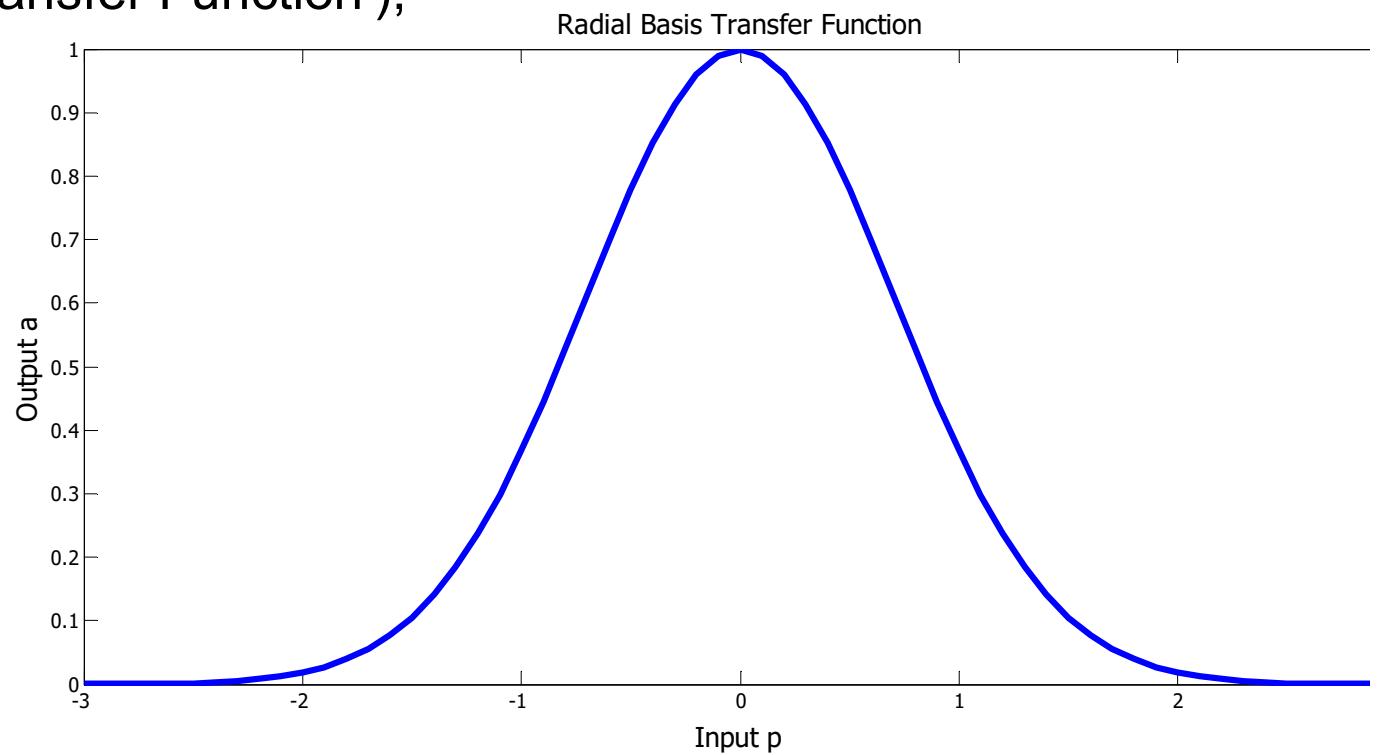
```
a = radbas(x);
```

```
plot(x,a)
```

```
title('Radial Basis Transfer Function');
```

```
xlabel('Input p');
```

```
ylabel('Output a');
```



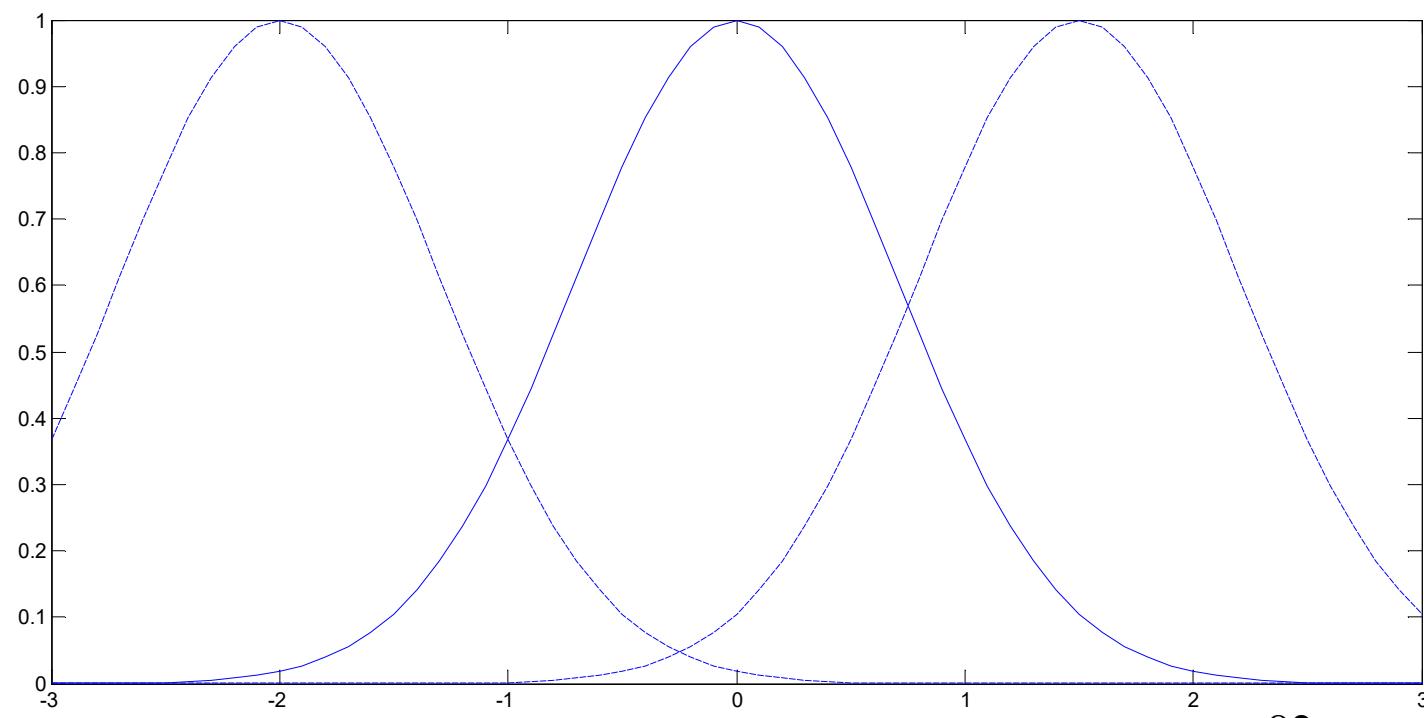
Fonction de base radiale

%2 autres fonctions de base centrées sur 1.5 et -2

a2 = radbas(x-1.5);

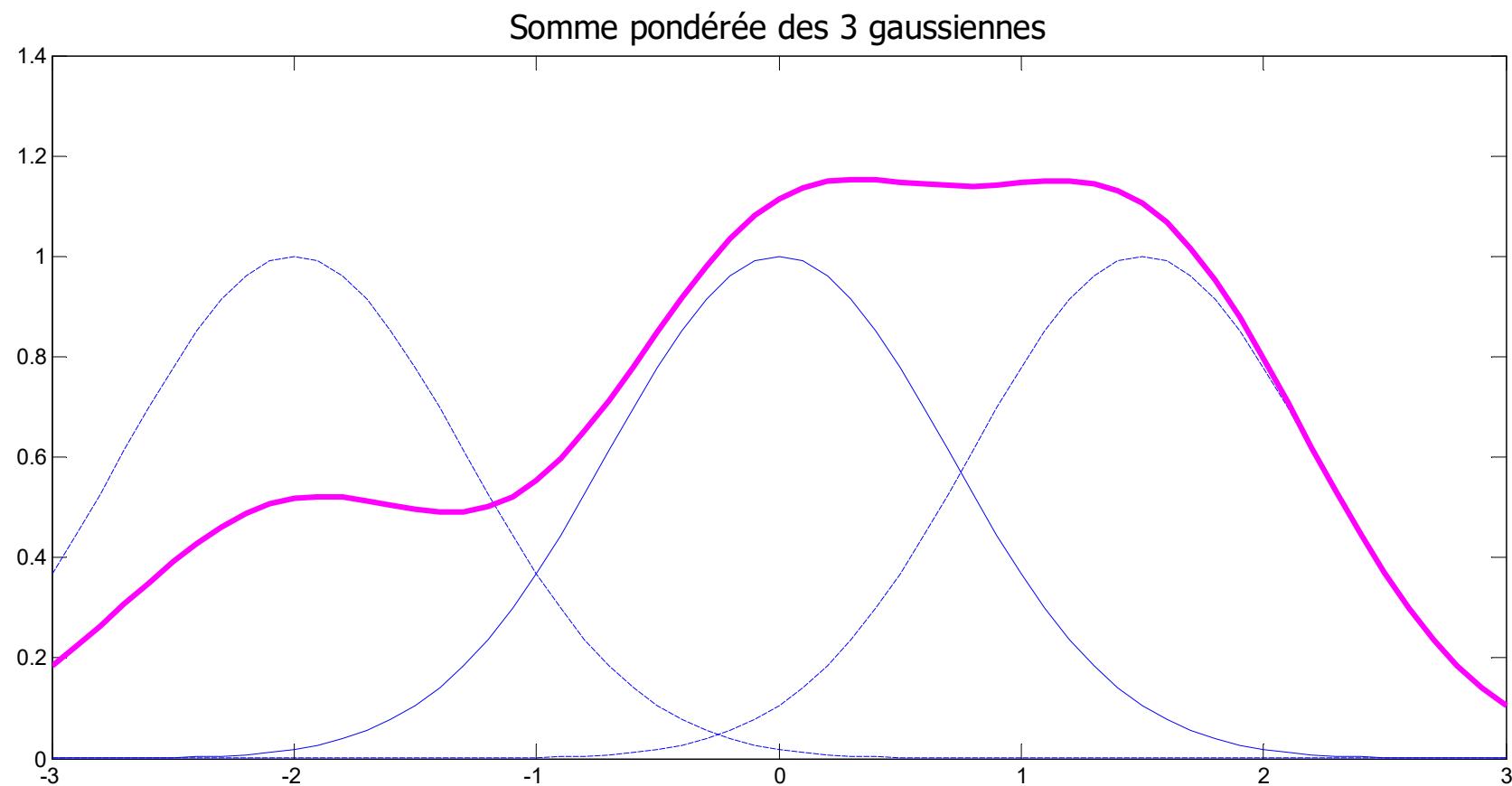
a3 = radbas(x+2);

plot(x,a,'b- ',x,a2,'b- ',x,a3,'b- ')



Fonction de base radiale

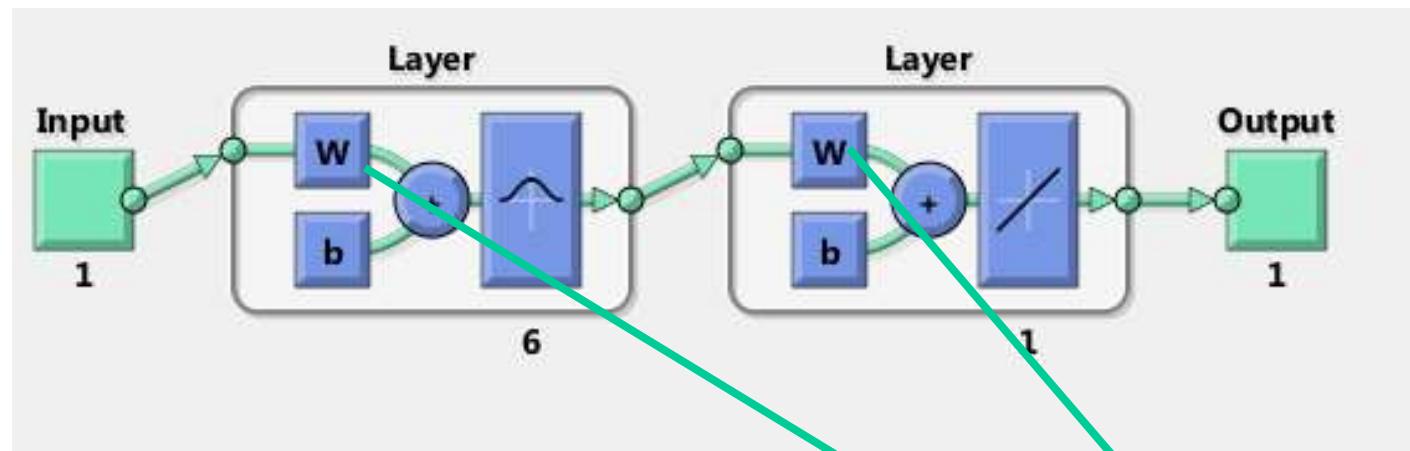
```
a4 = a + a2*1 + a3*0.5;  
plot(x,a,'b-',x,a2,'b--',x,a3,'b--',x,a4,'m-')
```



eg = 0.02; % somme des carrés des erreurs

sc = 1; % ecart-type gaussienne

rbf = newrb(X,T,eg,sc);



NEWRB, neurons = 0, MSE = 0.176192

NEWRB, neurons = 2, MSE = 0.160368

NEWRB, neurons = 3, MSE = 0.128338

NEWRB, neurons = 4, MSE = 0.0275185

NEWRB, neurons = 5, MSE = 0.0264878

NEWRB, neurons = 6, MSE = 0.00046188



Part avec aucun
neurone dans la
première
couche
et incrémenté
pour obtenir la
précision
demandée

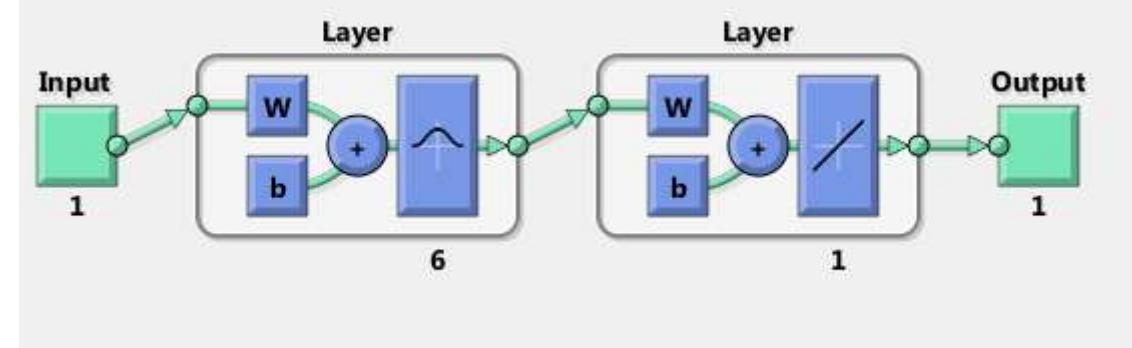
-1	9.99e4
-0,9	- 3.96e5
-0,8	5.94e5
-0,7	-4.01e5
1	-123
-0,6	1.03e5

Modification de l'écart-type

eg = 0.02;

SC = 3; % ecart-type gaussienne

rbf = newrb(X,T,eg,sc);



RBF2

NEWRB, neurons = 0, MSE = 0.176192
 NEWRB, neurons = 2, MSE = 0.159307
 NEWRB, neurons = 3, MSE = 0.15239
 NEWRB, neurons = 4, MSE = 0.0376415
 NEWRB, neurons = 5, MSE = 0.0353498
 NEWRB, neurons = 6, MSE = 0.00244785

POIDS
Première couche

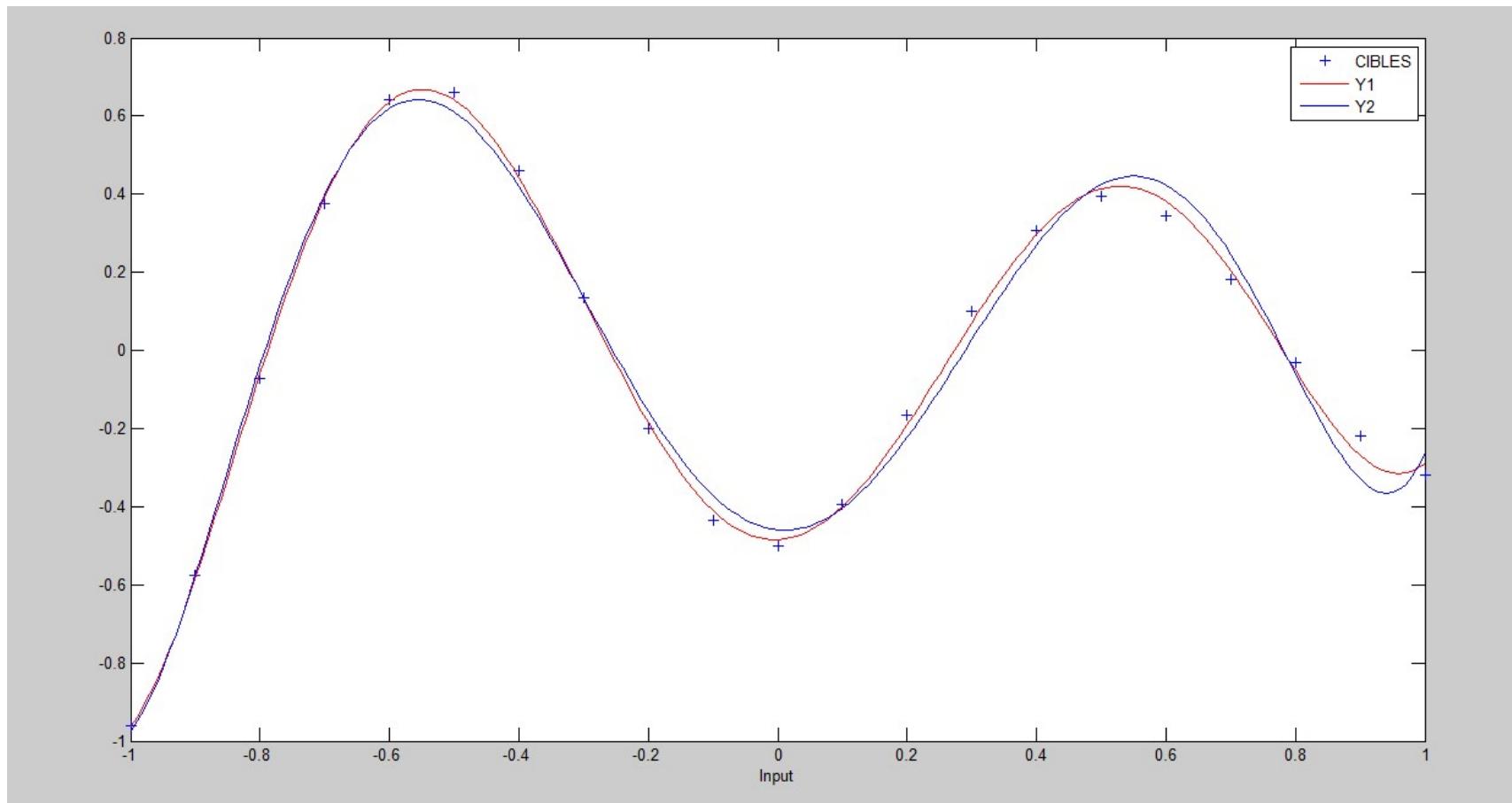
-1	-1
-0,9	-0,9
-0,8	-0,8
-0,7	-0,7
-0,6	1
1	-0,6

POIDS
Deuxième couche

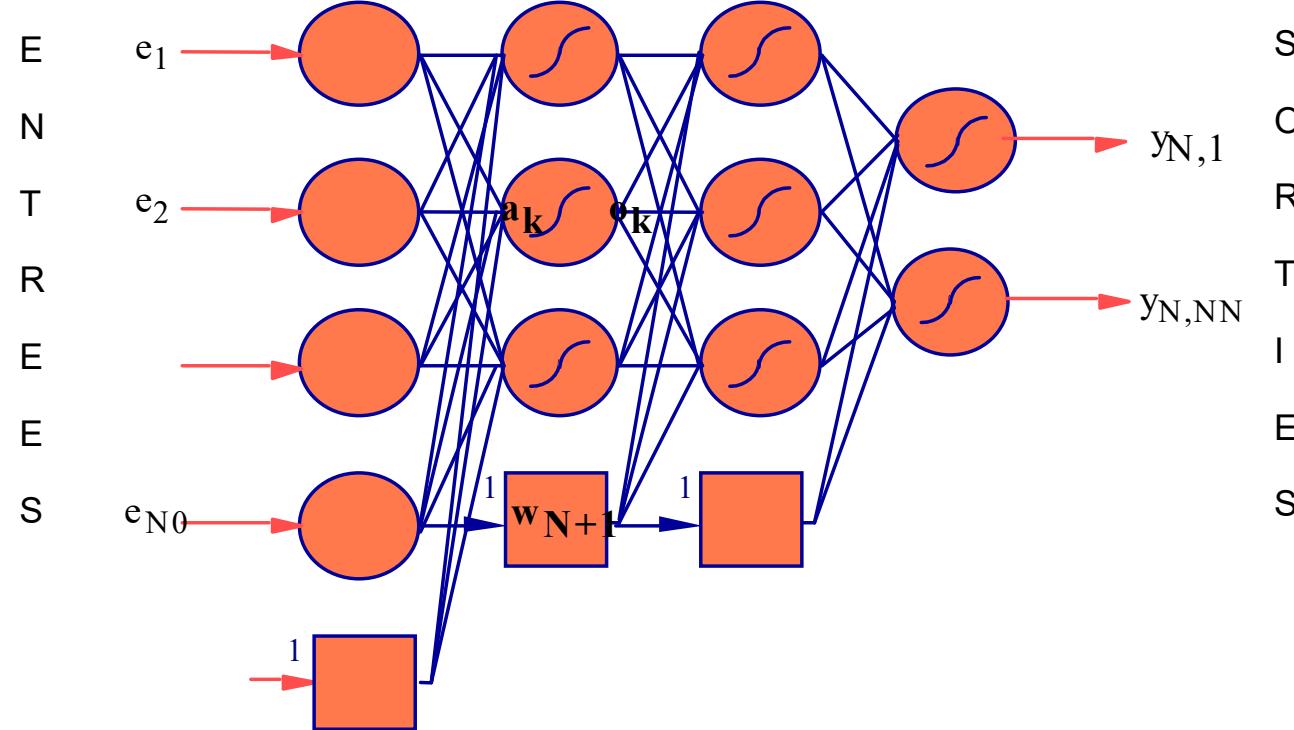
NEWRB, neurons = 0, MSE = 0.176192
 NEWRB, neurons = 2, MSE = 0.160368
 NEWRB, neurons = 3, MSE = 0.128338
 NEWRB, neurons = 4, MSE = 0.0275185
 NEWRB, neurons = 5, MSE = 0.0264878
 NEWRB, neurons = 6, MSE = 0.00046188

5e9	9.99e4
-2e10	-3.96e5
3.8e10	5.94e5
-2.8e10	-4.01e5
7.7e9	-123
-3e6	1.03e5

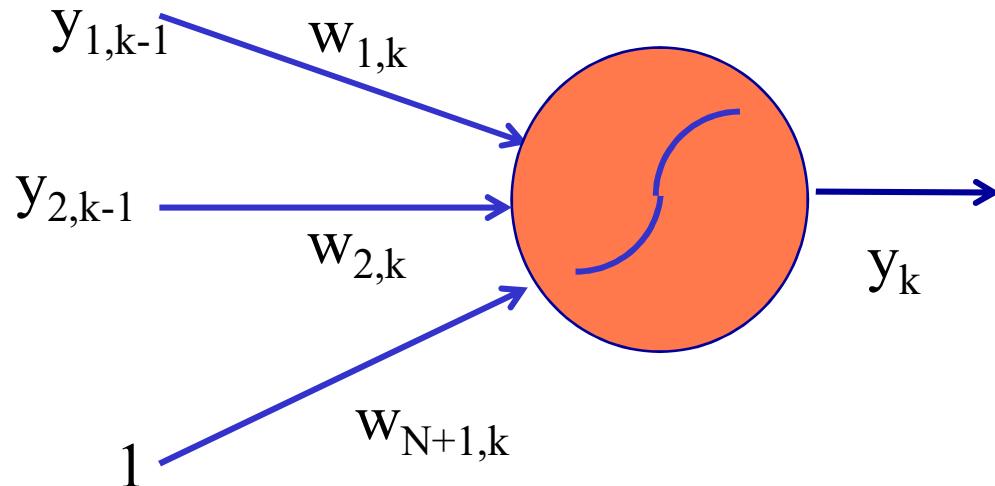
Comparaison des deux réseaux



Le Perceptron Multicouche (MLP)



Cellule de calcul élémentaire



$$a_k = \sum_{j=1}^N w_{j,k} y_{j,k-1} + w_{N+1,k}$$

$$y_k = \frac{1}{1 + \exp(a_k)}$$

Fonction sigmoïde

- $i_{j,k}$ est l'information reçue par le neurone k en provenance du neurone j de la couche précédente
- $w_{j,k}$ est le “poids” de la connexion entre le neurone k et le neurone j de la couche précédente
- w_{N+1} est appelé “biais” et permet d’ajouter un terme constant à la somme



Apprentissage d'un réseau supervisé multicouche

→ Un problème d'identification de paramètres

$$\text{Min}_{w_{ij}} \ ep = \frac{1}{2} \sum_{l=1}^{Nd} \sum_{k=1}^{NN} (d_k(l) - y_{k,N}(l))^2$$

Nombre d'exemples
dans la base d'apprentissage

Nombre de sorties
du réseau

valeurs des
sorties
désirées
(cibles)

valeurs des sorties calculées
par le Réseau (3 couches :
1 couche d'entrée,
une couche cachée,
une couche de sortie)

Méthode d'optimisation itérative : $w_{j,i}(n) = w_{j,i}(n-1) + \alpha \Delta w_{j,i}(n)$

$\Delta w_{ij}(i)$ = fonction $\frac{\partial ep}{\partial w_{ij}}$
de

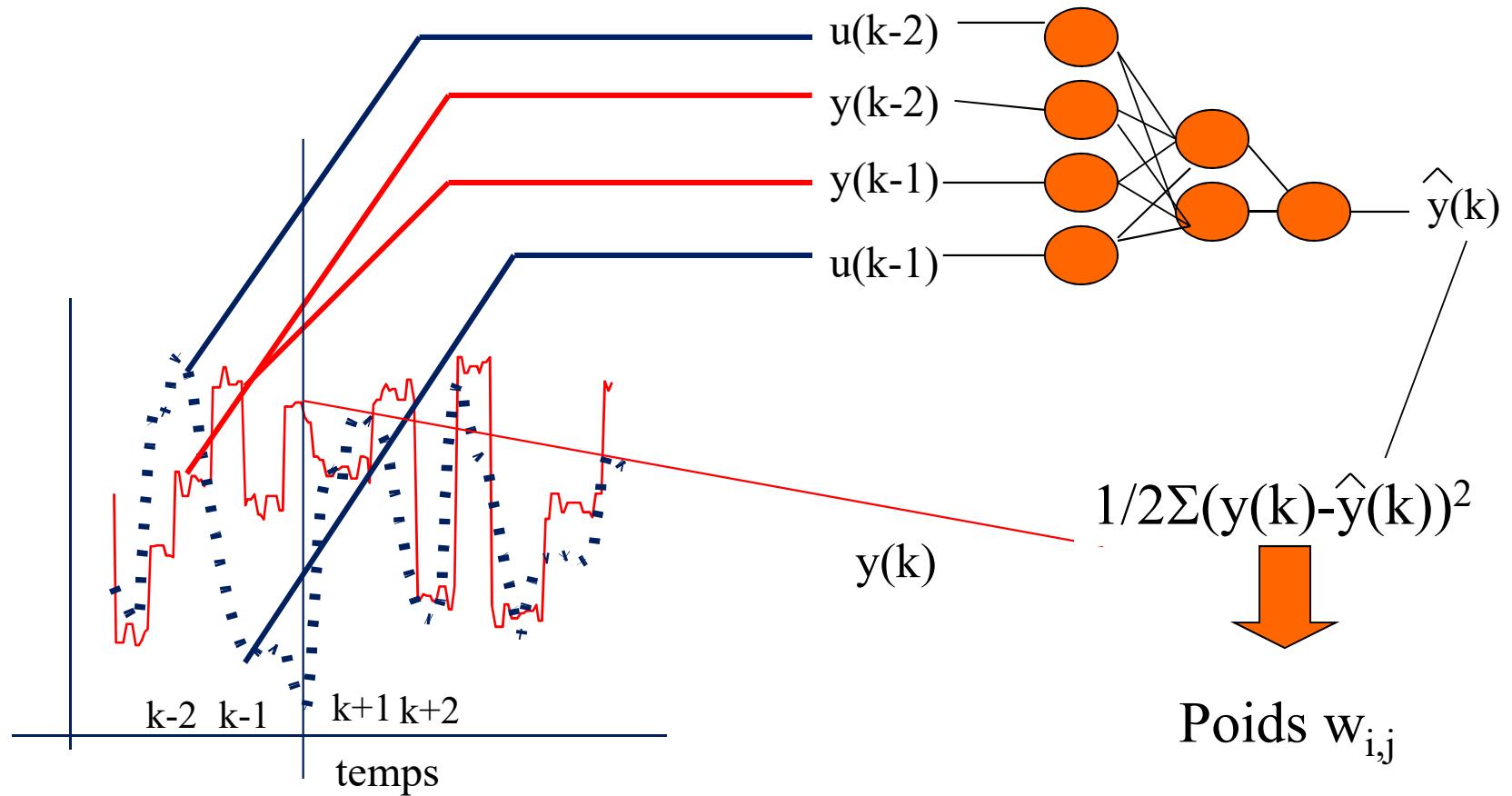
Gradient, Quasi-Newton

Le réseau de neurones pour la modélisation et la commande

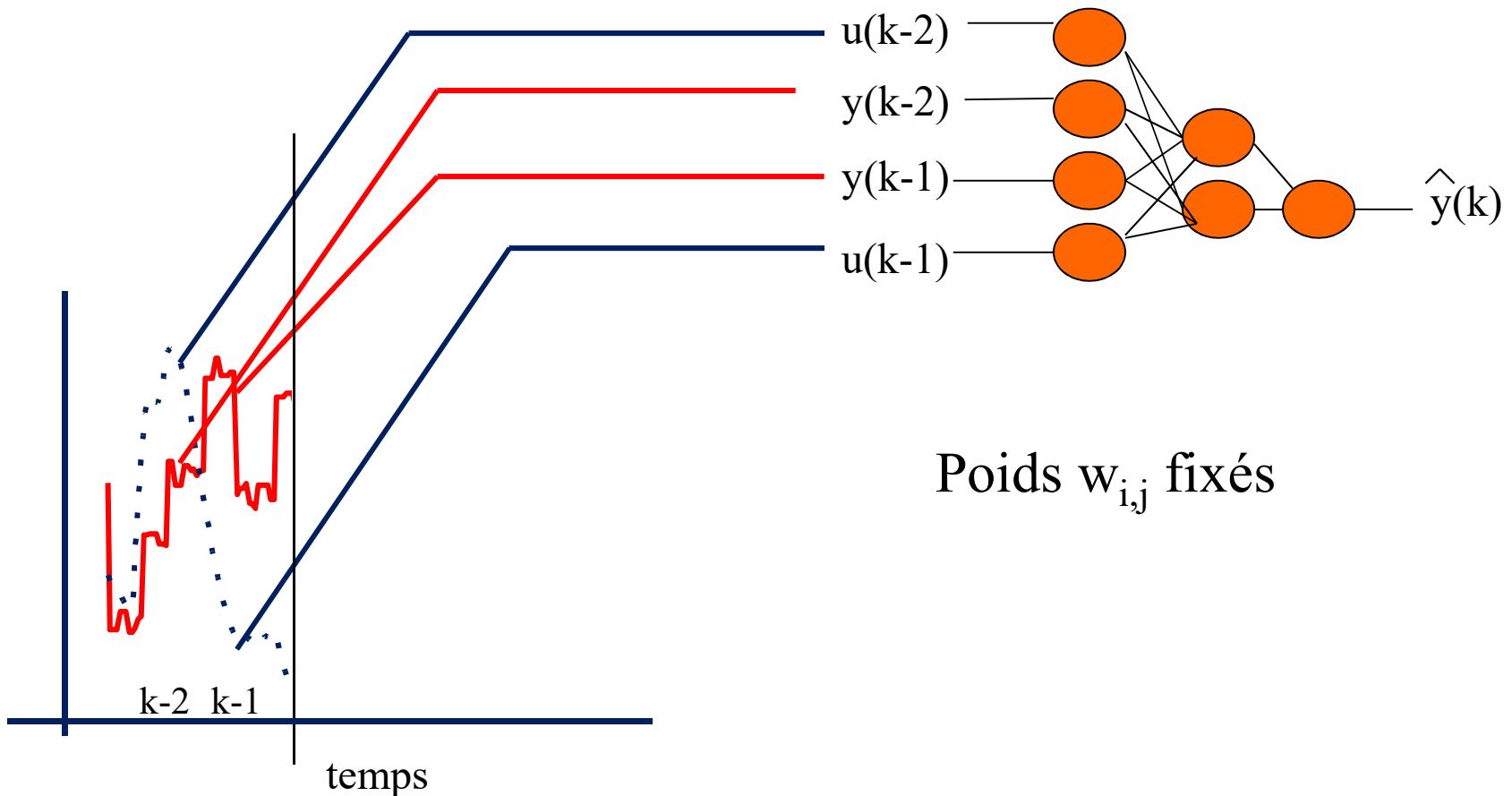
- Le réseau modèle – phase d'apprentissage
- Le réseau modèle – prédicteur à un pas
- Le réseau modèle (prédicteur à plusieurs pas) – phase d'apprentissage
- Le réseau modèle (prédicteur à plusieurs pas) – utilisation en ligne
- Commande prédictive à base de modèle neuronal
- Le réseau modèle inverse – phase d'apprentissage
- Le réseau modèle inverse– contrôleur en ligne



Le réseau modèle – phase d'apprentissage

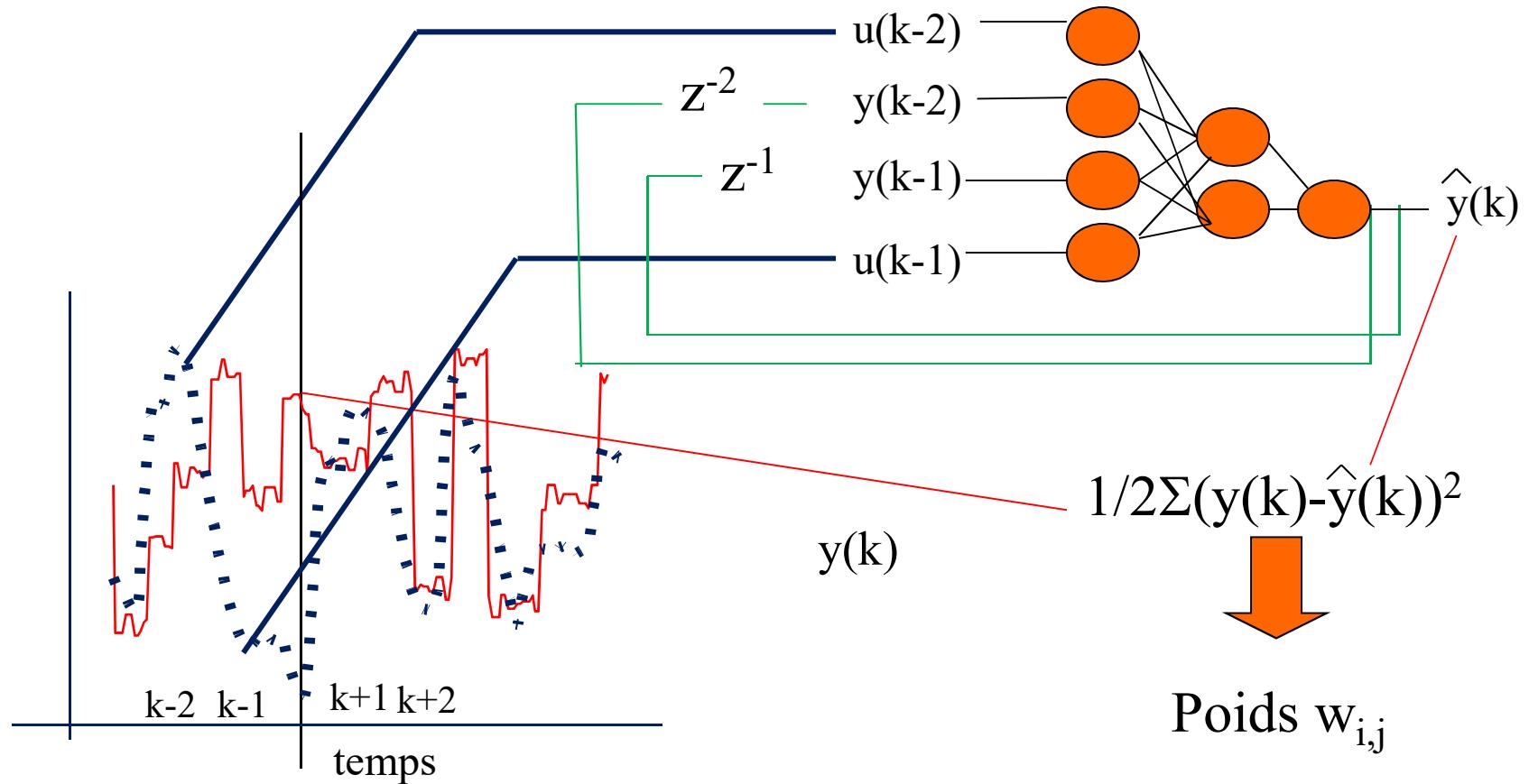


Le réseau modèle - prédicteur à un pas

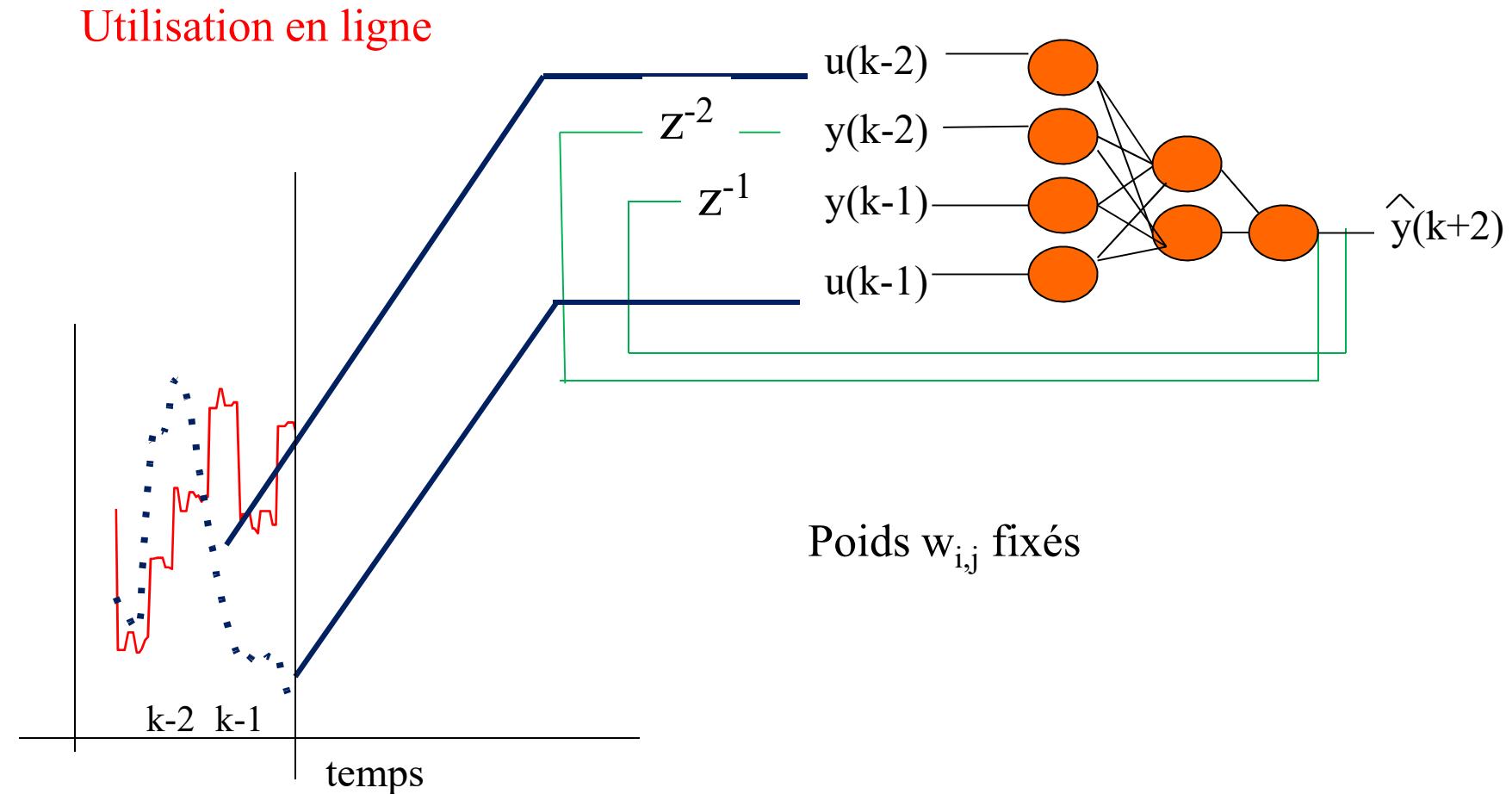


Le réseau modèle (prédicteur à plusieurs pas) – phase d'apprentissage

Phase d'apprentissage



Le réseau modèle (prédicteur à plusieurs pas) – utilisation en ligne



Le réseau de neurones pour la modélisation et la commande

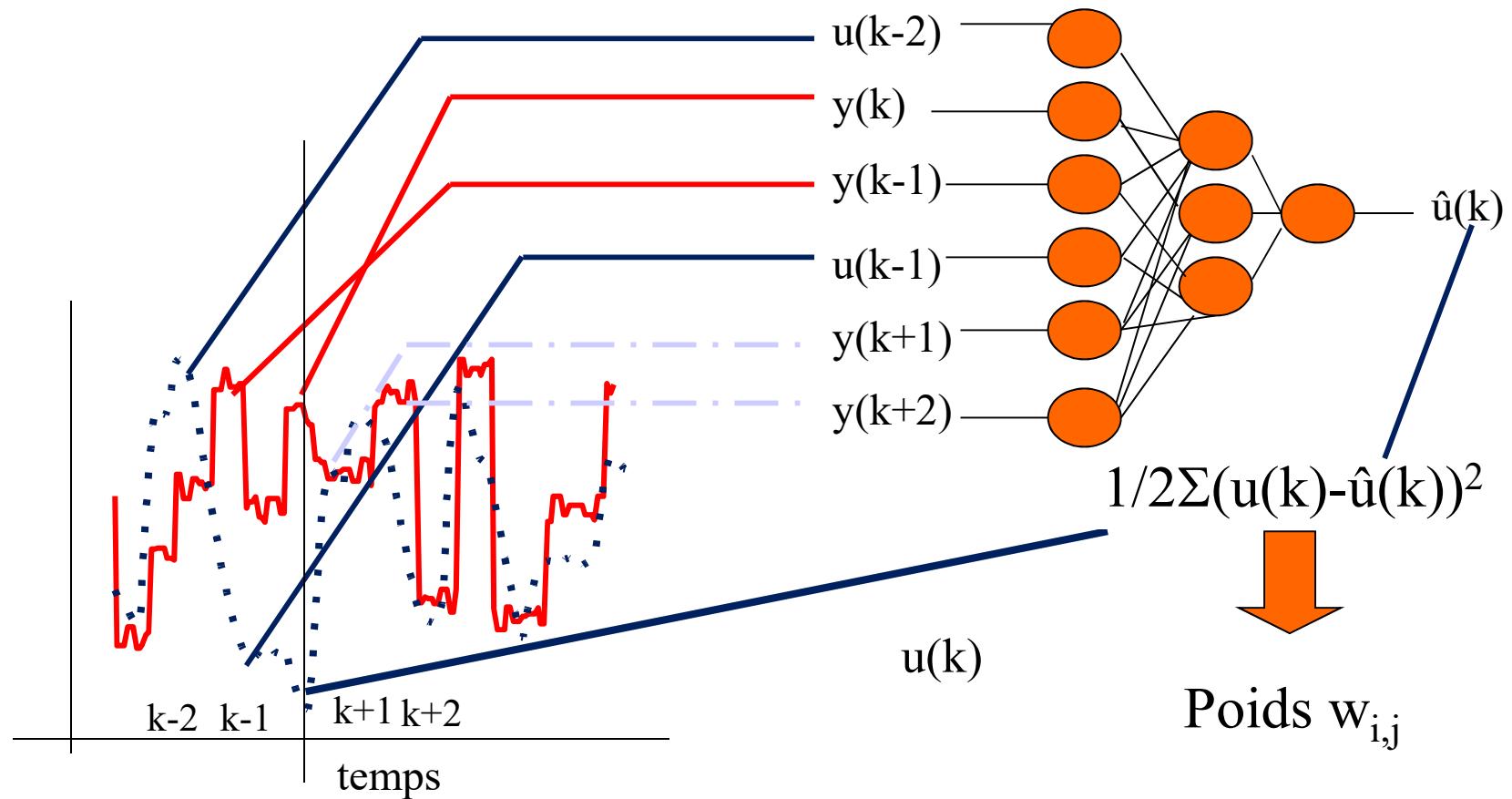


- Le réseau modèle – phase d'apprentissage
- Le réseau modèle – prédicteur à un pas
- Le réseau modèle (prédicteur à plusieurs pas) – phase d'apprentissage
- Le réseau modèle (prédicteur à plusieurs pas) – utilisation en ligne
- Le réseau modèle inverse – phase d'apprentissage
- Le réseau modèle inverse – contrôleur en ligne



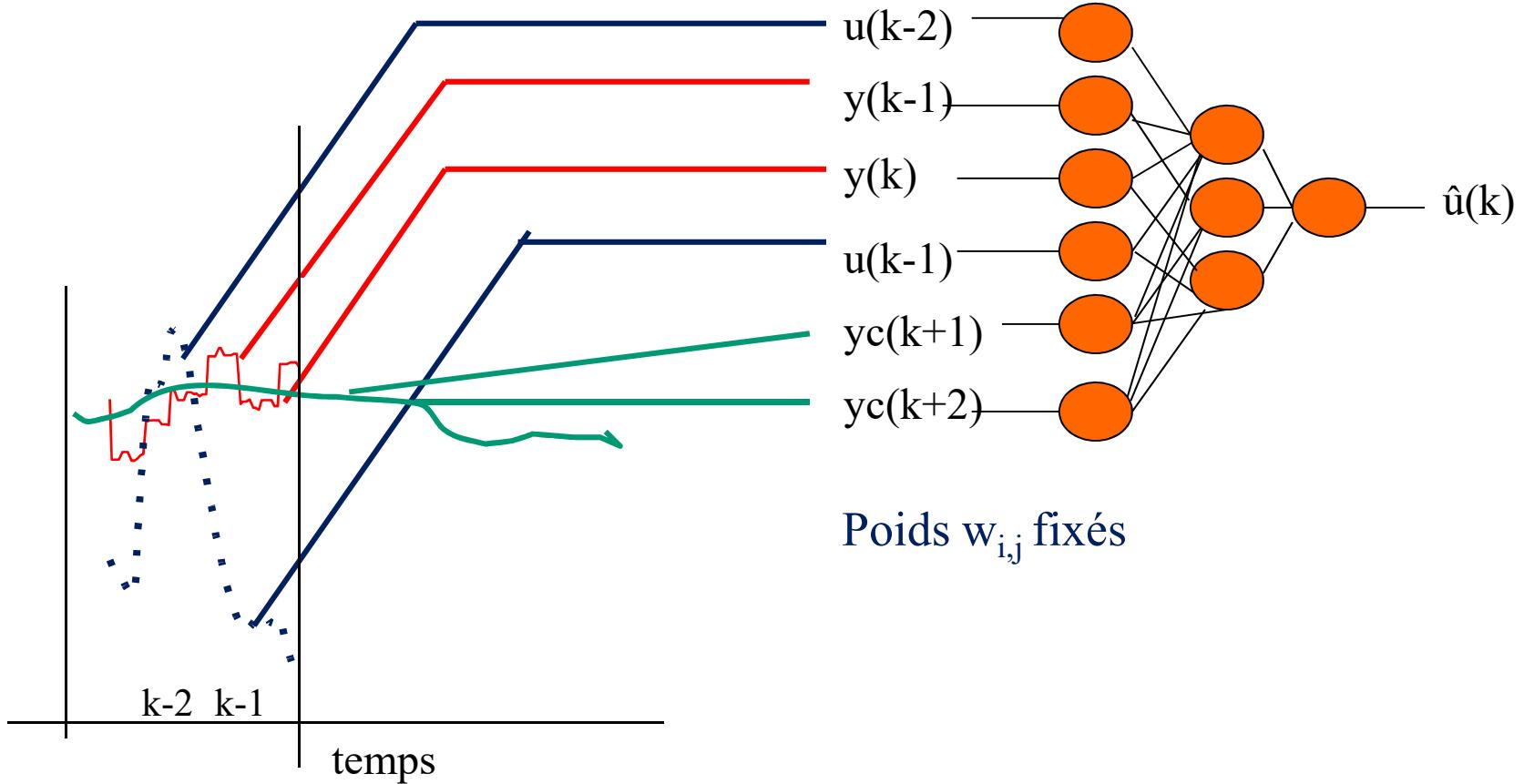
Le réseau modèle inverse – phase d'apprentissage

Phase d'apprentissage



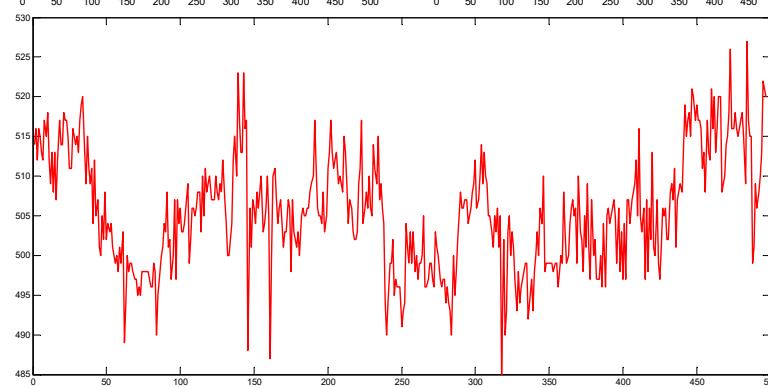
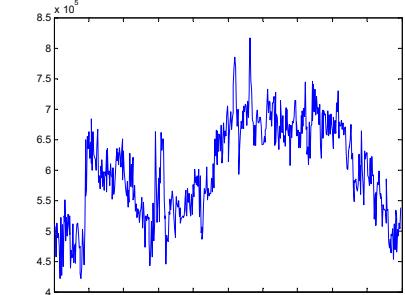
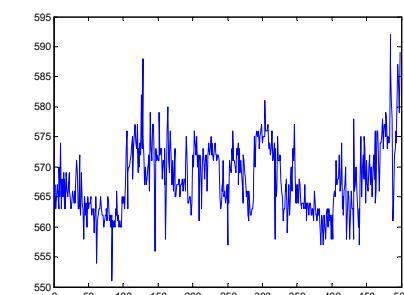
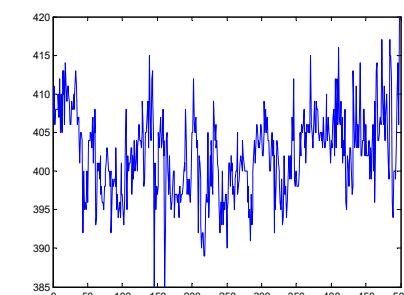
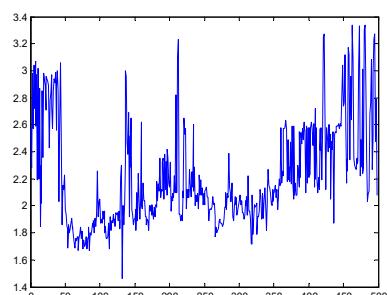
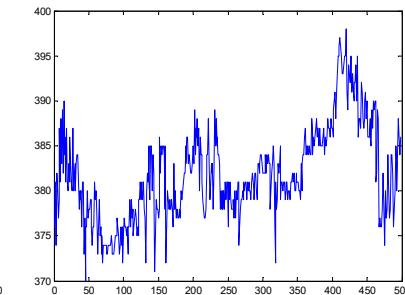
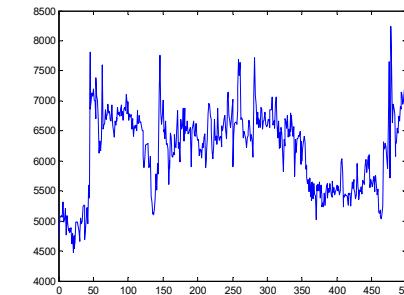
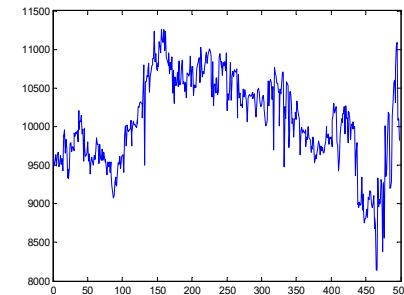
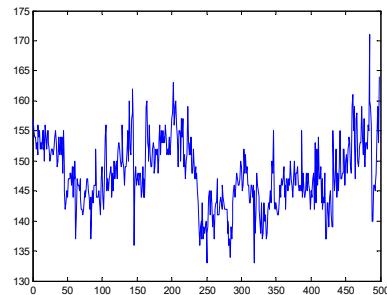
Le réseau modèle inverse – contrôleur en ligne

Contrôleur en ligne



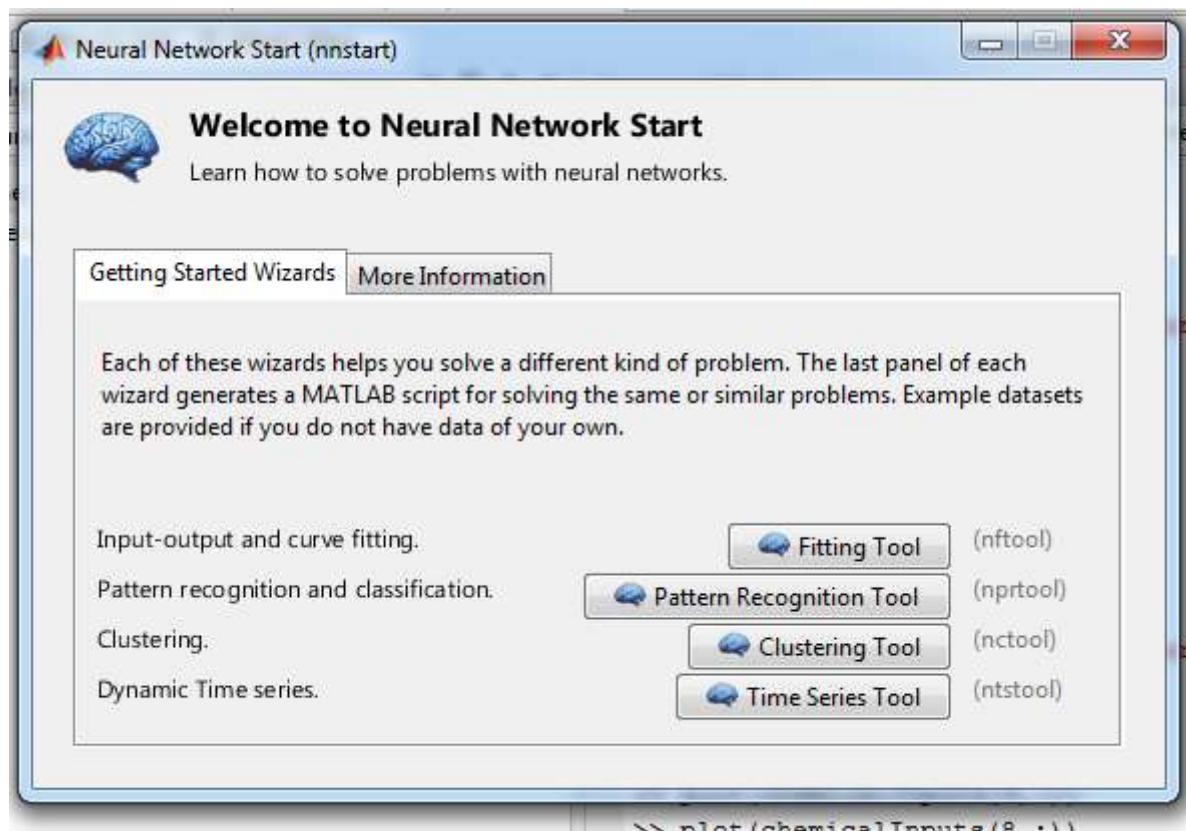
- Capteurs chimiques : analyse de composés chimiques
(Matlab), développement d'un capteur logiciel

Exemple : capteurs chimiques

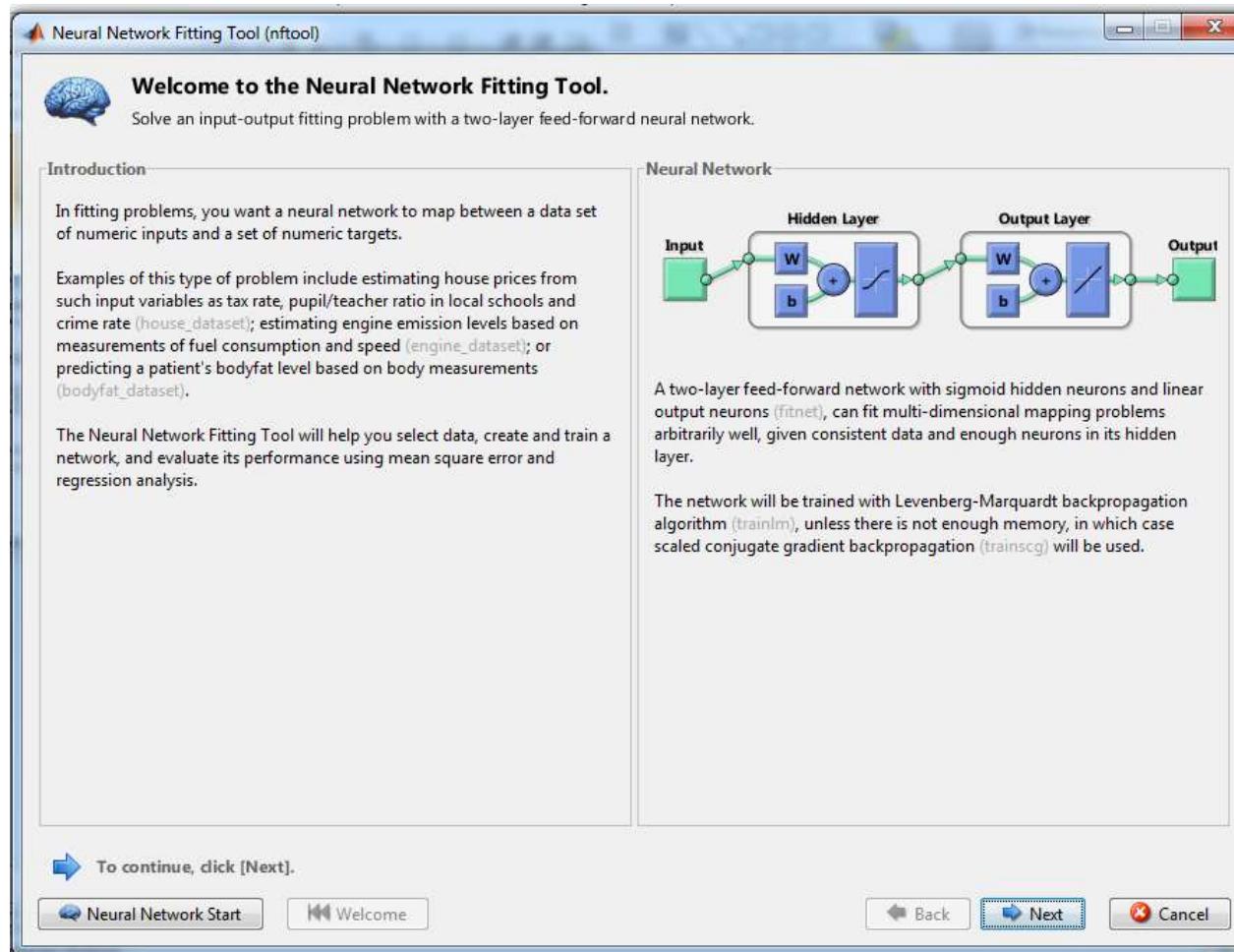


Exemple : capteurs chimiques

- Utilisation de la toolbox NNMatlab : nnstart

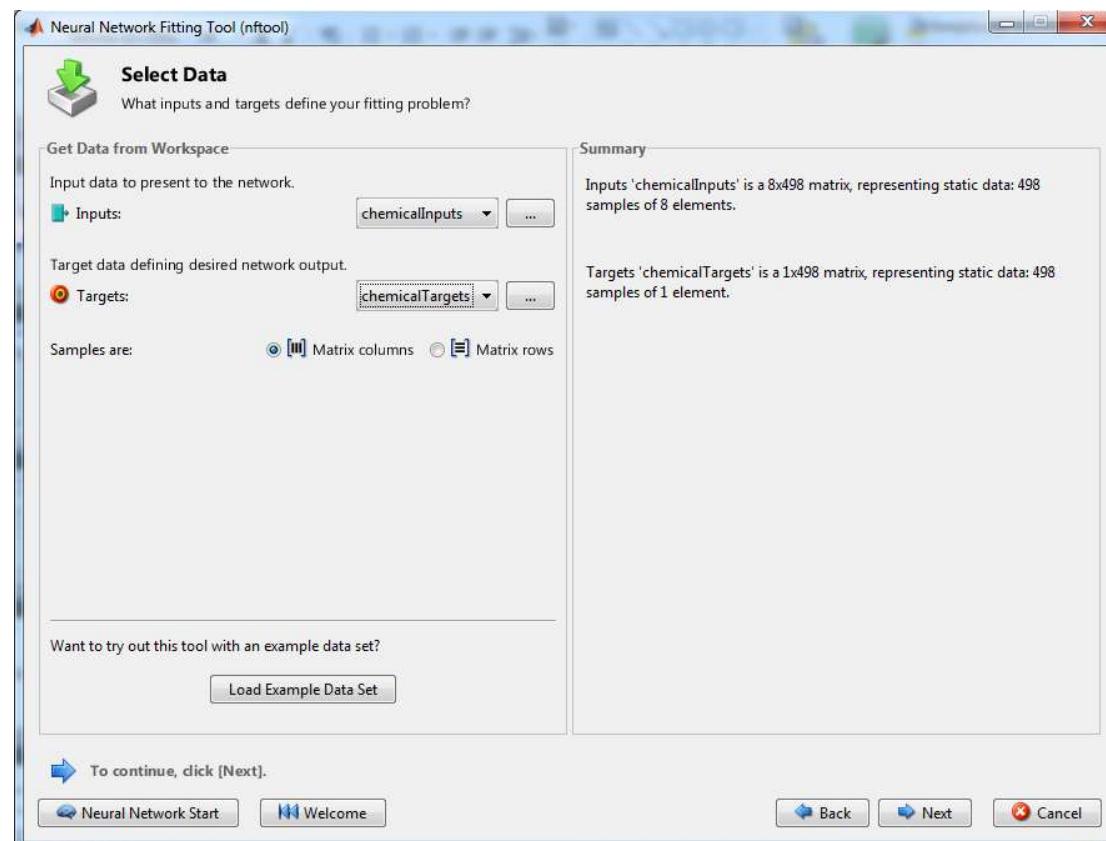


Exemple : capteurs chimiques



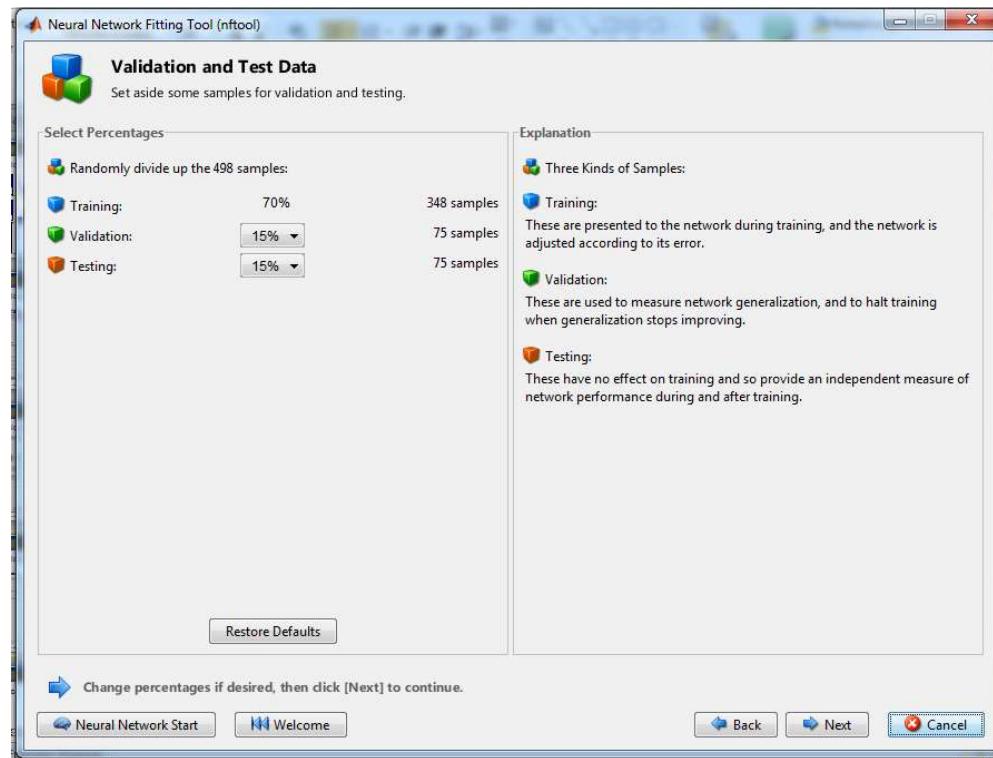
Exemple : capteurs chimiques

- Sélection des fichiers d'entrées et des cibles



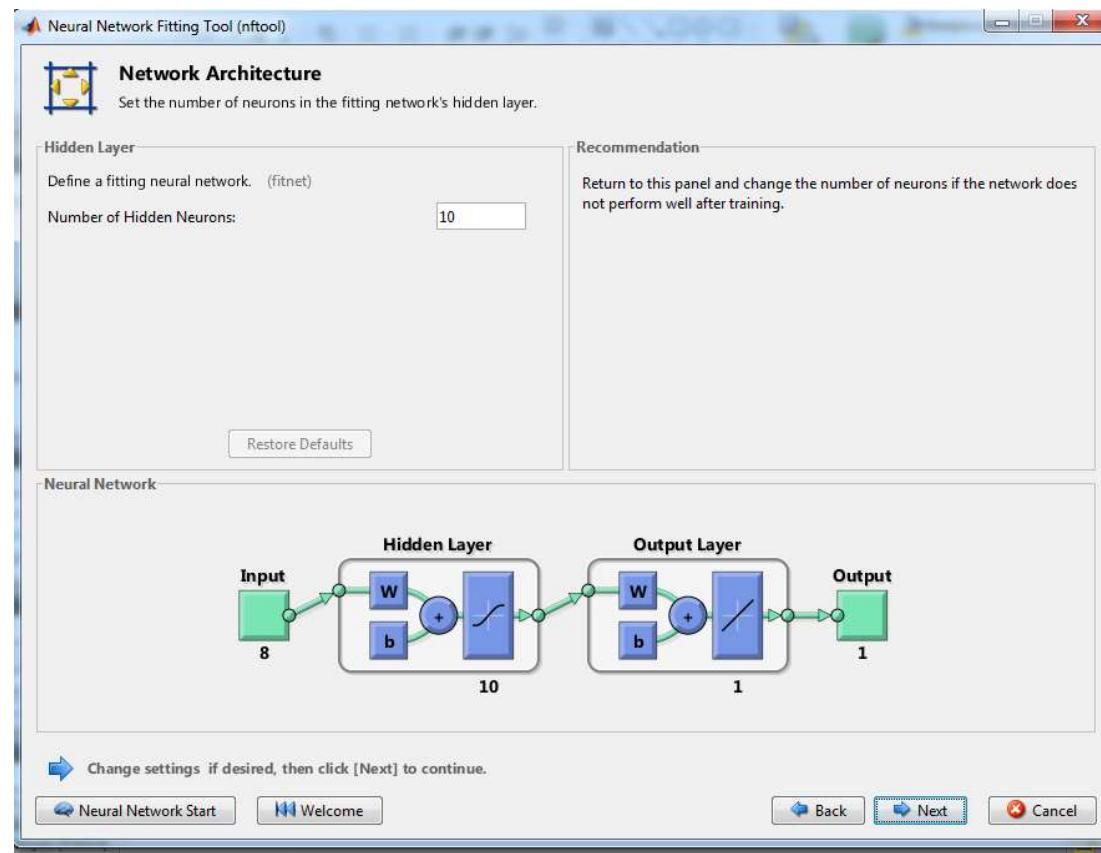
Exemple : capteurs chimiques

- Séparation de la base de données entre données pour l'apprentissage, la validation et le test



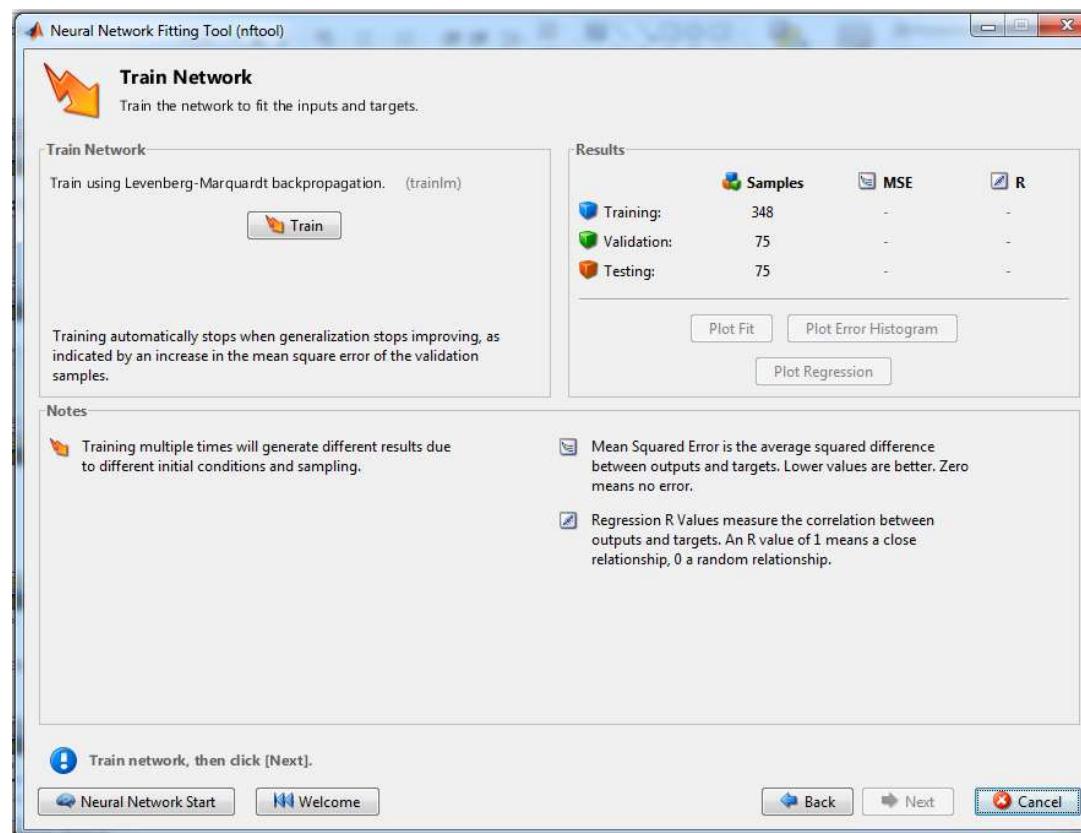
Exemple : capteurs chimiques

- Créer un réseau avec 10 neurones



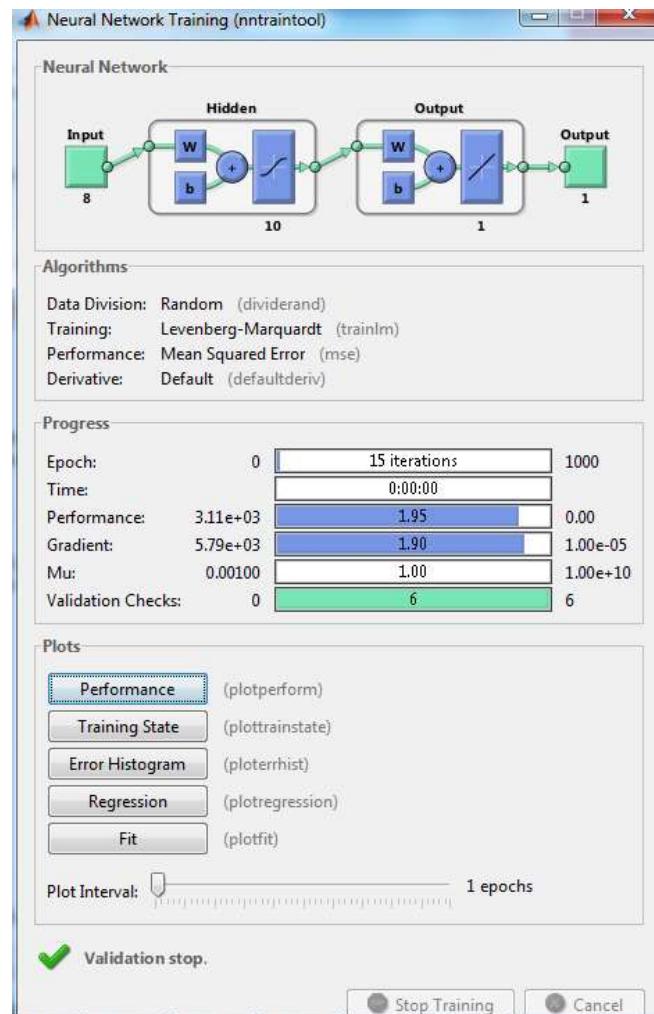
Exemple : capteurs chimiques

- Critères d'arrêt / résultats dépendent des valeurs initiales des poids

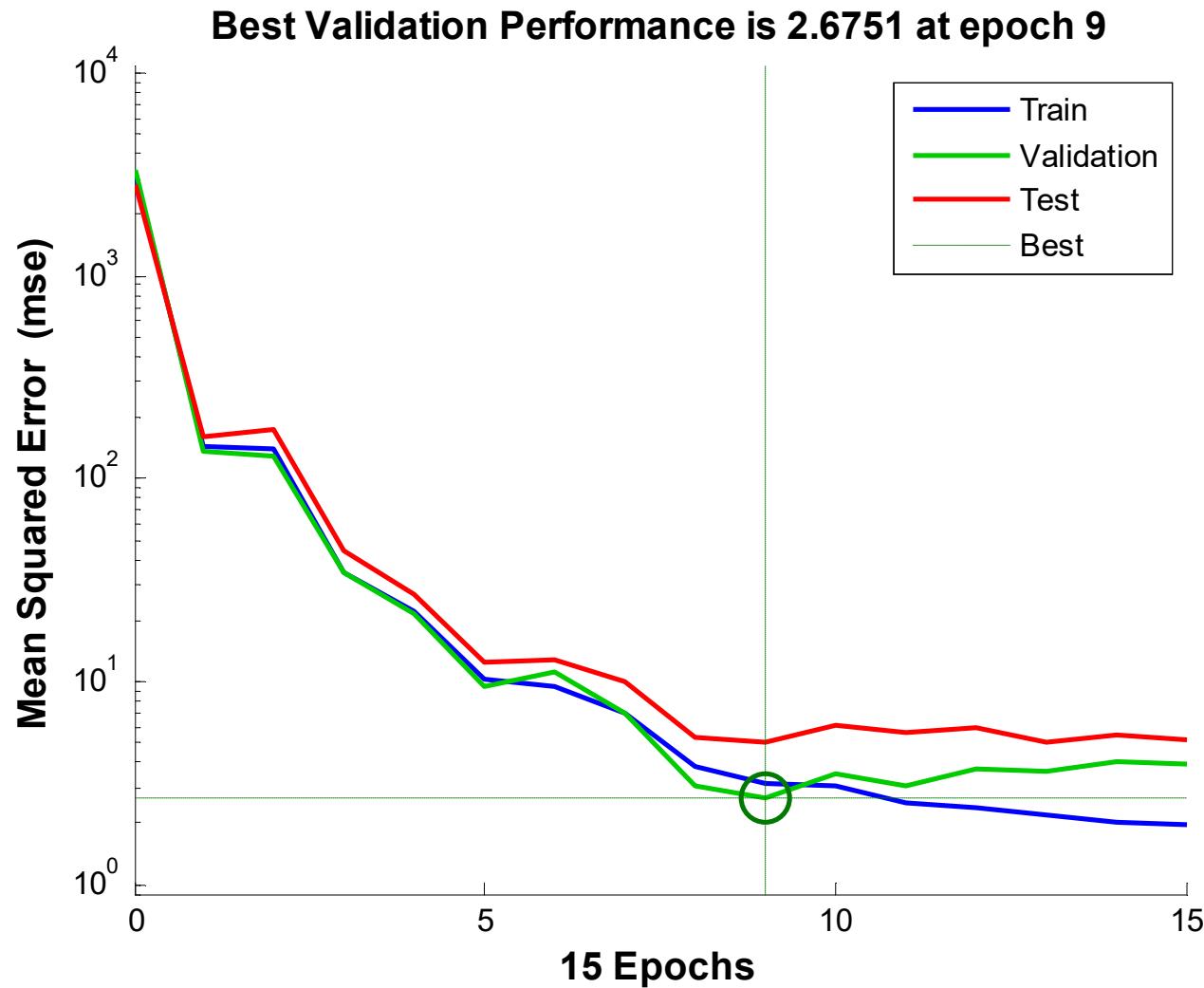


Exemple : capteurs chimiques

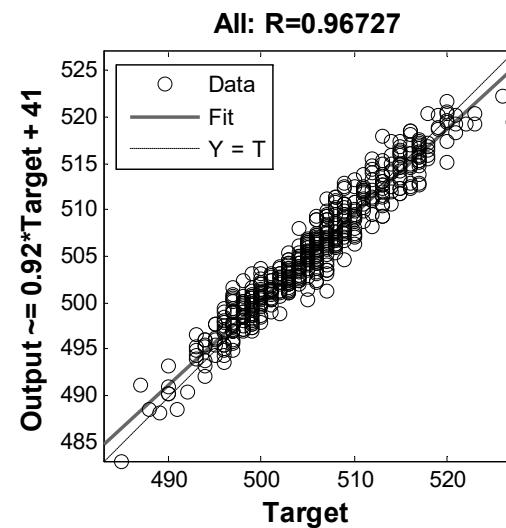
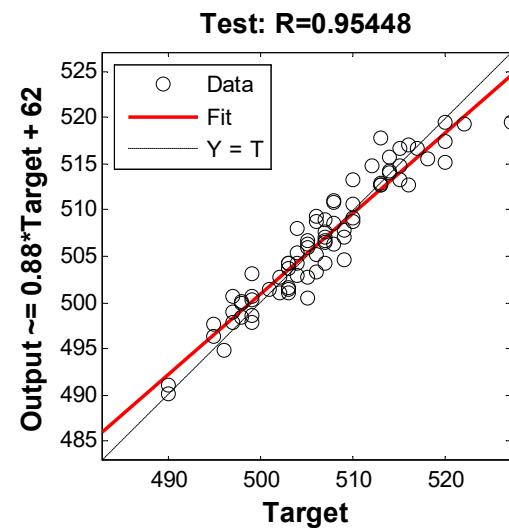
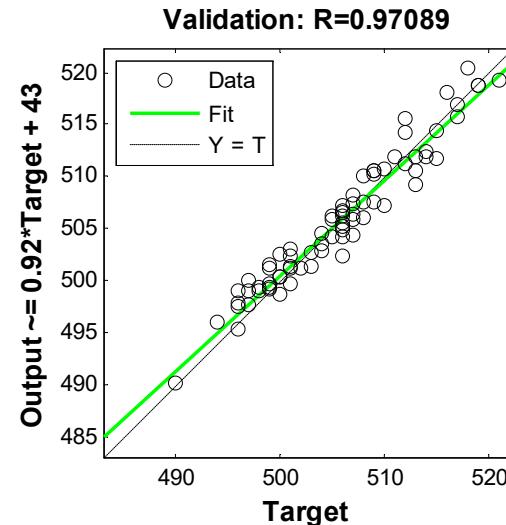
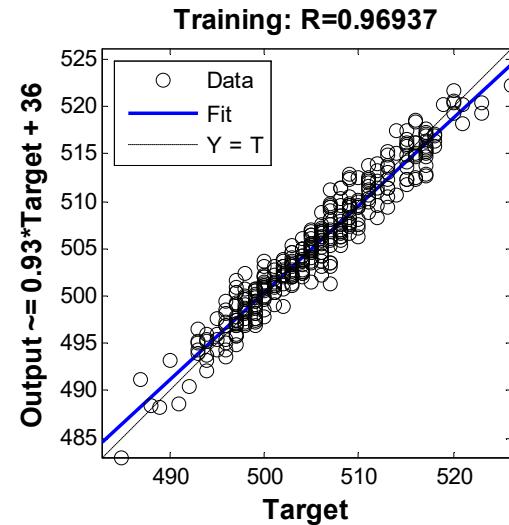
- Résultats



Exemple : capteurs chimiques

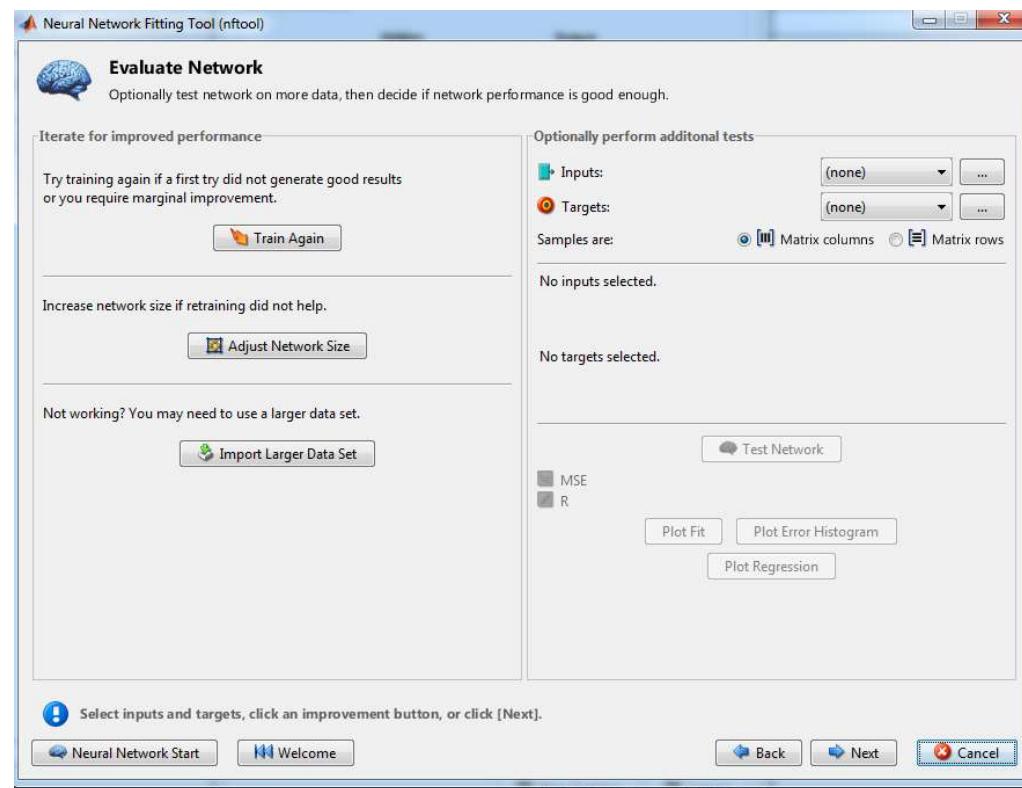


Exemple : capteurs chimiques



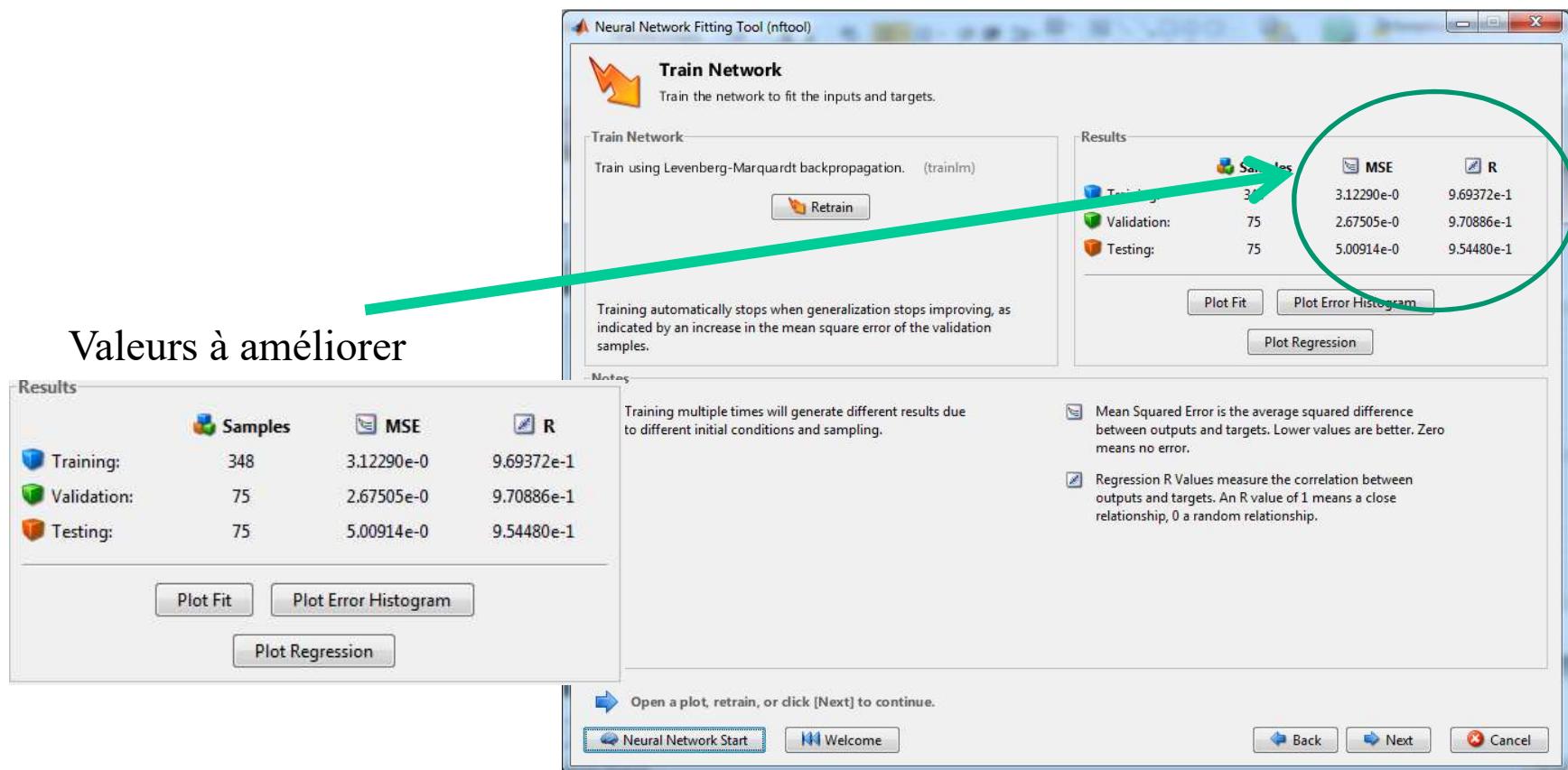
Exemple : capteurs chimiques

- Refaire un « training » avec d'autres valeurs initiales/ changer le nombre de neurones

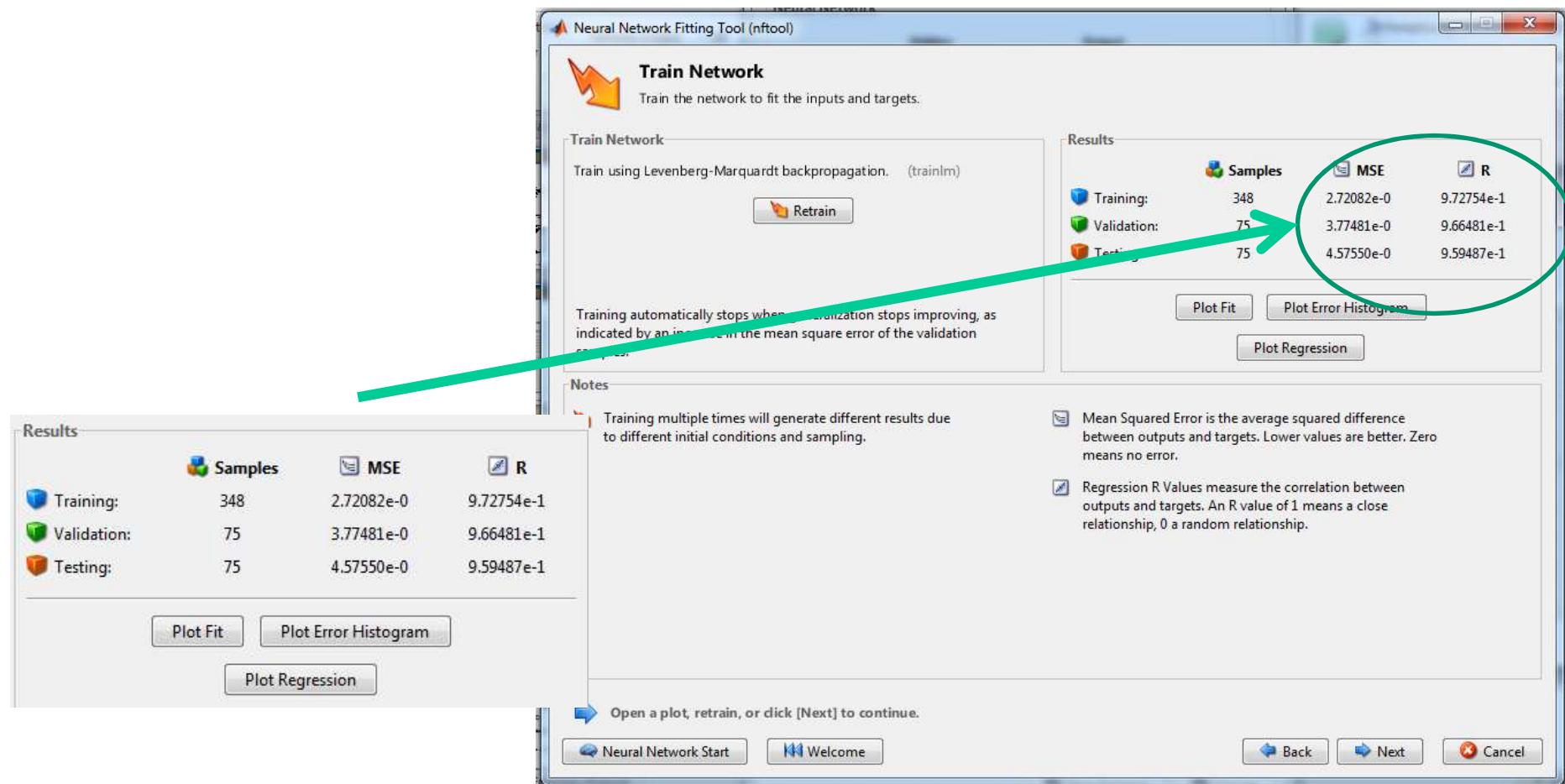


Exemple : capteurs chimiques

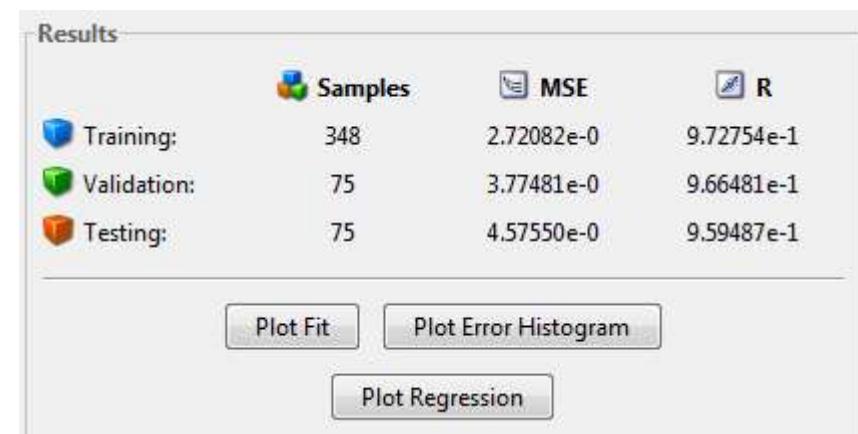
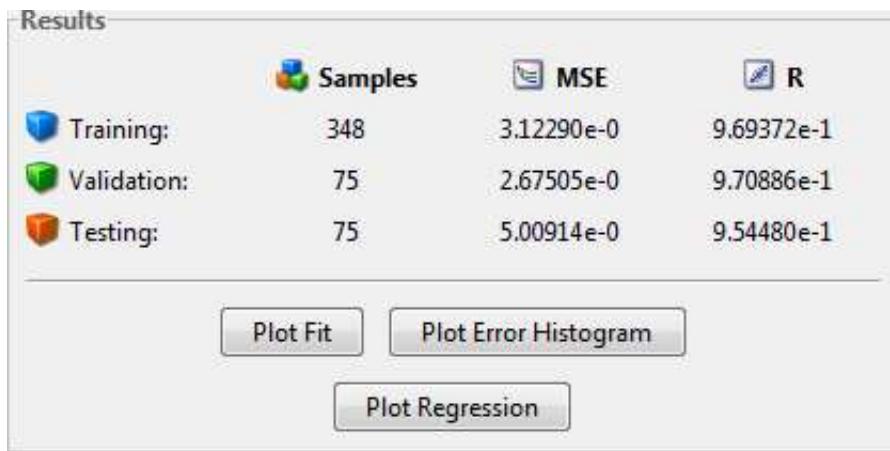
Valeurs à améliorer



Exemple : capteurs chimiques

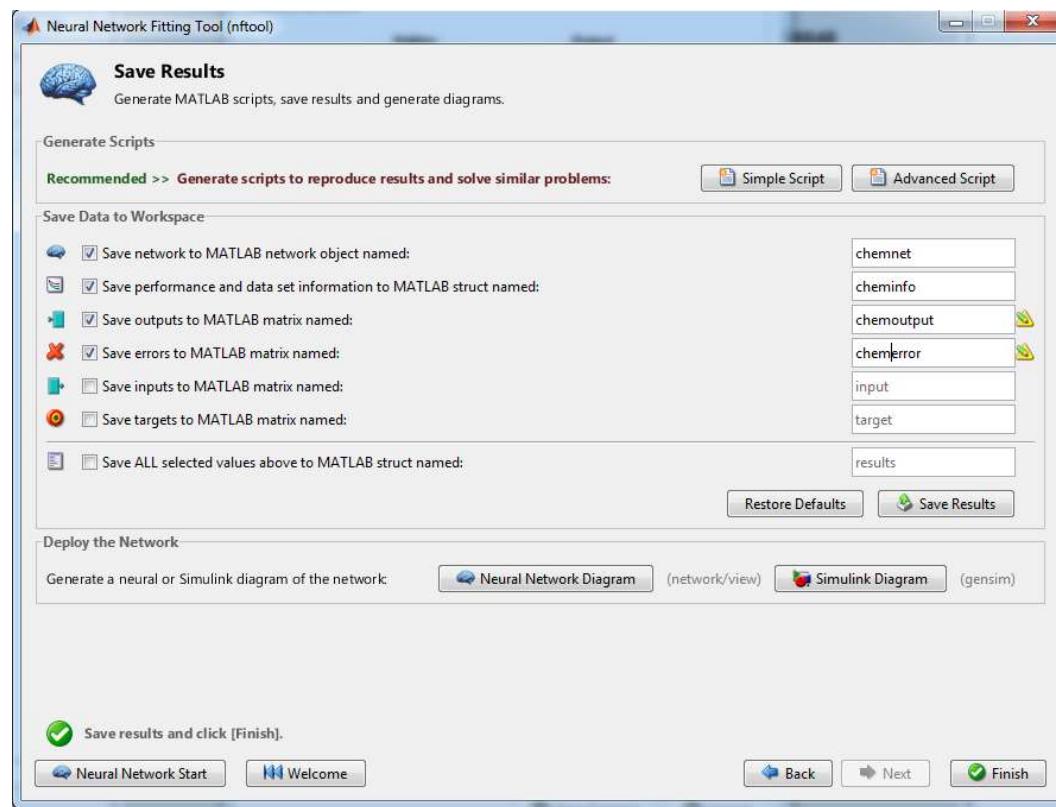


Exemple : capteurs chimiques



Exemple : capteurs chimiques

- Sauvegarde de la structure du réseau et de ses paramètres



Exemple : capteurs chimiques

- 20 neurones

Results			
	Samples	MSE	R
Training:	348	1.24510e-0	9.87950e-1
Validation:	75	6.02916e-0	9.45136e-1
Testing:	75	4.30668e-0	9.53534e-1

- 50 neurones

Results			
	Samples	MSE	R
Training:	348	1.13923e-0	9.89135e-1
Validation:	75	4.77284e-0	9.58617e-1
Testing:	75	4.44257e-0	9.47199e-1

- 100 neurones

- → excellent en training

mais très mauvais en

validation et en test

Results			
	Samples	MSE	R
Training:	348	1.89043e-22	1.00000e-0
Validation:	75	30.74229e-0	7.72809e-1
Testing:	75	50.73893e-0	7.58139e-1

- **Commande d'un four de LPCVD**

Capteur logiciel

Commande neuronale par modèle inverse

Implémentation/résultats expérimentaux

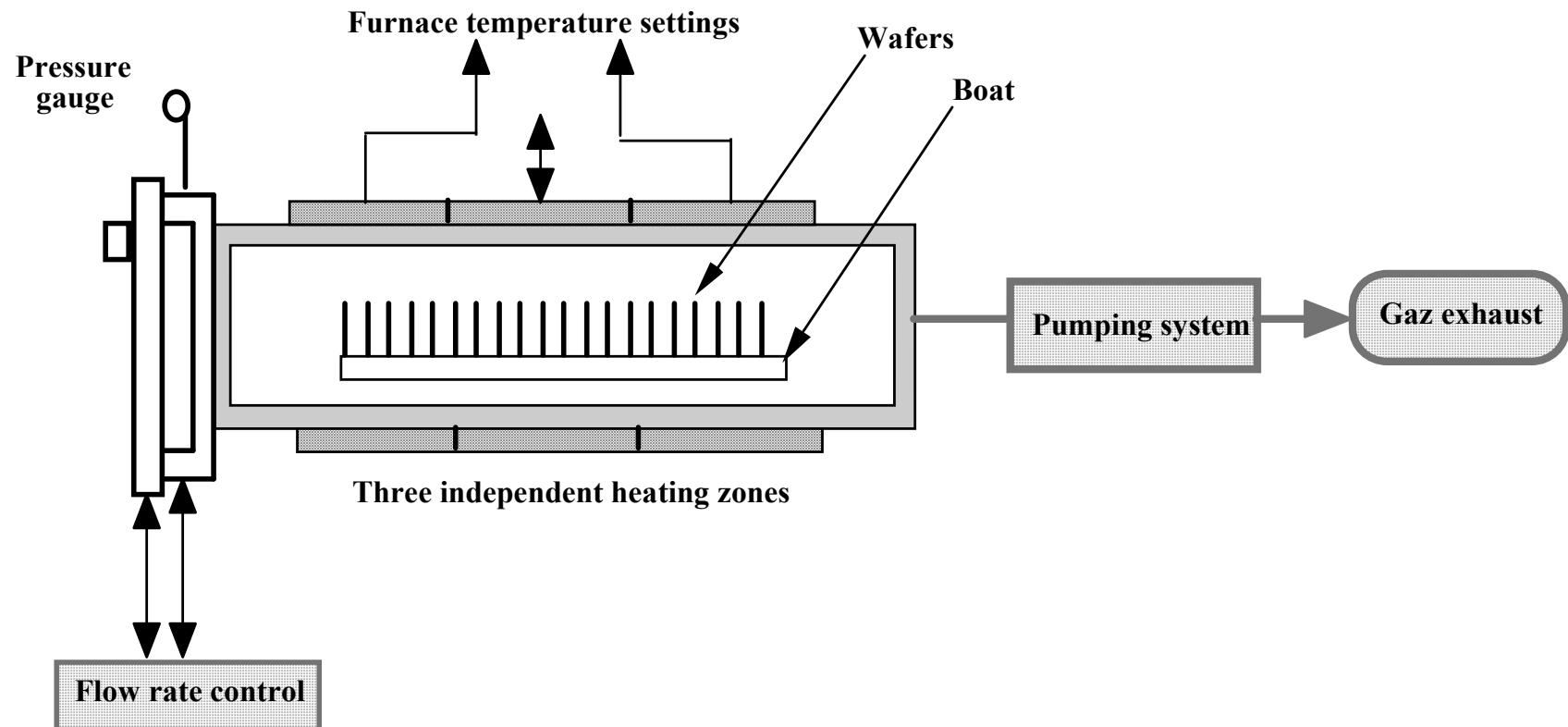
- **Détermination en ligne la dose de coagulant**

Sélection des informations/Modèle linéaire

Développement du réseau de neurones (capteur logiciel)

Comparaison des deux modèles

Commande d'un four de LPCVD



- Objectif : obtenir des plaquettes avec un dépôt uniforme
- Problèmes :

Comment mesurer l'épaisseur du dépôt ?

Il existait un modèle complexe impossible à utiliser en ligne

- *Développement d'un capteur logiciel*
- *Optimisation du profil de température le long du réacteur pour avoir une épaisseur uniforme*
- *Commande en temps réel:*
asservissement du profil de température dans le réacteur en agissant sur les 3 consignes des PID (consignes de température des enroulements)

- **Calcul de l'épaisseur du dépôt en temps réel**

Ne pas prendre un modèle global : découper en **zones de 10 plaquettes**

Introduire de la connaissance dans le capteur logiciel

Le débit de SiH₄ consommé dans la cellule (cinétique): $Q_{\text{SiH}_4} = F_{\text{Si}} (RT_0 / P_0)$

Le débit de SiH₄ entrant dans la cellule suivante : $D_{\text{SiH}_4}(n+1) = D_{\text{SiH}_4}(n) - Q_{\text{SiH}_4}(n)$

La réaction produisant 2 moles de H₂ pour une mole de SiH₄ consommé :

$$D_{\text{H}_2}(n+1) = D_{\text{H}_2}(n) + 2 * Q_{\text{SiH}_4}(n)$$

Utilisation du réseau de neurone pour modéliser uniquement les éléments difficilement modélisables par la physique (en ligne)

Calcul des vitesse de dépôt des plaquette 3 et 7 de la cellule de 10 :

La vitesse moyenne de dépôt approximée par : $V_{\text{dSi}} = (V_3 + V_7) / 2$

Le nombre de moles de Si déposées par seconde dans une cellule est :

$$F_{\text{Si}} = V_{\text{dSi}} * v_{\text{mSi}} * st$$

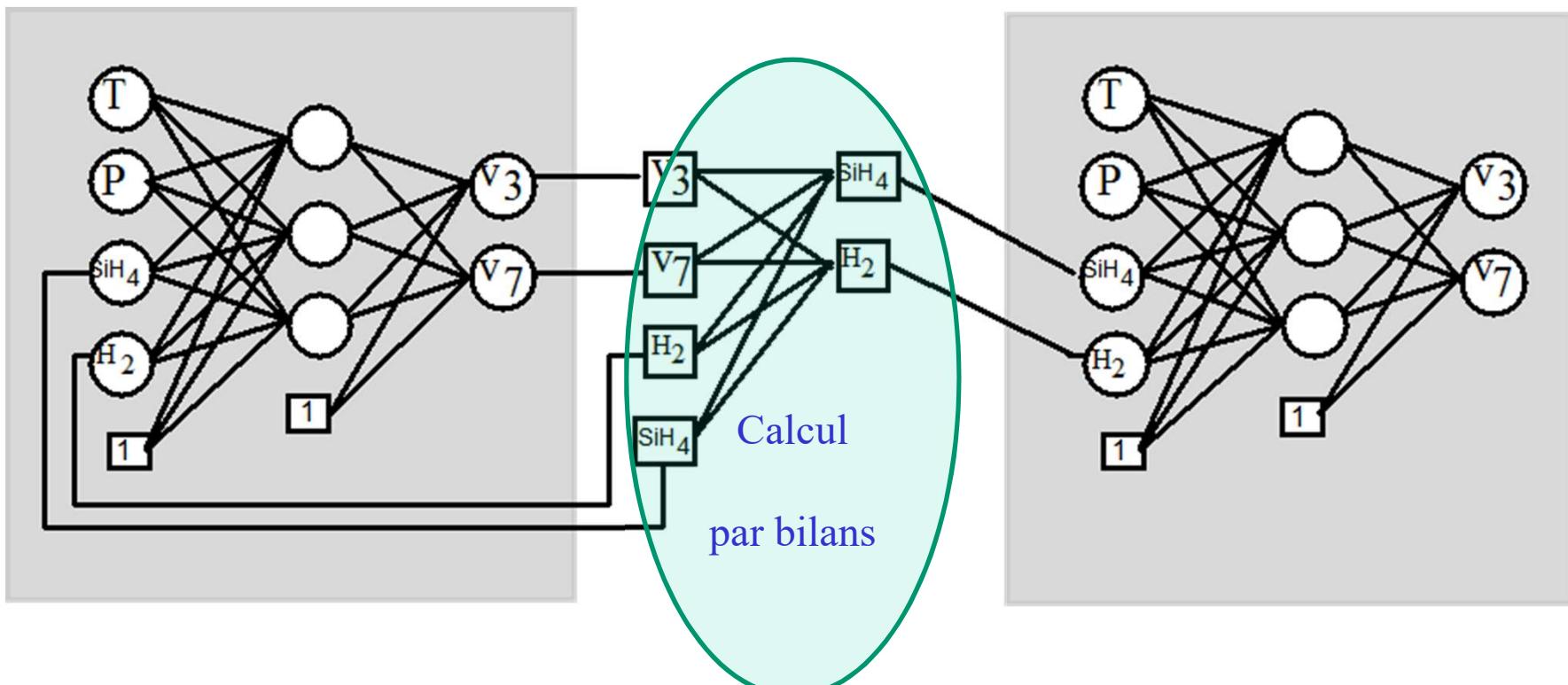




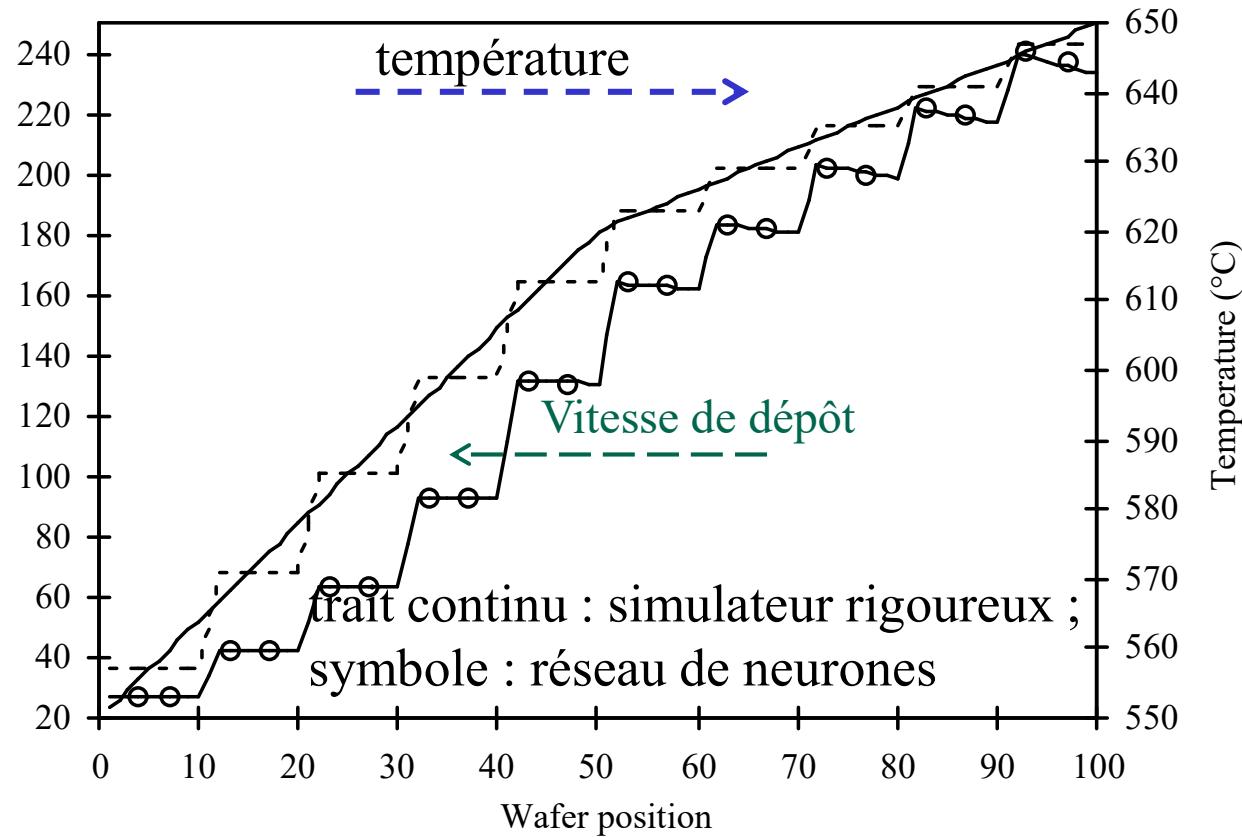
Développement d'un capteur logiciel combinant réseaux de neurones et modèle physique

Découpage du réacteur en tronçon de 10 plaquettes

Température=constante à l'intérieur du tronçon de 10 plaquettes



Commande d'un four de LPCVD



Comparaison de la vitesse de dépôt du silicium calculée par le réseau et le simulateur

Commande d'un four de LPCVD

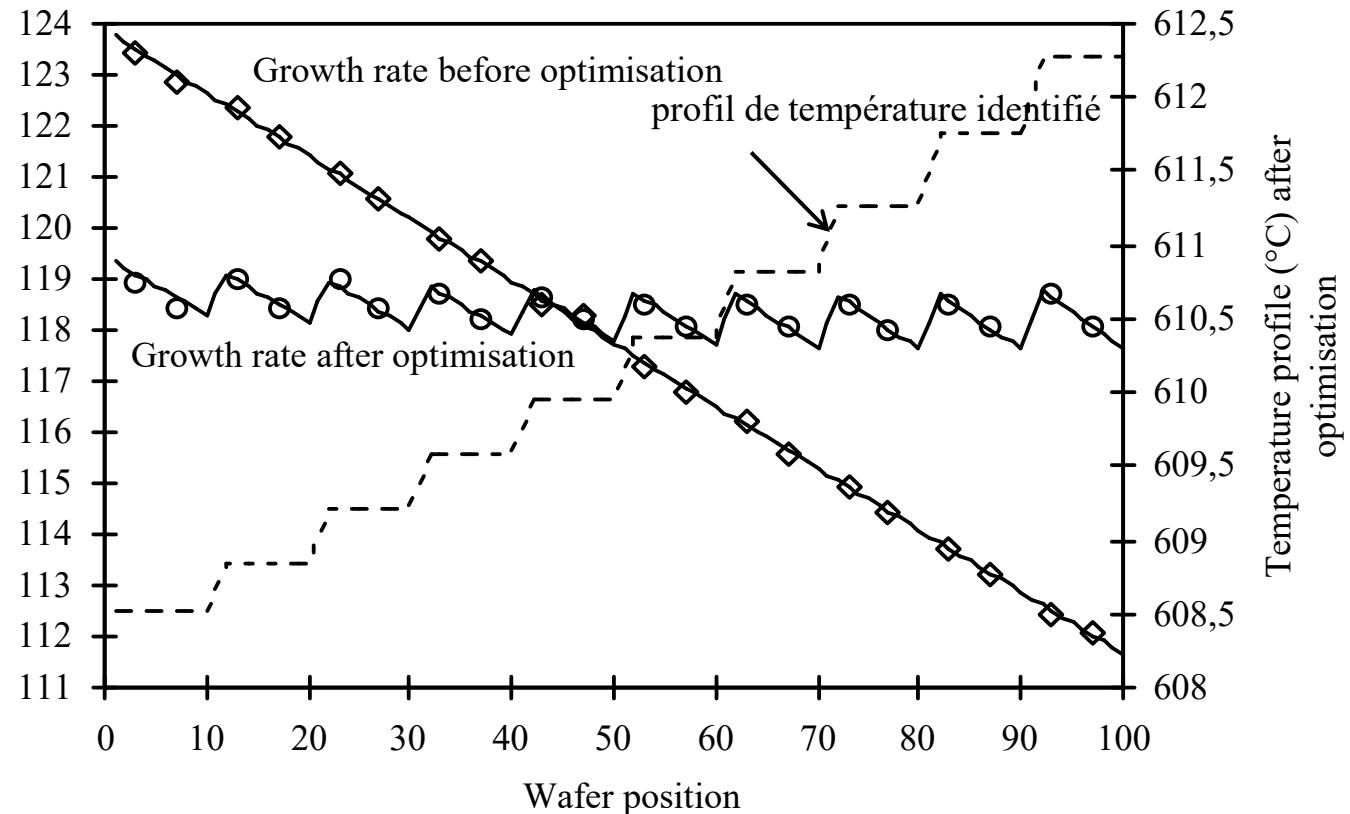
Optimisation en ligne du profil de température : $T_{ini} = 610^{\circ}\text{C}$

$$F_{obj} = \frac{1}{2} \sum_{j=2}^N \left[(V3Si_{ref} - V3Si(j))^2 + (V7Si_{ref} - V7Si(j))^2 \right]$$

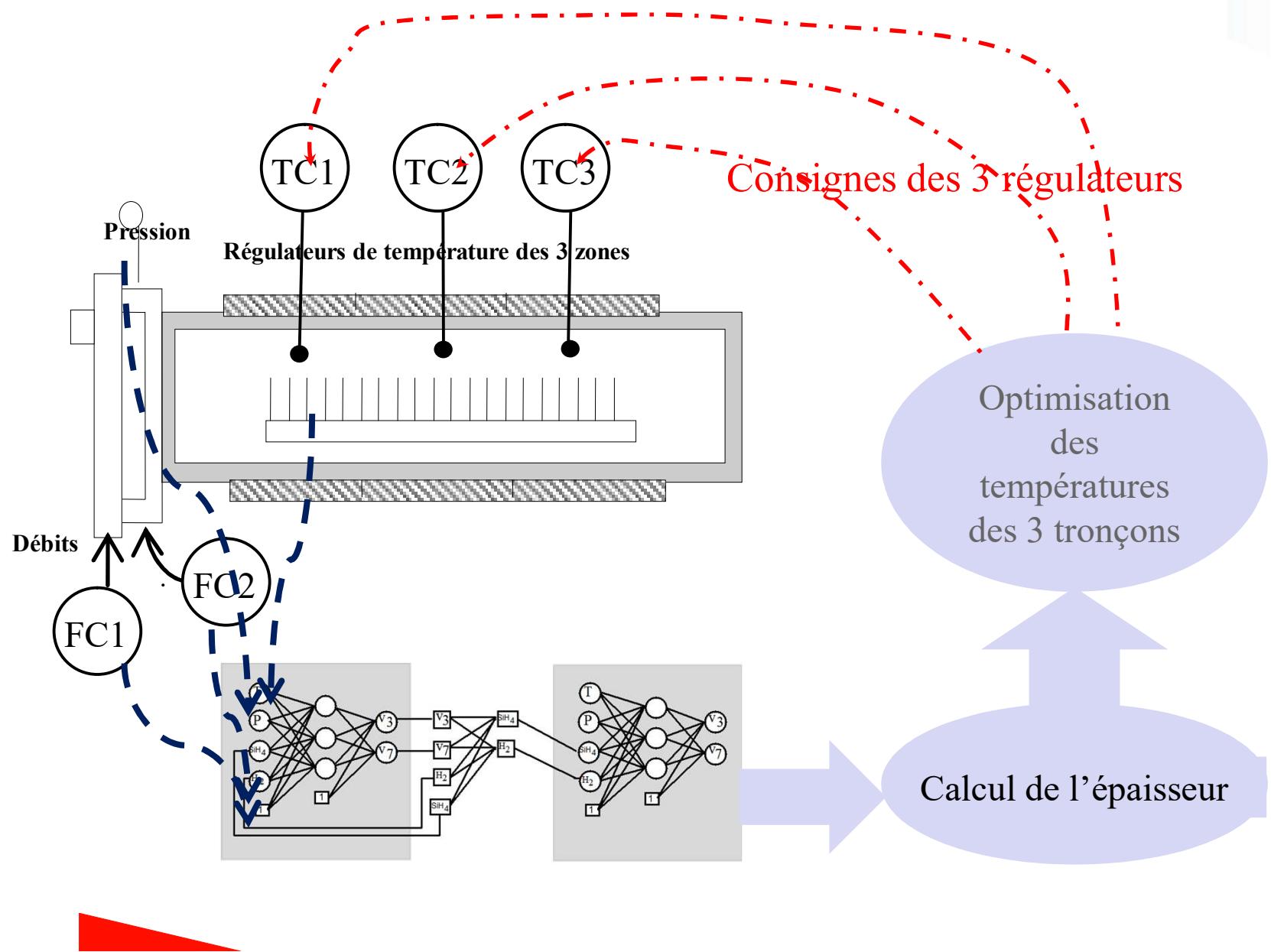
Recherche de la température constante dans chaque tronçon



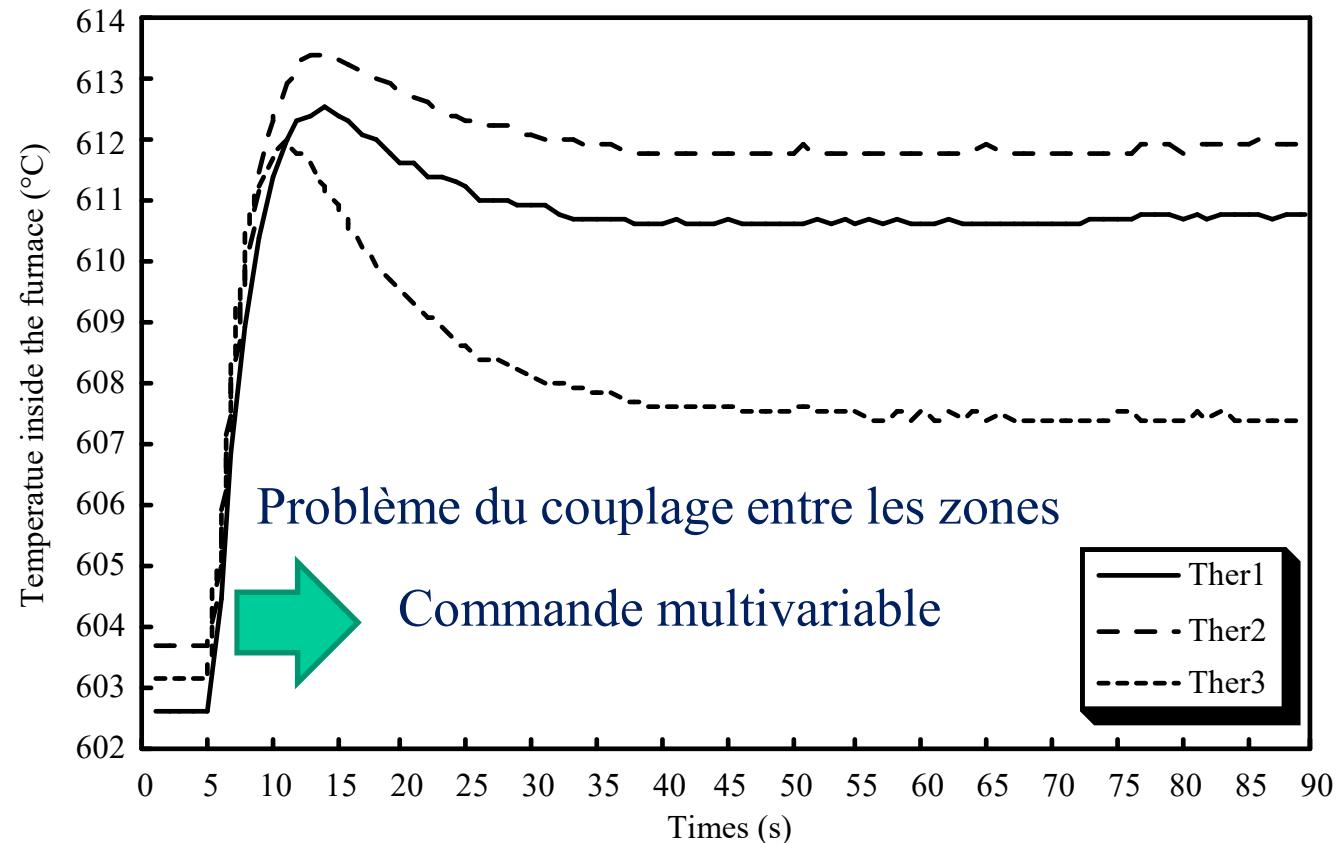
Optimisation monovariable sur chaque tronçon



Commande d'un four de LPCVD

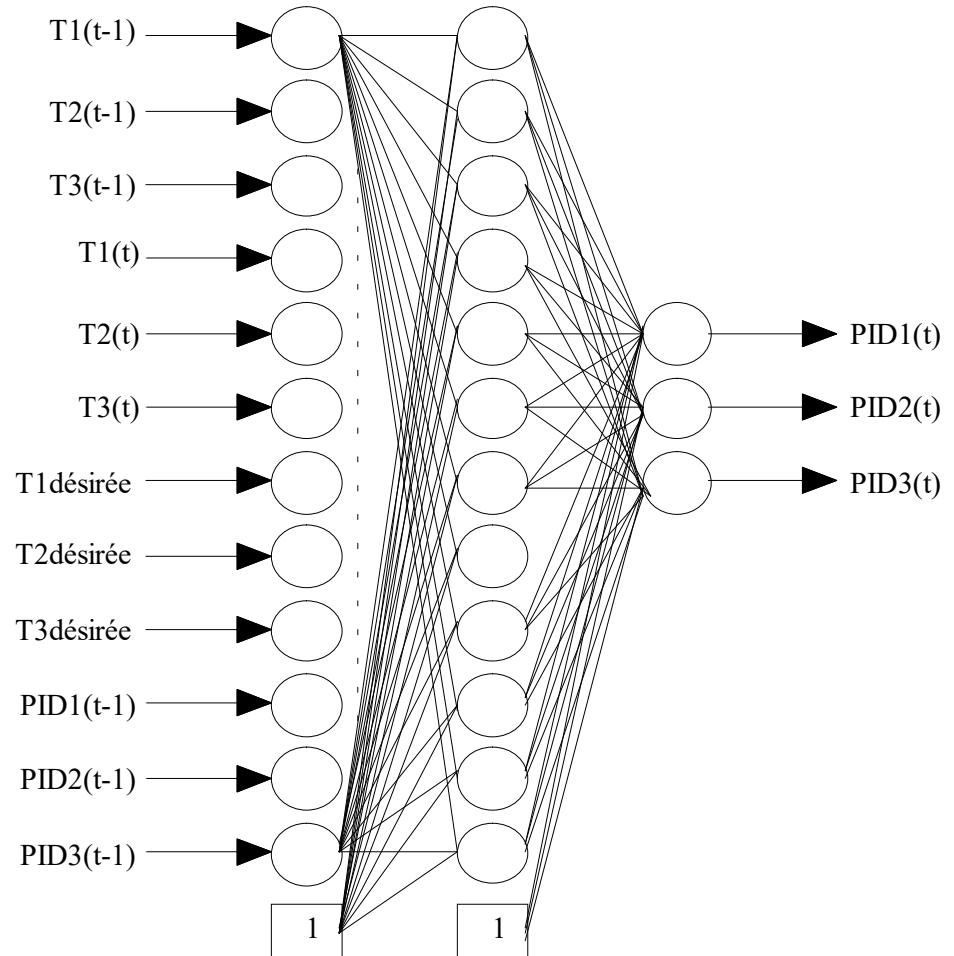


Commande d'un four de LPCVD



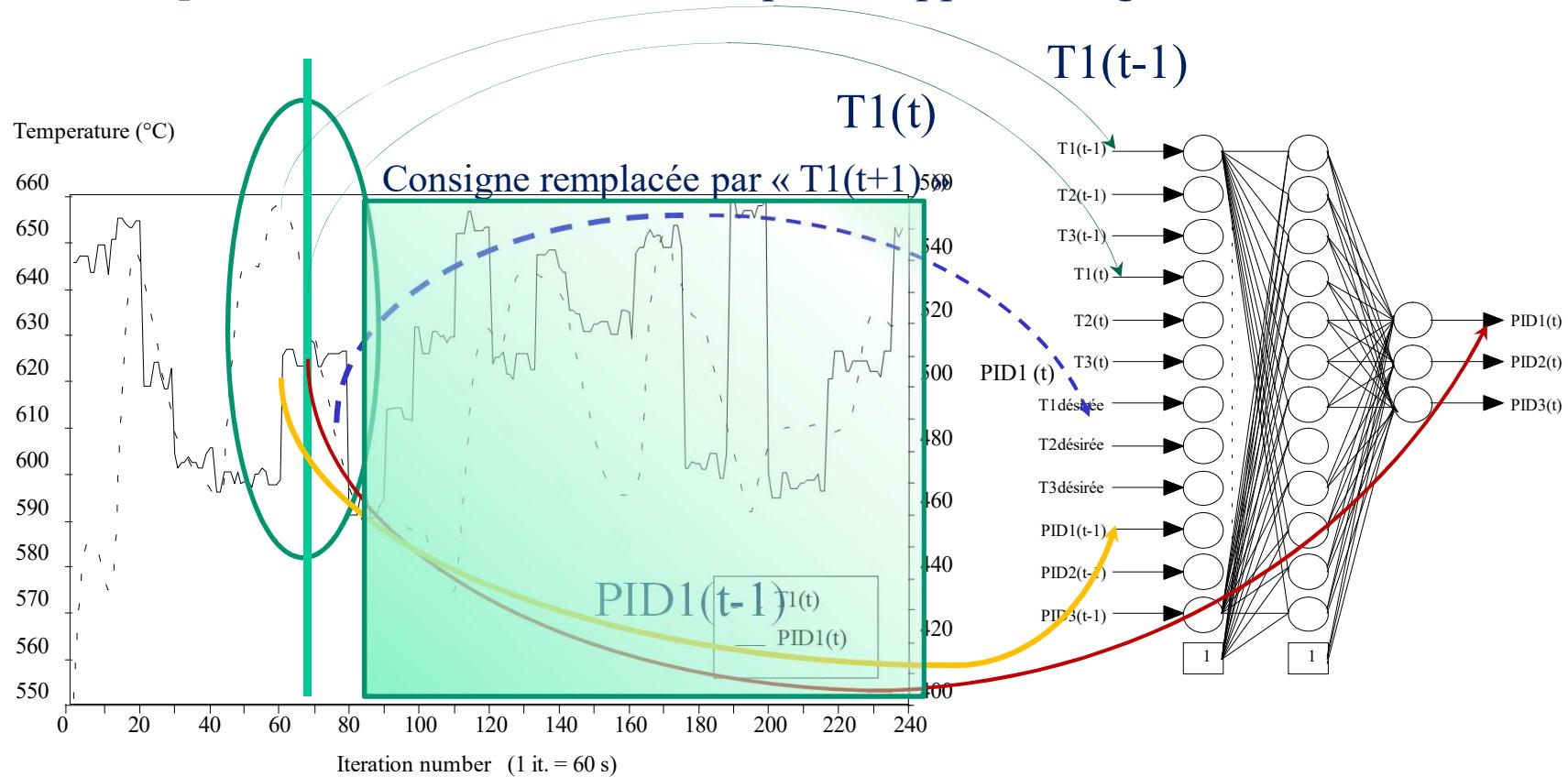
Réponse dynamique du four à une variation de 10°C appliquée sur la consigne du PID de la zone centrale (zone 2) échelon appliqué à t=5s.

Développement d'un réseau de neurones pour commande multivariable

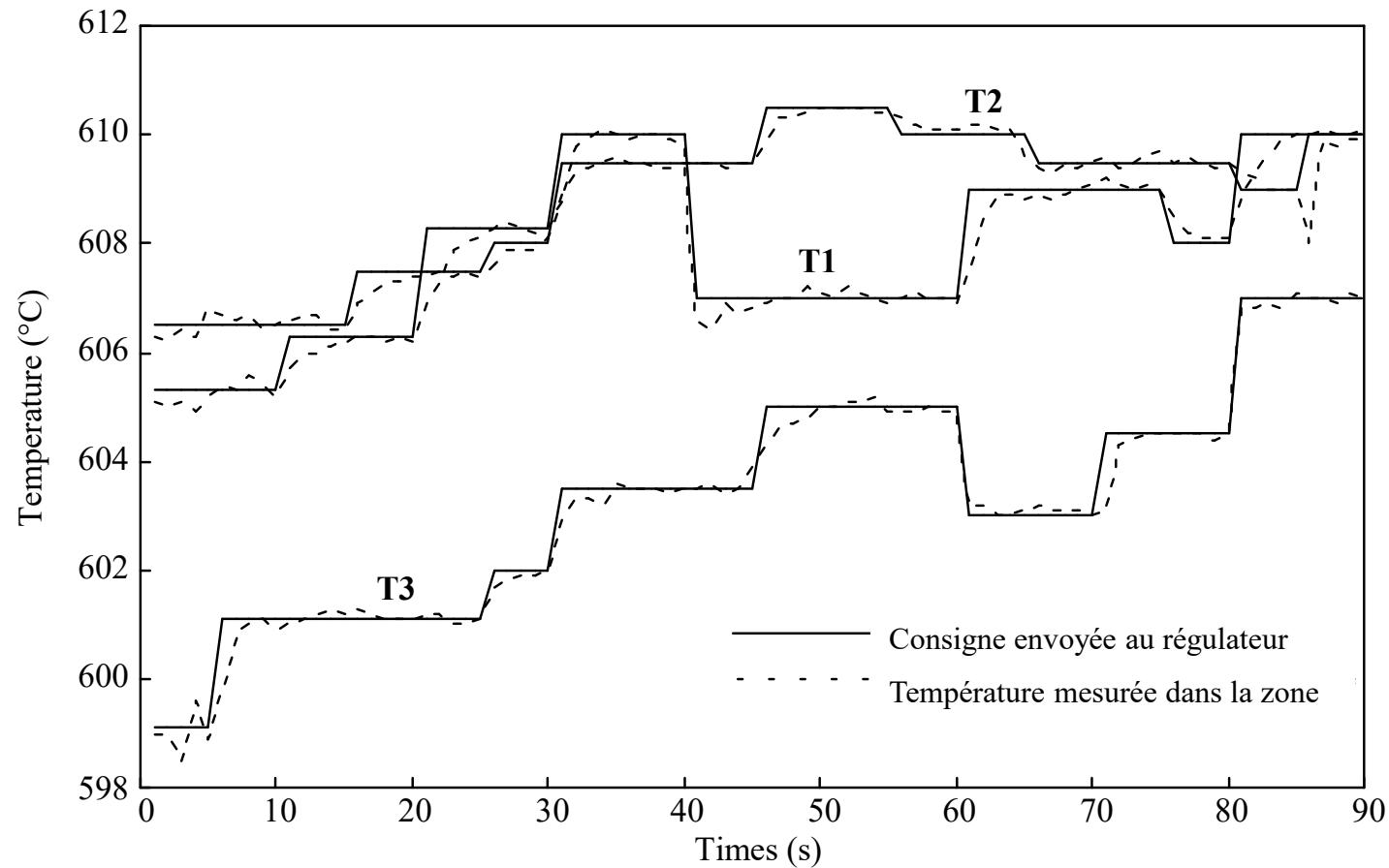


Structure du réseau de type modèle inverse

Expériences en boucle “ouverte” pour l'apprentissage



Commande d'un four de LPCVD

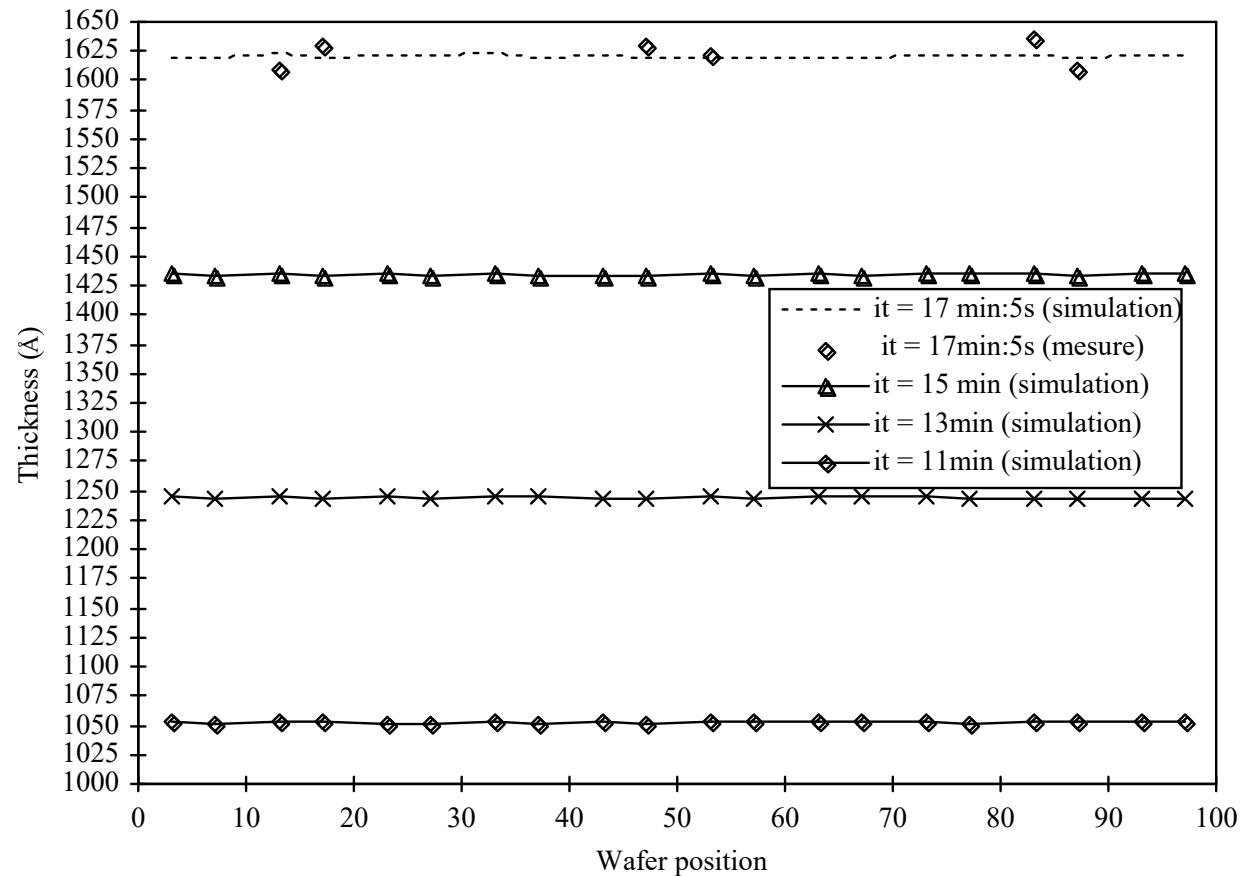


Commande d'un four de LPCVD

Epaisseur moyenne =

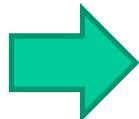
$$Nb \text{ de cellules} \frac{V_3(j) + V_7(j)}{\sum_{j=1}^{20}}$$

→ Arrêt pour l'épaisseur désirée = 1620 Å



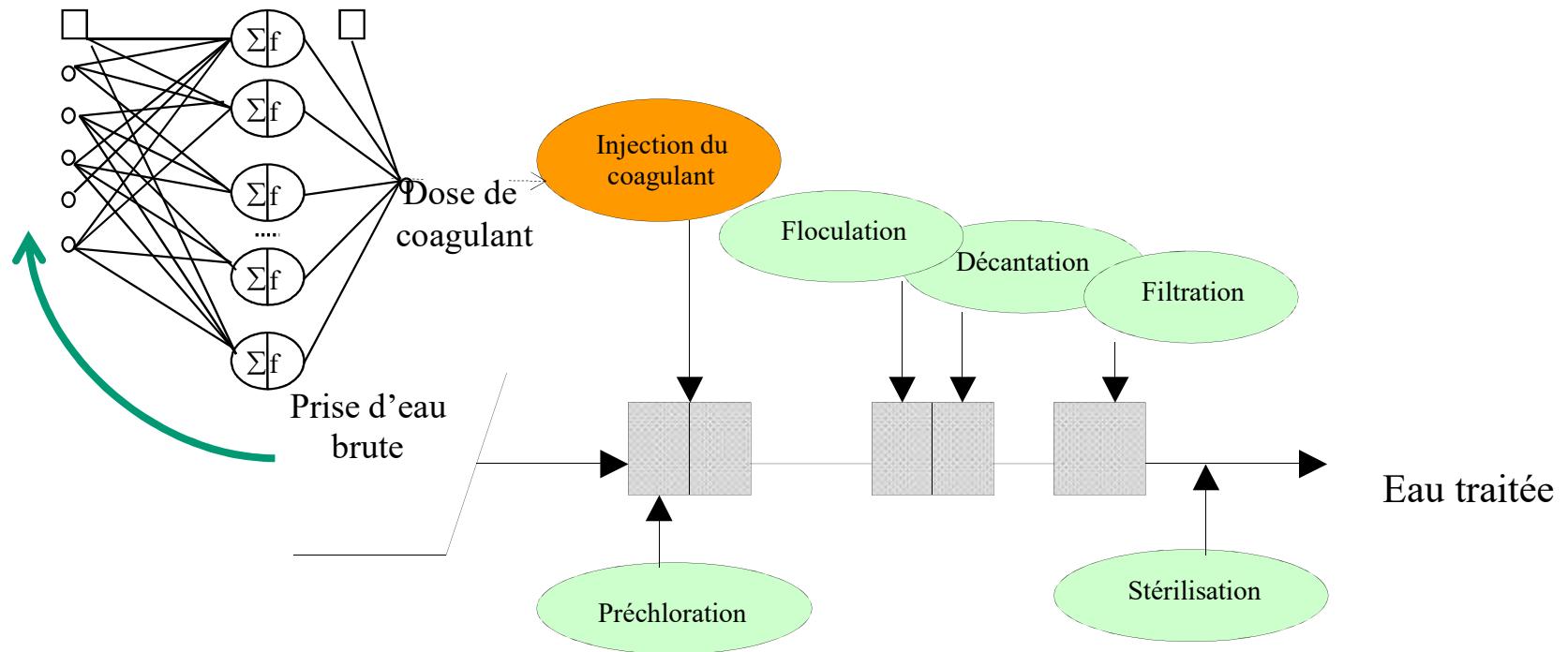
Détermination en ligne la dose de coagulant

- Station de production d'eau potable Rocade de Marrakech



Combien de coagulant ajouter en fonction de la qualité de l'eau brute ?

Actuellement, la réponse est donnée par des analyses en laboratoire effectuées chaque jour (jar test)



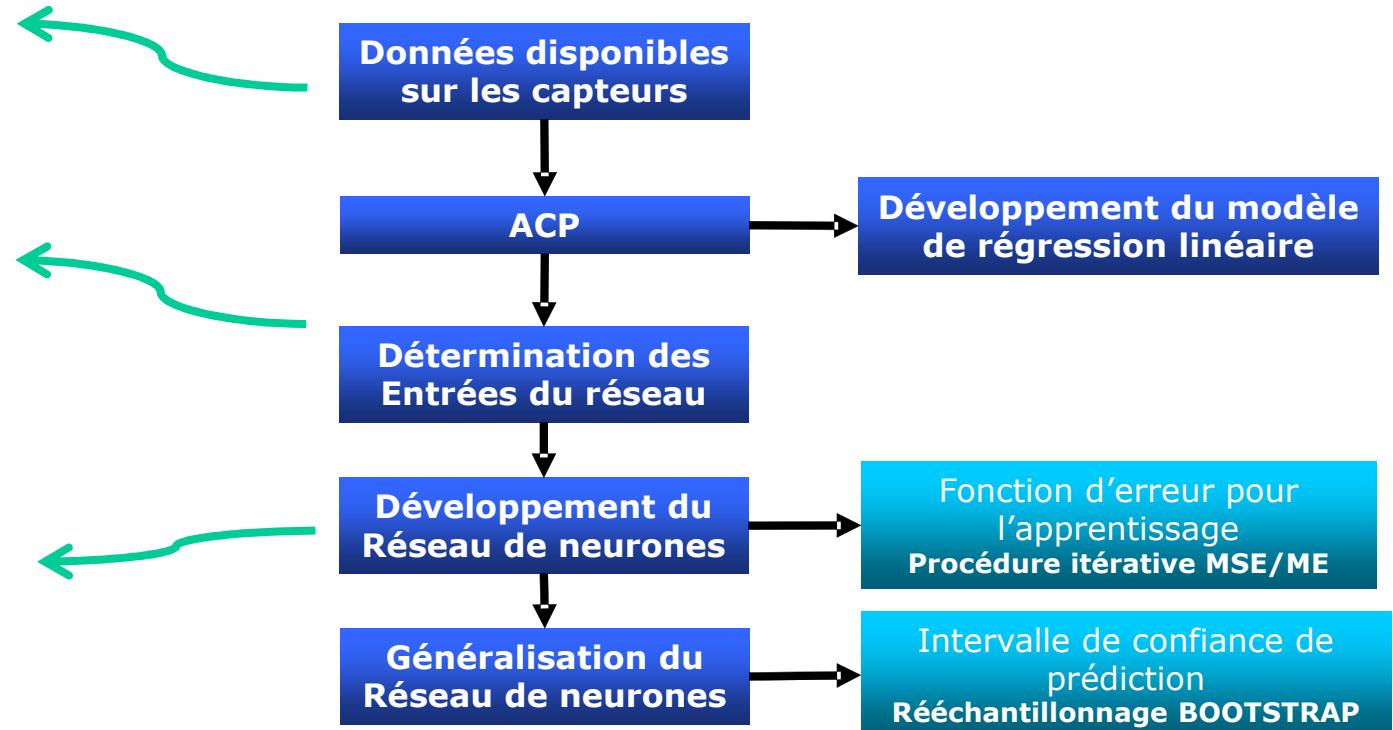
Détermination en ligne la dose de coagulant

→ Développement effectué par étapes successives

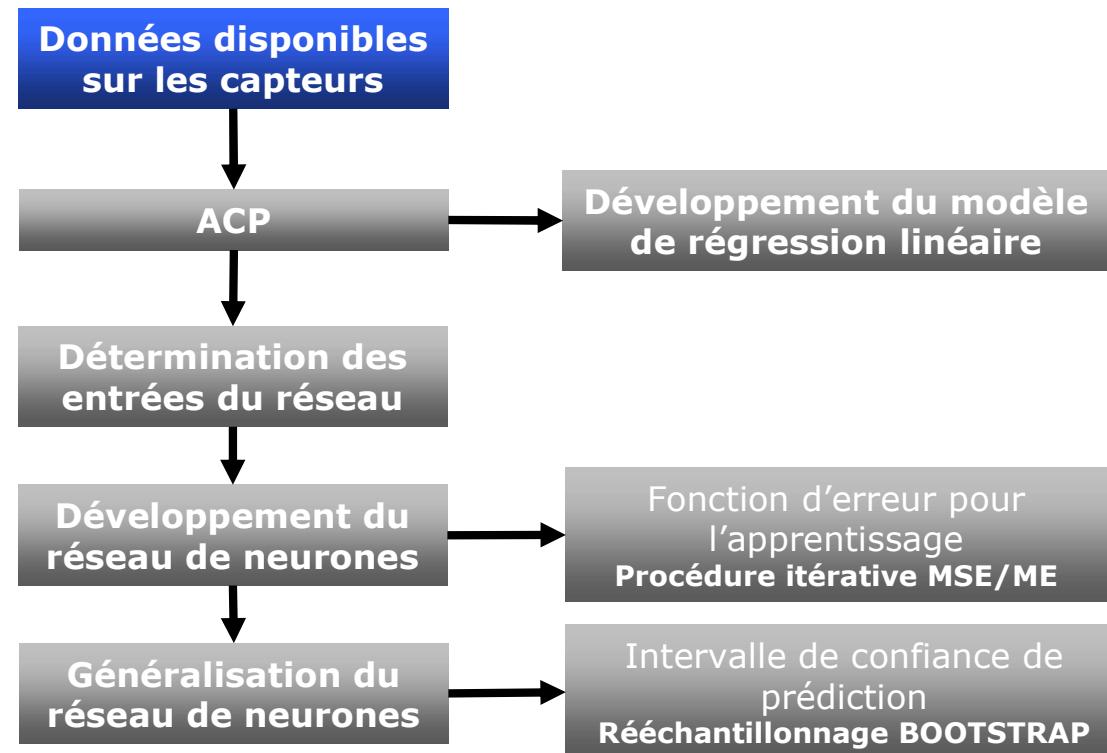
Analyse des données, période d'acquisition

Choix des données pertinentes, analyse des redondances ou corrélations

Détermination du nombre de neurones, des fonctions de base



Détermination en ligne la dose de coagulant



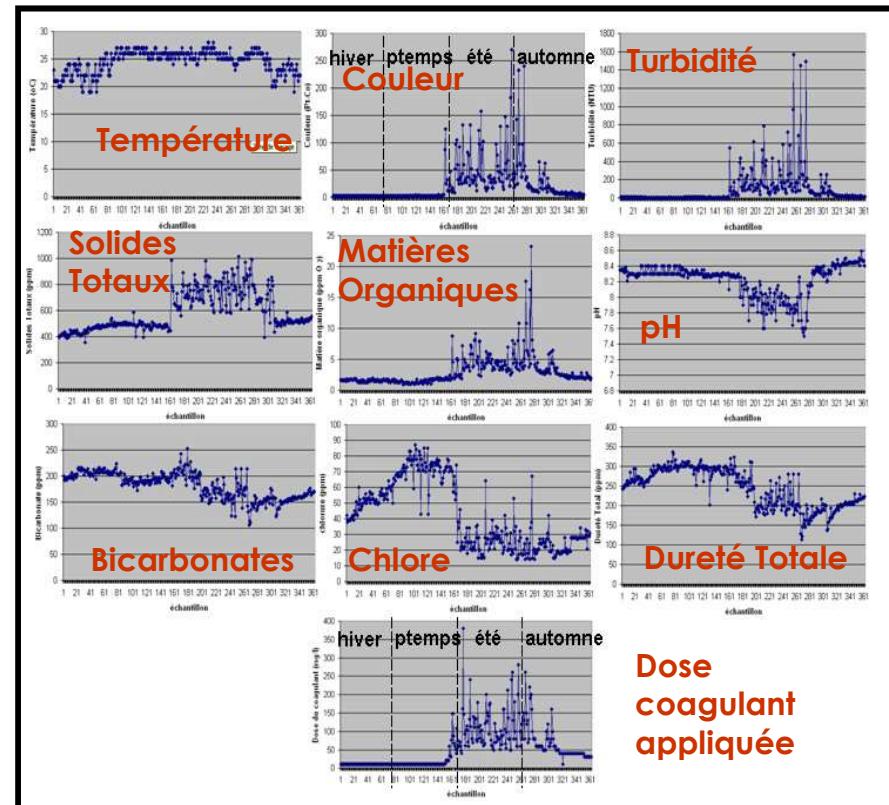
Détermination en ligne la dose de coagulant

- Données de la station Rocade :

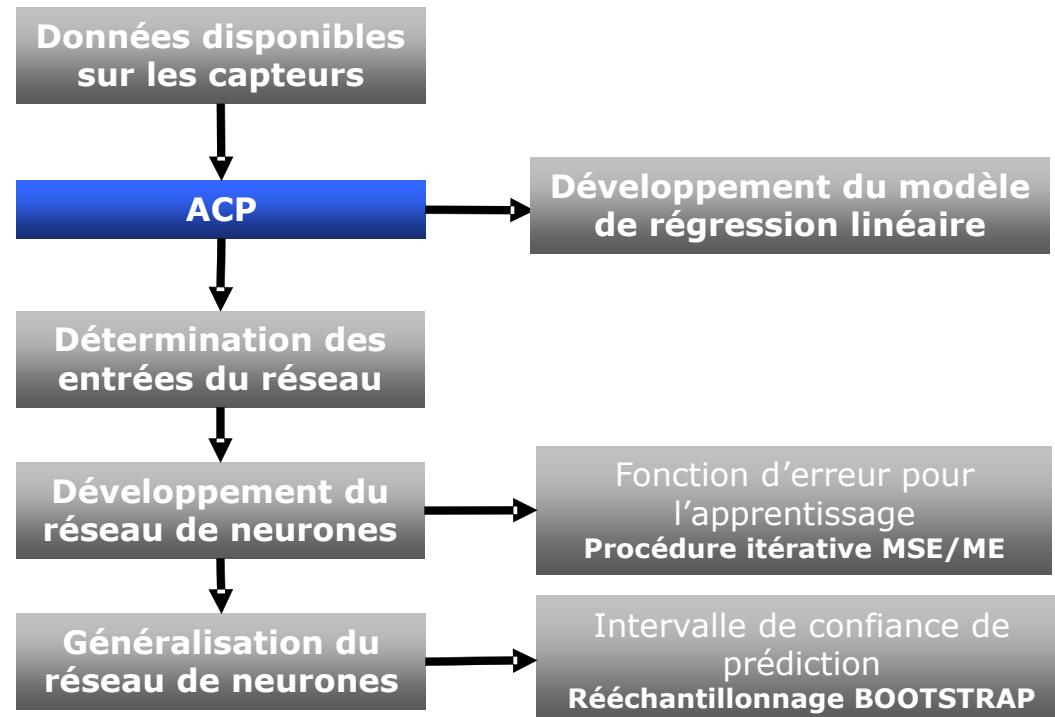
L'évolution de 9 différents paramètres de la qualité de l'eau mesurés au cours du temps

(2000-2004 = 5 ans)

La dose de coagulant appliquée sur la station au cours du temps



Détermination en ligne la dose de coagulant



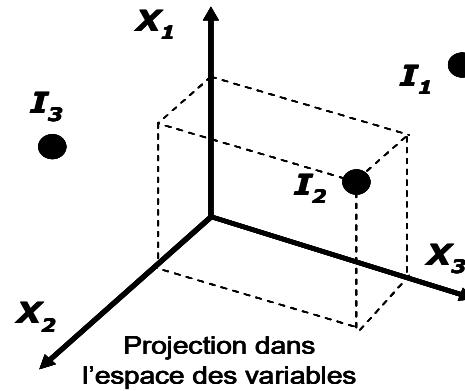
- **Analyse en composantes principales**
- *Objectifs*
 - Obtenir un modèle linéaire avec une dépendance également linéaire des variables explicatives
 - Extraire les corrélations entre variables
- **Forme générale** pour 1 variable de sortie et m variables d'entrée :

$$Y = w_0 + \sum_{j=1}^m w_j x_j + \varepsilon$$

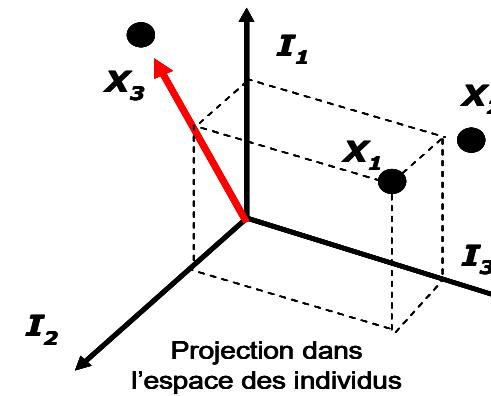


Analyse en composantes principales

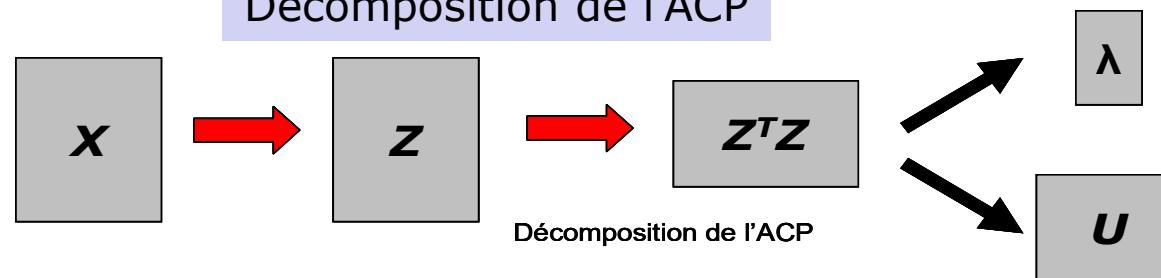
Projection dans
l'espace des variables



Projection dans
l'espace des composantes



Décomposition de l'ACP



$$CO = \frac{1}{N} \sum_{i=1}^N X_i X_i^T$$

$$COU = \lambda U$$

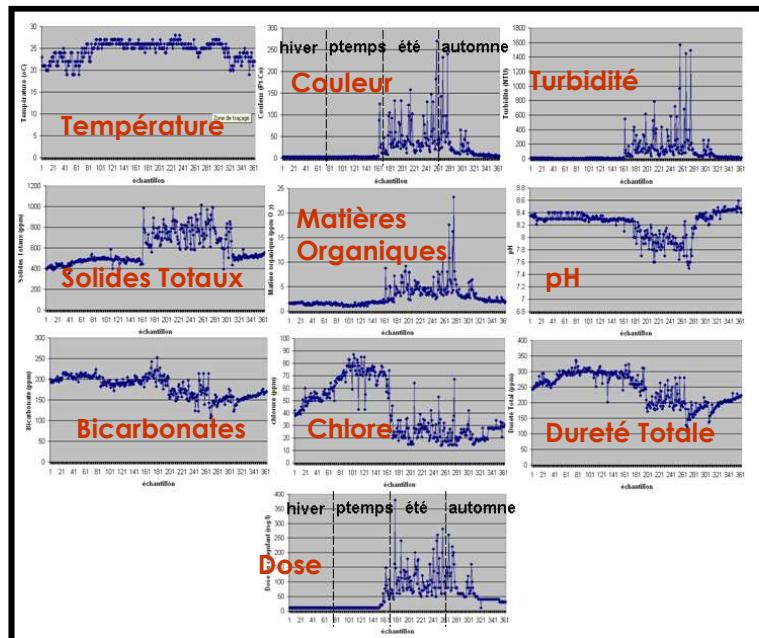
$X = (x_1, x_2, \dots, x_N)$ données

CO = Matrice de covariance

COU = Valeurs propres de CO



Détermination en ligne la dose de coagulant



Valeurs propres

ACP



Variable	Fact.1	Fact.2	Fact.3	Fact.4
Température (TEMP)	0,131	0,229	0,863	0,180
Couleur (C)	0,354	0,250	-0,232	0,236
Turbidité (TUR)	0,349	0,251	-0,270	0,306
Solides Totaux (ST)	0,393	0,102	0,097	-0,116
Matière organique (MO)	0,374	0,109	-0,159	0,062
pH	-0,216	-0,176	-0,207	0,616
Bicarbonate (B)	0,214	0,506	-0,191	-0,479
Chlorure (CL)	-0,267	0,455	0,084	0,436
Dureté Total (DT)	-0,274	0,540	-0,066	-0,034
DOSE	0,375	0,121	-0,057	-0,006

Vecteurs propres

#	Valeur	Pourcent	Cumul
1	5.446	54.455	54.455
2	1.812	18.116	72.571
3	1.010	10.099	82.670
4	0.535	5.354	88.024
5	0.420	4.197	92.221
6	0.250	2.500	94.721
7	0.218	2.180	96.901
8	0.154	1.541	98.442
9	0.090	0.900	99.342
10	0.066	0.658	100.000

Détermination en ligne la dose de coagulant

Matrice de corrélation

	Temp	Couleur	Turb	SolidesT	MatiereO	pH	Bicarb	Chlor	DureteT	Dose
Temp	1	0.194	0.161	0.383	0.185	-0.363	-0.120	0.048	-0.036	0.267
Couleur	0.194	1	0.856	0.788	0.734	-0.538	-0.215	-0.314	-0.281	0.760
Turb	0.161	0.856	1	0.693	0.810	-0.566	-0.240	-0.263	-0.281	0.725
SolidesT	0.383	0.788	0.693	1	0.755	-0.705	-0.393	-0.514	-0.463	0.836
MatiereO	0.185	0.734	0.810	0.755	1	-0.642	-0.361	-0.423	-0.453	0.734
pH	-0.363	-0.538	-0.566	-0.705	-0.642	1	0.207	0.329	0.327	-0.598
Bicarb	-0.120	-0.215	-0.240	-0.393	-0.361	0.207	1	0.560	0.843	-0.344
Chlor	0.048	-0.314	-0.263	-0.514	-0.423	0.329	0.560	1	0.800	-0.469
DureteT	-0.036	-0.281	-0.281	-0.463	-0.453	0.327	0.843	0.800	1	-0.404
Dose	0.267	0.760	0.725	0.836	0.734	-0.598	-0.344	-0.469	-0.404	1

Détermination en ligne la dose de coagulant

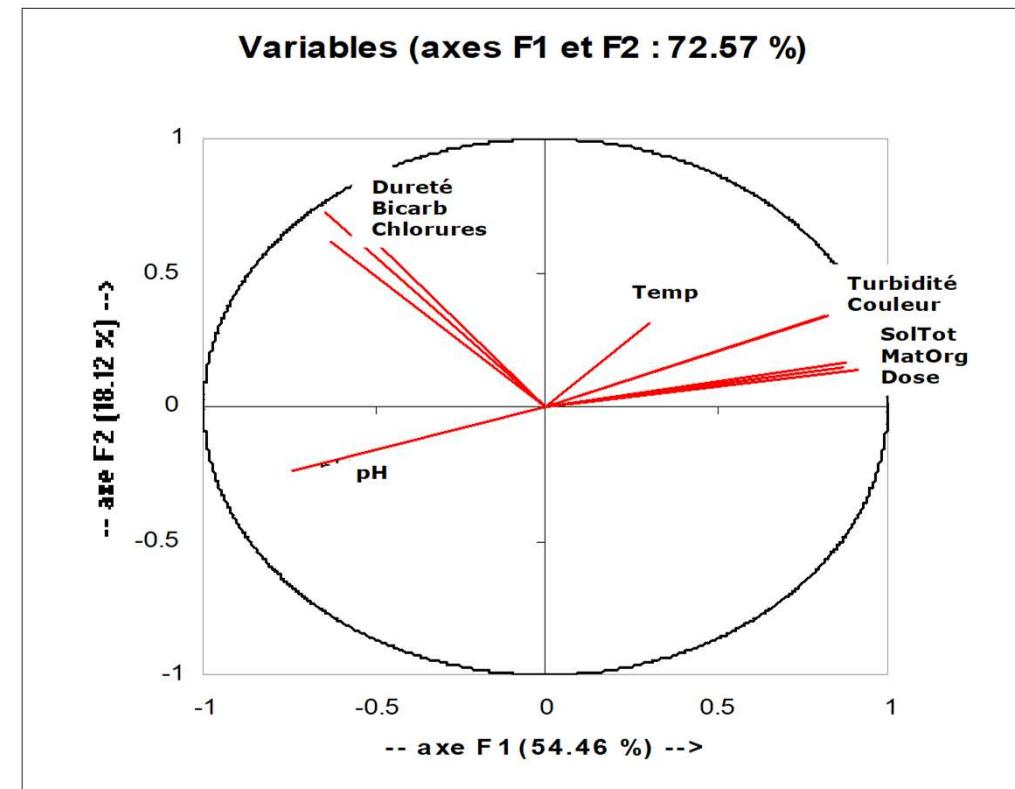
Cercle de corrélation



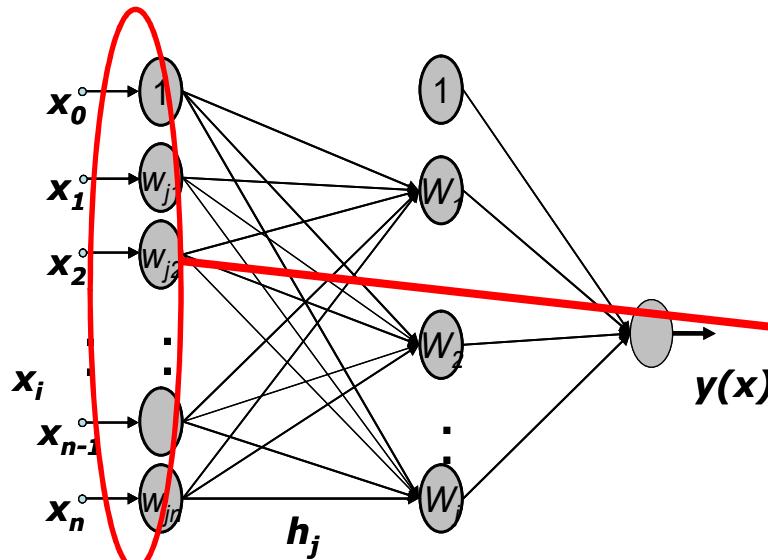
Variables:

Turb = f (couleur, solides Totaux, matières Organiques);

Dureté = f (bicarbonate, chlorure); **Temp et pH**



Détermination en ligne la dose de coagulant

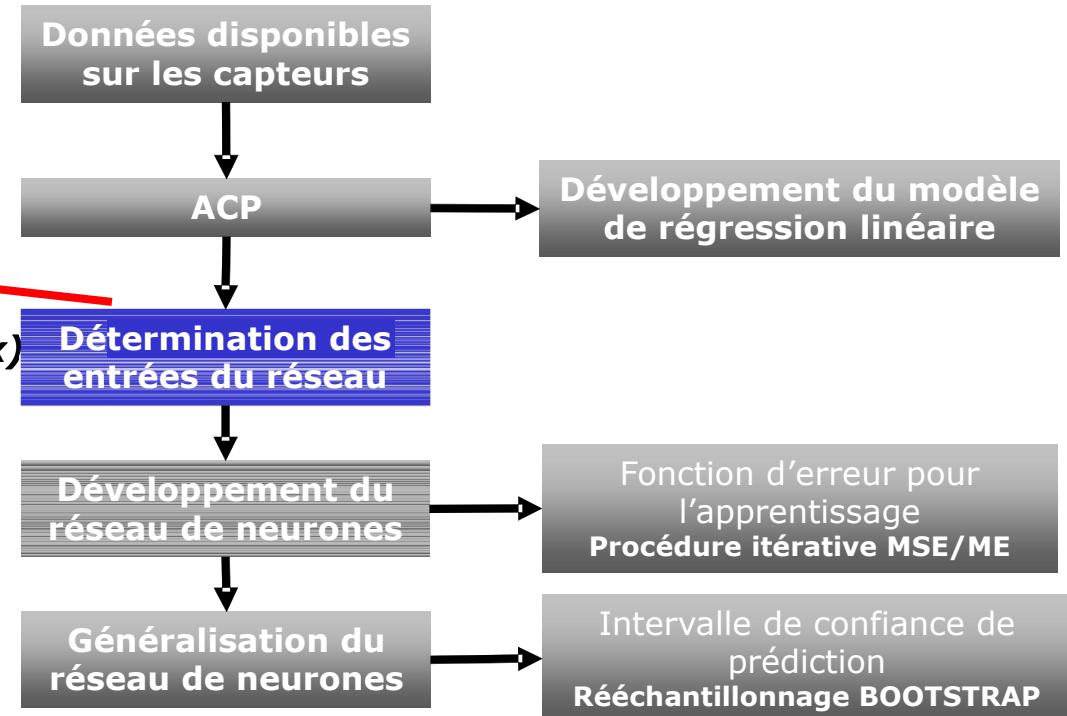


$$h_j = \sum_{i=1}^n w_{ji} x_i + w_{j0} \quad \text{et} \quad y(x) = f\left(\sum_{j=1}^H w_j h_j + w_0\right)$$

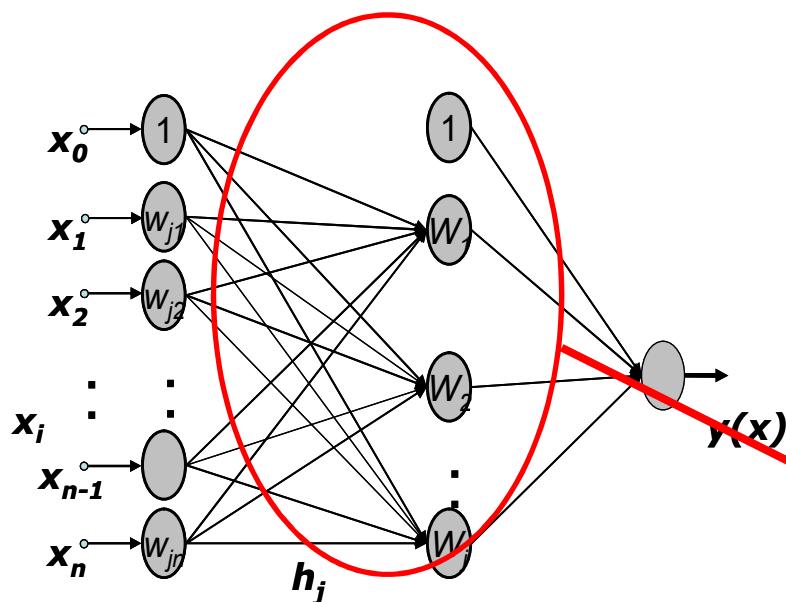
$y(x)$ = Sortie du perceptron multicouche
 f = Fonction d'activation (sigmoïde)

5 entrées

Turbidité
 Température
 pH
 Dureté
 Oxygène Dissous

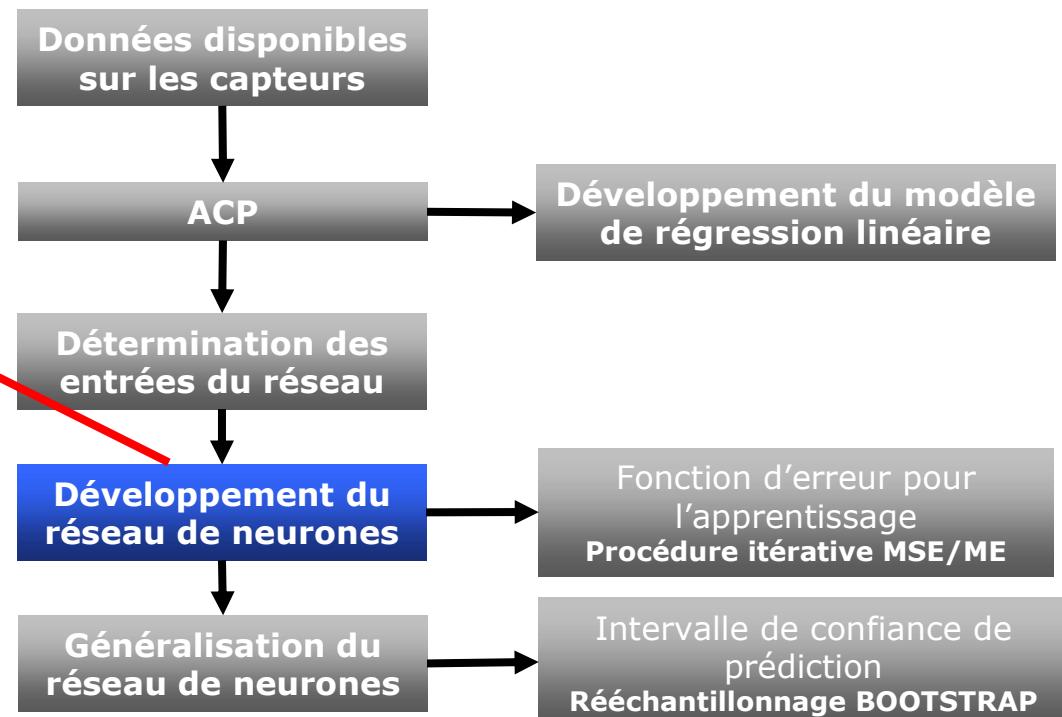


Détermination en ligne la dose de coagulant

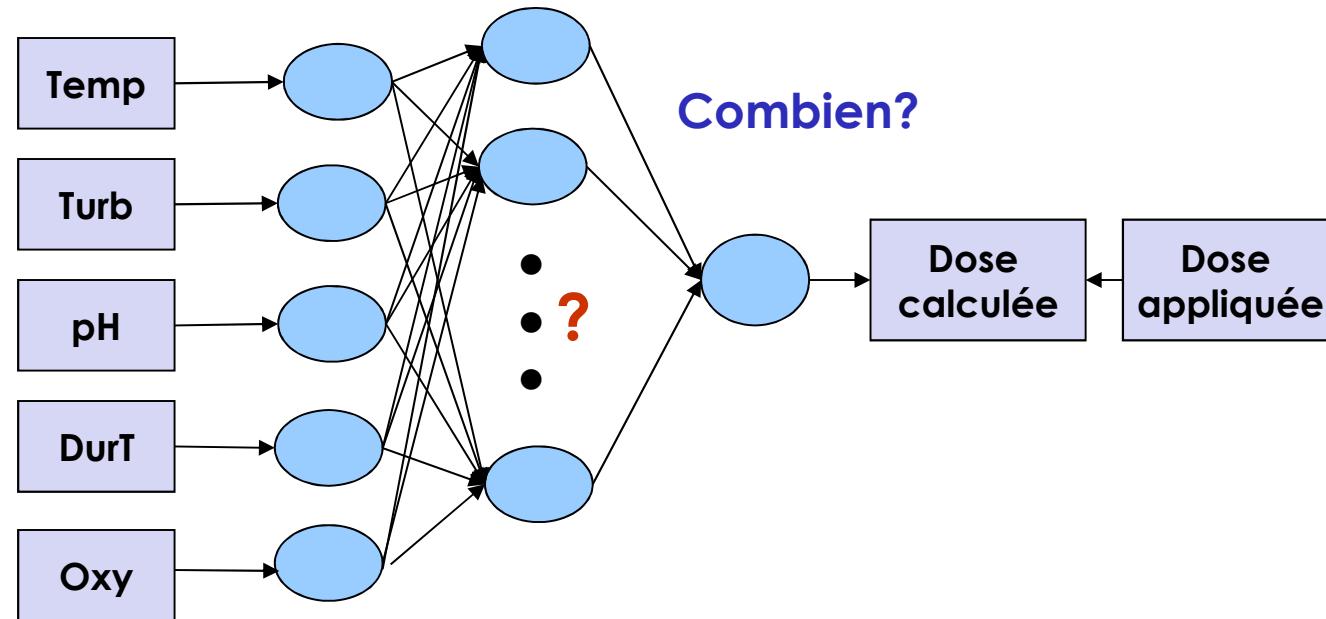


$$h_j = \sum_{i=1}^n w_{ji}x_i + w_{j0} \quad \text{et} \quad y(x) = f\left(\sum_{j=1}^H W_j h_j + W_0\right)$$

$y(x)$ = Sortie du perceptron multicouche
 f = Fonction d'activation (sigmoïde)



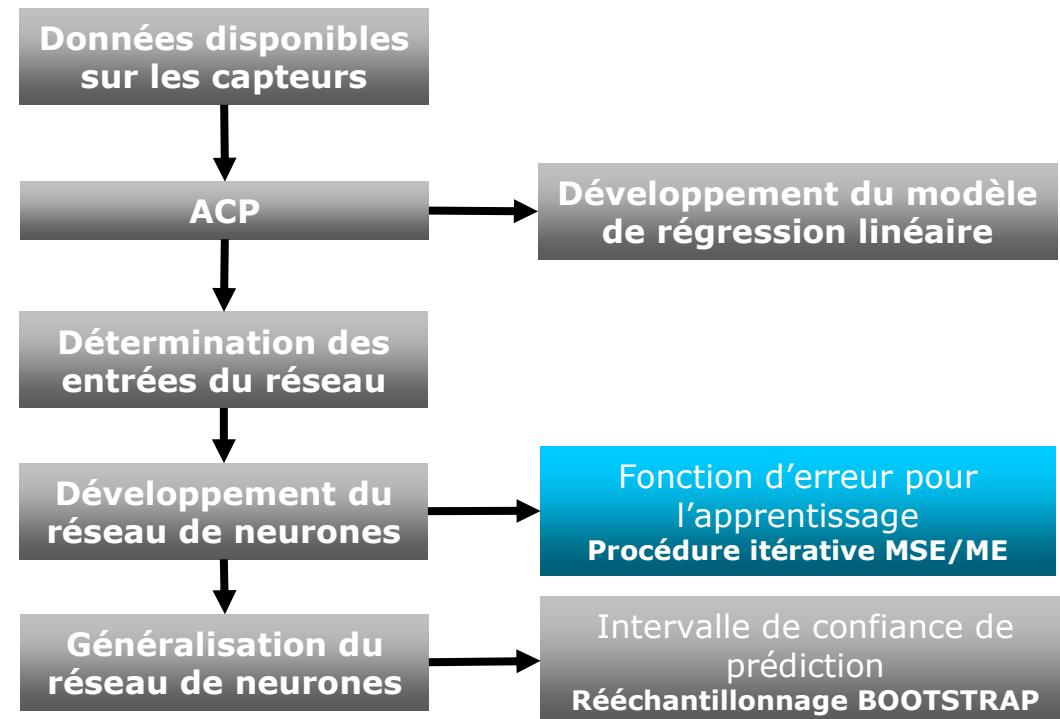
Détermination en ligne la dose de coagulant



- Nombre de neurones:
 - Couche d'entrée : 5 (Temp, Turb, pH, DuretéT, Oxy)
 - Couche de sortie : 1 (Dose de coagulant)
 - Couche cachée : Une seule (fonction continue)
- Procédure d'apprentissage: supervisé
- Architecture perceptron multicouche: rétropropagation
- Algorithme d'apprentissage: Levenberg Marquardt
- Fonction d'activation: sigmoïde



Détermination en ligne la dose de coagulant



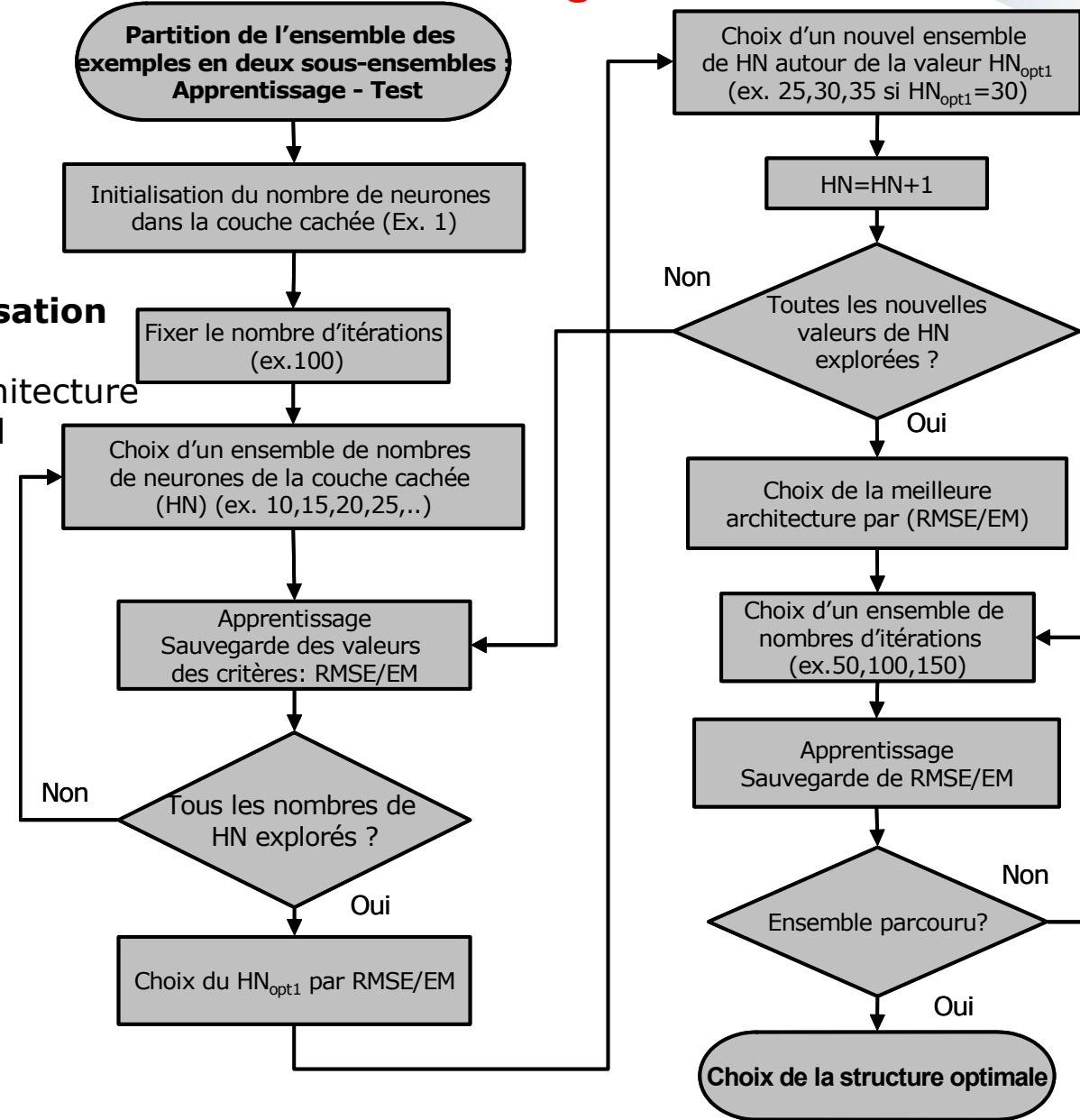
La fonction d'erreur MSE/ME (Erreur quadratique moyenne/Erreur moyenne)

→ Le choix de la fonction d'erreur utilisée pour l'apprentissage des réseaux de neurones multicouche a une certaine influence sur la rapidité d'apprentissage et sur la qualité de généralisation du réseau.

Détermination en ligne la dose de coagulant

Procédure de minimisation MSE/ME

Permet d'obtenir une architecture de réseau optimal



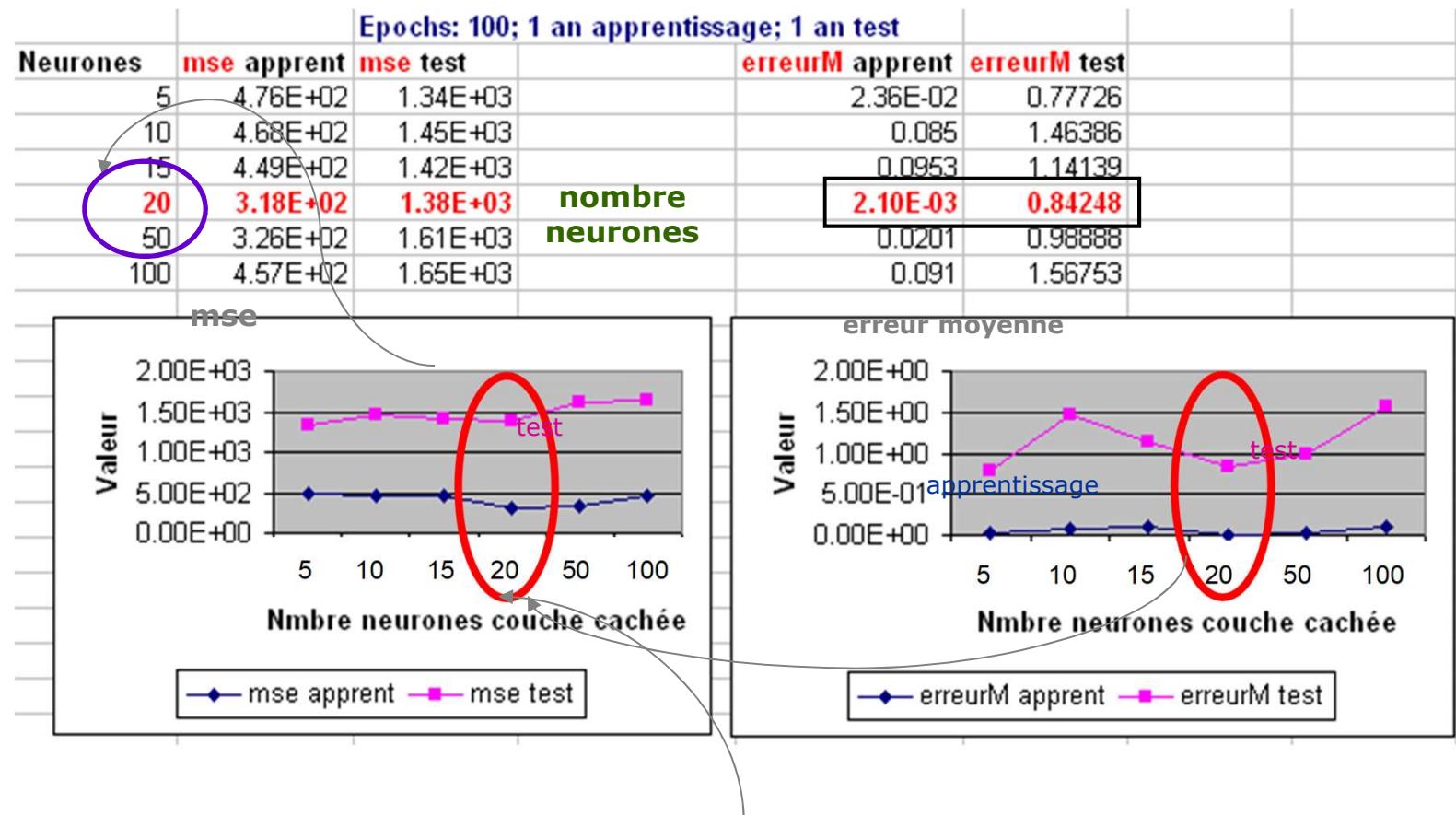
Neurones de la couche cachée: Combien ?

➤ Méthode:

- réservier une partie des données comme jeu de test
- essayer différents réseaux avec différents nombres de neurones
- effectuer l'apprentissage
- vérifier la performance (mse) et l'erreur (erreur moyenne)
- choisir le réseau donnant le meilleur résultat de généralisation sur le jeu de test



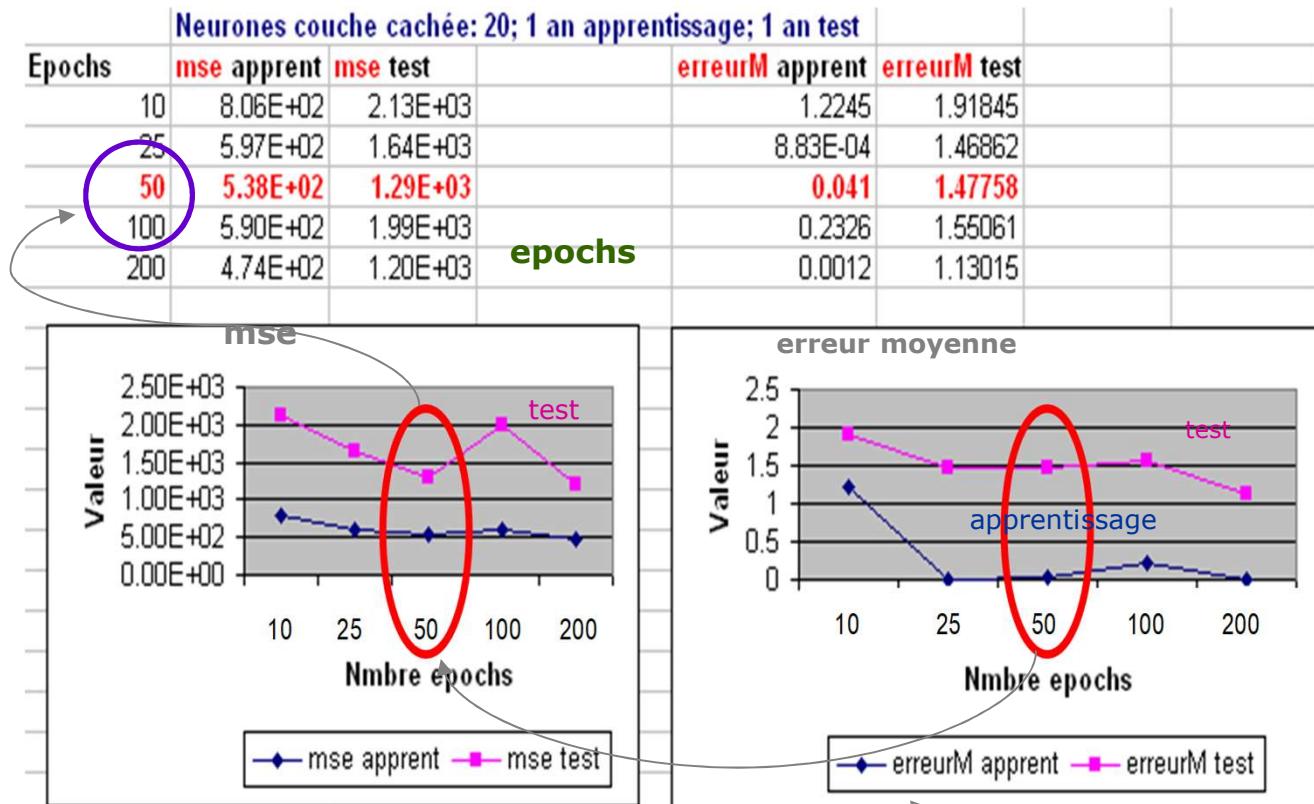
Détermination en ligne la dose de coagulant



Nbre neurones couche cachée: **20**



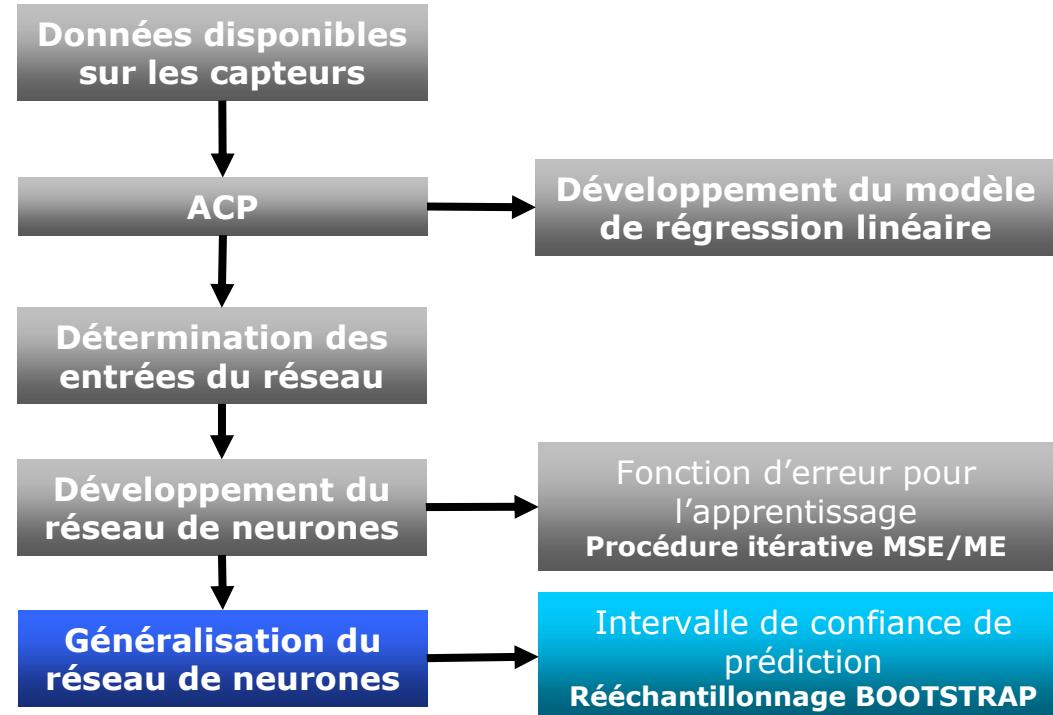
Détermination en ligne la dose de coagulant



nombre d'itérations epochs: **50**



Détermination en ligne la dose de coagulant



L'intervalle de confiance de prédiction (BOOTSTRAP)

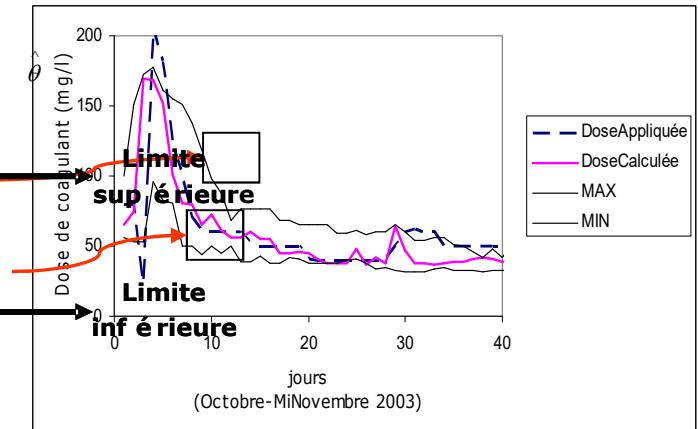
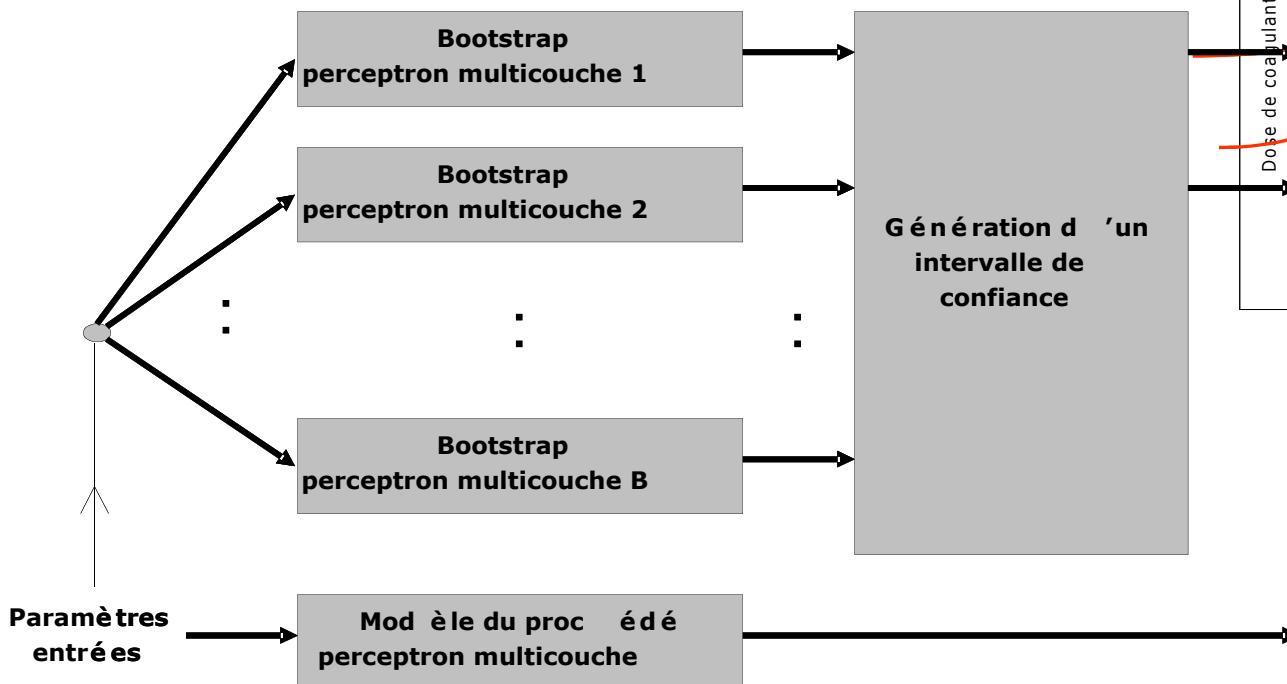
Technique d'inférence statistique qui crée un nouvel ensemble d'apprentissage par rééchantillonnage avec remise à partir de l'ensemble de départ

Détermination en ligne la dose de coagulant

Si $X = \{x^1, x^2, x^3, x^4, x^5\}$ = échantillon

$X_{boot} = \{x^1, x^1, x^2, x^5, x^5\}$ = échantillon bootstrap (fichier original)

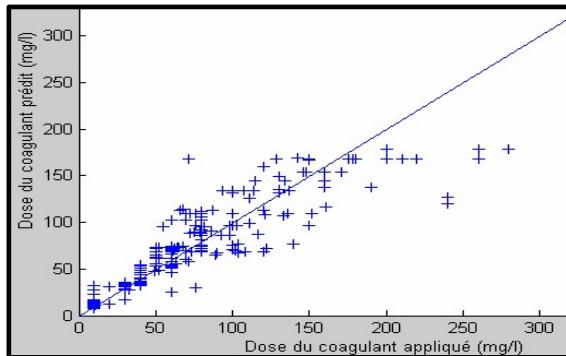
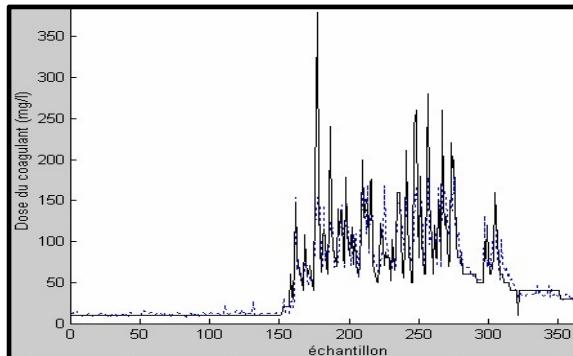
Si $(x_{boot}^1, \dots, x_{boot}^B)$ = sont B échantillons de bootstrap générés à partir de X
 $s(x_{boot}^1), \dots, s(x_{boot}^B)$ Distribution que approxime à la distribution de l'estimateur



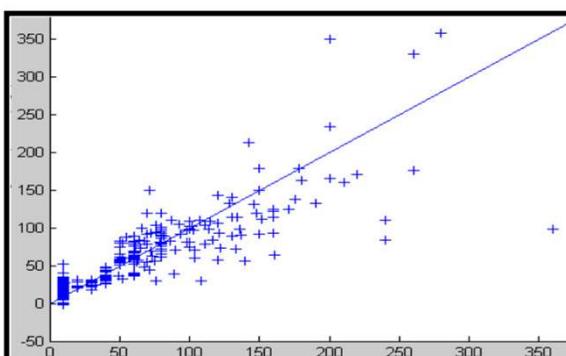
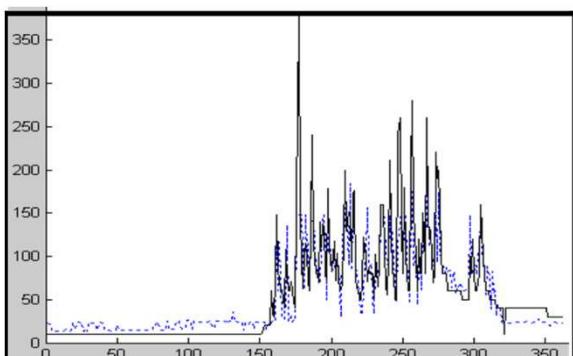
Dans cette approche, en se fixant un seuil à 10% et 90%, l'intervalle de confiance pour $B=50$ ->
l'estimateur du point 10% = 5 ème plus grande valeur

et l'estimateur du point 90% = 45 ème plus grande valeur

Détermination en ligne la dose de coagulant



Dose de coagulant appliquée et dose de coagulant prédictive (ligne pointillée et +)



Dose de coagulant appliquée et dose de coagulant prédictive (ligne pointillée et +) en utilisant le modèle par régression linéaire

Indices de comparaison	RNA	Régression linéaire
R^2 sur les données d'apprentissage	0.97	0.72
R^2 sur les données de test	0.96	0.61
Critère MSE sur les données d'apprentissage	619.7	859.3

Détermination en ligne la dose de coagulant

Problème de classification:

→ on a affaire à des formes (par exemples des chiffres manuscrits, des phonèmes, des pièces mécaniques, etc.) susceptibles d'appartenir à des catégories, ou classes, différentes.

Un classifieur est capable d'attribuer une classe à une forme inconnue qui lui est présentée, ou - caractéristique très importante - de refuser de lui attribuer une classe si la forme inconnue est trop éloignée des formes utilisées pour l'apprentissage.

Il permet d'obtenir une information beaucoup plus riche : *la probabilité d'appartenance de l'objet à chacune des classes.*

Détermination en ligne la dose de coagulant

Réalisation industrielles:

- la poste française : machines de tri automatique faisant intervenir des réseaux de neurones pour la reconnaissance du code postal.
- les réseaux de neurones sont utilisés industriellement pour la lecture automatique du montant écrit en toutes lettres (en écriture cursive) sur les chèques bancaires et postaux



Développé par la société A2iA

- Formulation générale du problème:

$$(PC) \begin{cases} \text{Min } f(x) & (1) \\ x \in \Re^n \\ \text{sous les contraintes :} \\ g_i(x) = 0 & (2) \\ h_j(x) \leq 0 & (3) \end{cases}$$



I Conditions du 1er ordre

Problème : Trouver $\hat{x} \in \mathbb{R}^n$, minimum absolu ou relatif de $f(x)$

$$\text{Min } f(x) \quad x \in \mathbb{R}^n$$

Hypothèse : on suppose que $f(x)$ est continue et continûment dérivable jusqu'au 1^{er} ordre
 Pour δx suffisamment petit, on peut donc effectuer un développement de $f(x)$ en série de Taylor jusqu'au 1^{er} ordre :

$$f(x + \delta x) \approx f(x) + f_x^T \delta x + O(\|\delta\|^2) \quad (1)$$

Si \hat{x} est un minimum relatif de $f(x)$, on doit avoir dans un certain voisinage de \hat{x} :

$$f(\hat{x} + \delta x) \geq f(\hat{x}) \quad \forall \delta x \text{ petit}, \delta x \in \mathbb{R}^n \quad (2)$$

$$(1)+(2) \rightarrow f_x^T \delta x \geq 0 \quad \forall \delta x \in \mathbb{R}^n \quad (\text{On néglige le terme } O(\|\delta\|^2))$$

Cette relation doit se conserver pour tout δx faible, et en particulier lorsque l'on change δx en $-\delta x$

$$\rightarrow \boxed{f_x(\hat{x}) = 0}$$



Au voisinage de \hat{x} , on a donc, au 1^{er} ordre :

$$f(\hat{x} + \delta x) \approx f(\hat{x}) = cste$$

La condition nécessaire d'extremum local, pour une fonction $f(x)$ continue et continûment dérivable au 1^{er} ordre, est que cette fonction soit stationnaire, c'est-à-dire que :

$$f_x(\hat{x}) = 0 \quad (3)$$

II Conditions du 2^{ème} ordre

Pour δx petit, $f(x)$ continue et continûment dérivable jusqu'au 2^{ème} ordre, nous avons vu que nous pouvons écrire :

$$f(x + \delta x) = f(x) + f_x^T \delta x + \frac{1}{2} \delta x^T F_{XX} \delta x + (o(\|\delta\|^3)) \quad (4)$$

F_{XX} : matrice dérivée d'élément $\frac{\partial^2 f}{\partial x_i \partial x_j}$, carrée, symétrique (appelée HESSIEN)

Si \hat{x} est un minimum relatif, on a au voisinage de \hat{x} :

$$f(\hat{x} + \delta x) \geq f(\hat{x})$$



Tenant compte de la condition nécessaire (condition du 1^{er} ordre $f_x(\hat{x}) = 0$), et en reportant dans (4) il reste si δx petit, en négligeant les termes du 3^{ème} ordre :

$$\delta x^T F_{XX}(\hat{x}) \delta x \geq 0 \quad \forall \delta x \in \Re^m \quad (5)$$



$F_{XX}(\hat{x})$ est non négative

Une condition nécessaire de minimum relatif est donc :

$$f_x(\hat{x}) = 0, F_{XX}(\hat{x}) \text{ matrice définie non négative} \quad (6)$$

Cependant, si $F_{XX}(\hat{x})$ est seulement définie non négative, il peut exister des $\delta x \neq 0$ pour lesquels :

$$\delta x^T F_{XX}(\hat{x}) \delta x = 0$$

Et donc dans ce cas : $f(\hat{x} + \delta x) = f(\hat{x}) + (o(\|\delta\|^3))$



Il est nécessaire de connaître le terme en O^3 pour conclure



Les conditions ci-dessus ne constituent pas une condition suffisante



En revanche, si $F_{XX}(\hat{x})$ est définie positive et $f_x(\hat{x}) = 0$, alors :

$$\delta x^T F_{XX}(\hat{x}) \delta x > 0 \quad \forall \delta x \neq 0$$

Les termes d'ordre 3 étant négligeables pour δx petit, $f(\hat{x})$ est un minimum local strict :

$$f(\hat{x} + \delta x) > f(\hat{x})$$

Une condition nécessaire de minimum relatif strict est donc :

$$f_x(\hat{x}) = 0, F_{XX}(\hat{x}) \text{ matrice définie positive} \quad (7)$$

III Fonctions convexes

Fonctions convexes 

$F_{XX}(x)$ non négative toujours satisfait

La condition nécessaire de minimum relatif se réduit donc à : $f_x(\hat{x}) = 0$
et on peut démontrer qu'il **n'y a pas de maximum**

(8)



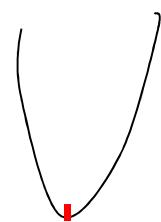
De plus minimum local (relatif) = minimum absolu (nécessairement)



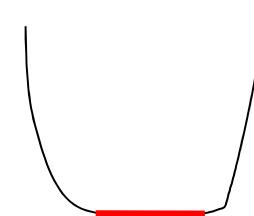
Si $f(x)$ est une fonction continue, dérivable, convexe et qu'en \hat{x} on ait :

$f'_x(\hat{x}) = 0$ alors \hat{x} est un minimum absolu de $f((x))$ (9)

De plus, si \hat{x} correspond à un minimum strict, alors c'est un minimum absolu unique



\hat{x} minimum strict



\hat{x} minimum large

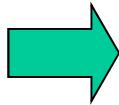


$$f(x_1, x_2) = 3x_1^4 - 4x_1^2 x_2 + x_2^2$$

Il est facile de voir que le point $(0, 0)$ est un point critique mais on ne peut pas conclure immédiatement

- Calcul du Hessien:

$$H(0,0) = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$$

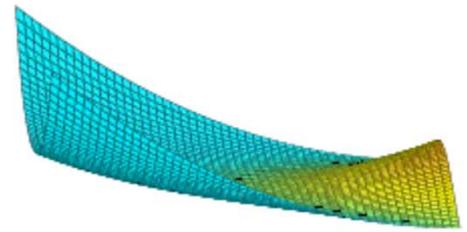


$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 36x_1^2 - 8x_2 & -8x_1 \\ -8x_1 & 2 \end{bmatrix}$$

Matrice définie non-négative :
valeurs propres = 0 et 2

$$f(0, 1) = 1 > f(0, 0) = 0 > f(0.5, 0.5) = -0.0625 .$$

Point $(0,0)$ ni un minimum ni un maximum



Soit la fonction continue et continûment dérivable au 2^{ème} degré en x :

$$\begin{cases} f(x) = a + b^T x + x^T C x \\ f_x(x) = b + 2Cx \end{cases}$$

C : matrice symétrique

Si $\det(C) \neq 0$



$$\hat{x} = -\frac{1}{2}C^{-1}b$$

Si $\det(C) = 0$

→
Utilisation de la pseudo – inverse de C et la solution n'est pas unique

La condition du 2^{ème} ordre donne : $F_{XX} = 2C$

Si C définie non-négative alors \hat{x} est un minimum

Si C est définie positive alors \hat{x} est un minimum absolu strict
(car minimum relatif strict unique $\det(C) \neq 0$)

Remarque : C définie non négative → f convexe et \hat{x} est un minimum absolu
→ Hypothèse de convexité permet de se limiter aux conditions du 1^{er} ordre



Exemple : la méthode des moindres carrés , régression linéaire

Considérons un phénomène physique qui lie deux grandeurs observées x et y selon une loi linéaire du type :

$$y = ax + b$$

On veut déterminer les coefficients a et b par N mesures (disons $N=3$) de y , x étant connu. On a théoriquement:

$$y_1 = ax_1 + b$$

$$y_2 = ax_2 + b$$

$$y_3 = ax_3 + b$$

Un tel système est en général surdéterminé et n'admet pas de solution pour a et b .

Supposons les mesures de y entachées d'erreurs r_i (appelées résidus). On a alors:

$$y_1 + r_1 = ax_1 + b$$

$$y_2 + r_2 = ax_2 + b$$

$$y_3 + r_3 = ax_3 + b$$

Cette fois on a trop d'inconnues, si l'on considère r_1 , r_2 et r_3 également à résoudre, en même temps que a , et b . Introduisons le vecteur de résidus:

$$r = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$



On veut déterminer a et b de sorte que l'ensemble des résidus soit minimal ou encore que $\|r\|$ soit minimale. Ce qui revient à poser, pour la somme des carrés:

$$\underset{a,b}{\text{Min}} \sum_{i=1}^3 r_i^2 = \underset{a,b}{\text{Min}} (r_1^2 + r_2^2 + r_3^2)$$

Exprimons les r_i en fonction de a et b

$$r_1 = ax_1 + b - y_1$$

$$r_2 = ax_2 + b - y_2$$

$$r_3 = ax_3 + b - y_3$$

En adoptant la notation matricielle on obtient: $r = Mp - y$

Avec :
$$r = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \quad M = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \quad p = \begin{pmatrix} a \\ b \end{pmatrix}$$

D'où :
$$\sum_{i=1}^3 r_i^2 = r^T r = (Mp - y)^T (Mp - y)$$



$$(Mp-y)^T(Mp-y) = p^T M^T M p - y^T M p - p^T M^T y + y^T y$$

$$= p^T Q p - 2S^T p + y^T y$$

Avec :

$$Q = M^T M = \begin{pmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 1+1+1 \end{pmatrix}$$

$$S = y^T M = \begin{pmatrix} x_1 y_1 + x_2 y_2 + x_3 y_3 \\ y_1 + y_2 + y_3 \end{pmatrix} = M^T y$$

$$\boxed{\min_{a,b} \sum_{i=1}^3 r_i^2 = \min_{a,b} (p^T Q p - 2S^T p + y^T y)}$$



Forme quadratique :

$$\begin{cases} f(p) = a + b^T p + p^T C p \\ f_p(p) = b + 2Cp \end{cases}$$

$$C = Q, b = -2S, a = y^T y$$

D'où :

$$\boxed{\hat{p} = -\frac{1}{2} C^{-1} b = Q^{-1} S}$$



Ou encore en développant :

$$p^T Q p = \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 1+1+1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a & a \end{pmatrix} \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

$$p^T Q p = q_{11} \cdot a^2 + 2q_{12} \cdot ab + q_{22} \cdot b^2$$

$$\frac{\partial(p^T Q p)}{\partial p} = \begin{pmatrix} \frac{\partial(p^T Q p)}{\partial a} \\ \frac{\partial(p^T Q p)}{\partial b} \end{pmatrix} = \begin{pmatrix} 2q_{11}a + 2q_{12}b \\ 2q_{12}a + 2q_{22}b \end{pmatrix} = 2 \begin{pmatrix} q_{11}a + q_{12}b \\ q_{12}a + q_{22}b \end{pmatrix} = 2Qp$$

$$2S^T p = 2 \begin{pmatrix} x_1 y_1 + x_2 y_2 + x_3 y_3 & y_1 + y_2 + y_3 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

$$2S^T p = 2(a(x_1 y_1 + x_2 y_2 + x_3 y_3) + b(y_1 + y_2 + y_3))$$

$$\frac{\partial(2S^T p)}{\partial p} = \begin{pmatrix} \frac{\partial(2S^T p)}{\partial a} \\ \frac{\partial(2S^T p)}{\partial b} \end{pmatrix} = 2 \begin{pmatrix} x_1 y_1 + x_2 y_2 + x_3 y_3 \\ y_1 + y_2 + y_3 \end{pmatrix} = 2S$$



LES METHODES DE DESCENTE

Soit $f: \mathbb{R}^n \rightarrow \mathbb{R}$.

On dira qu'un vecteur s est une direction de descente en x s'il existe $\omega^* > 0$ tel que :

$$f(x + \omega s) < f(x) \quad \omega \in]0, \omega^*]$$

Le principe d'une méthode de descente consiste à faire les itérations suivantes :

$$x_{k+1} = x_k + \omega_k s_k, \quad \omega_k > 0$$

tout en assurant la propriété :

$$f(x_{k+1}) < f(x_k)$$

Le vecteur s_k est la direction de descente en x_k .

Le scalaire ω_k est appelé le pas de la méthode à l'itération k

Hyp. : f est différentiable :

$$\lim_{\omega \rightarrow 0} \frac{f(x_k + \omega s_k) - f(x_k)}{\omega} = \left. \frac{df(x_k + \omega s_k)}{d\omega} \right|_{\omega=0} = s_k^T f_x(x_k)$$

Le vecteur s_k est la direction de descente en x_k si : $s_k^T f_x(x_k) < 0$

(dérivée directionnelle de $f(x_k)$ dans la direction s_k)



- La longueur de pas de descente w_k est un scalaire strictement positif ; c'est une mesure de la distance le long de la direction de descente s_k entre deux points successifs x_k et x_{k+1}
 - Différentes approches sont possibles pour la détermination du pas w_k
- pas de descente non optimal

$$f(x_k + \omega_k s_k) \leq f(x_k) \quad \omega_k > 0$$

pas de descente optimal

Etant donné un point x_k et une direction de descente s_k , un pas w_k est dit “pas optimal” s'il est déterminé comme solution du problème :

$$f(x_k + \omega_k s_k) = \underset{\omega}{\operatorname{Min}} \quad f(x_k + \omega s_k)$$

On est donc amené à résoudre un problème de minimisation monovariable par un méthode de recherche directe (par ex. Nombre d'or)



Détermination d'une direction de descente

- **Principe fondamental**

Soit M une matrice symétrique définie positive. Tout vecteur s_k défini par :

$$s_k = -M f_x(x_k)$$

est une *direction de descente*.

Démonstration :

$$(s_k)^T f_x(x_k) = -f_x^T(x_k) M^T f_x(x_k) = -f_x^T(x_k) M f_x(x_k) < 0$$

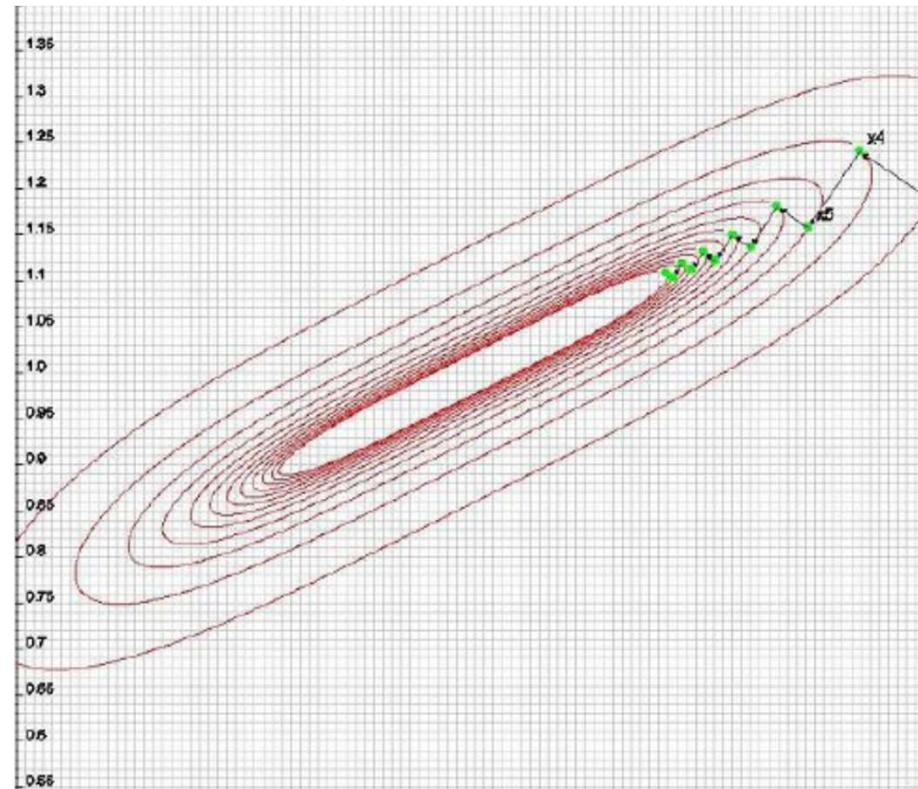
➔ Les méthodes proposées diffèrent par le choix de la matrice M

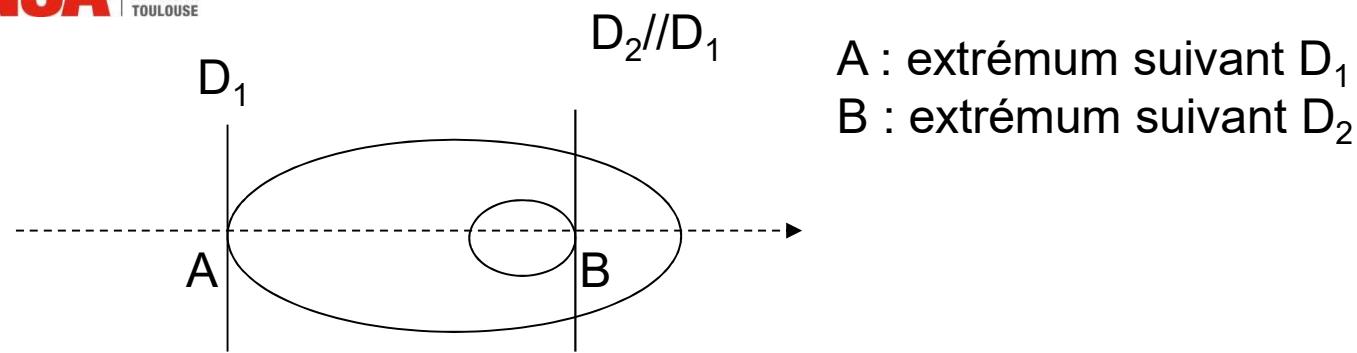
- Le choix le plus simple pour M est la matrice identité I .
La direction de descente est alors :

$$s_k = -f_x(x_k)$$

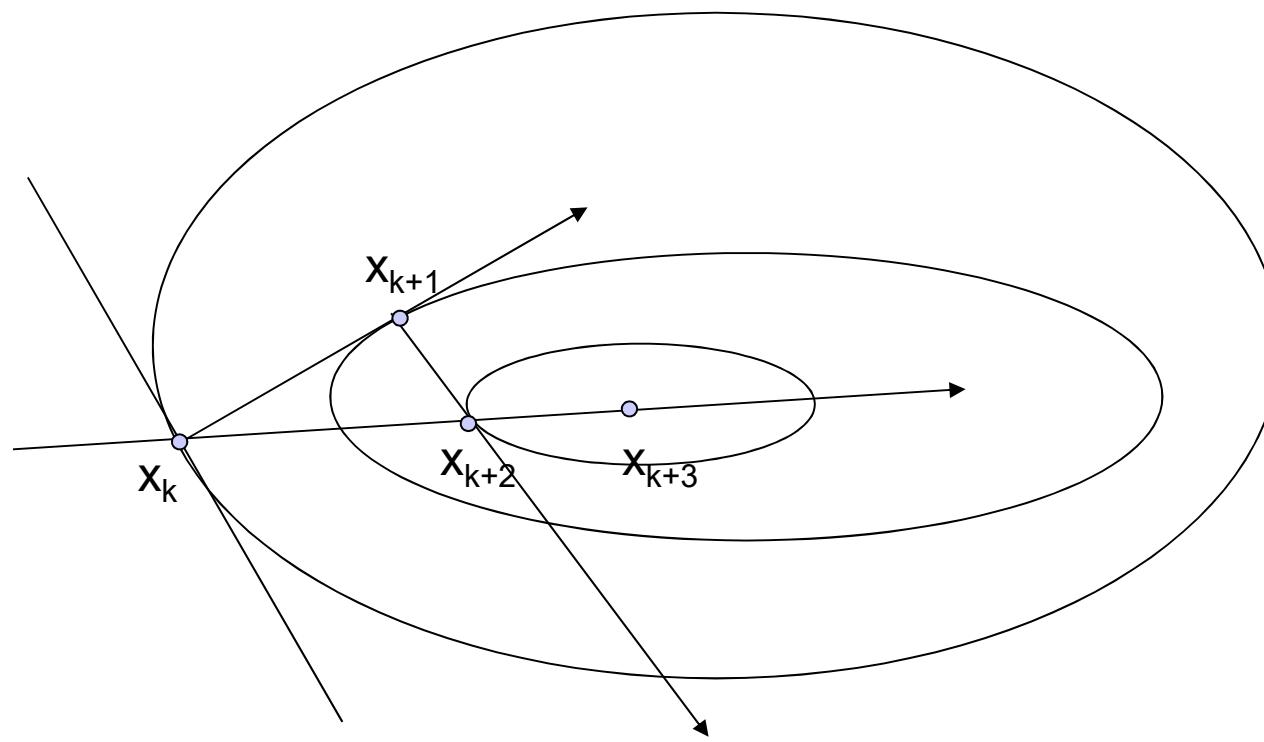
- Cette direction est appelée direction de la plus grande pente. Elle correspond au gradient de la fonction $f(x)$.
- Le calcul des directions de descente s_k ne nécessite que la connaissance des dérivées partielles premières de la fonction f (c'est pour cela que la méthode est dite d'ordre 1)

Minimiser $f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$
Méthode du gradient





A : extrémum suivant D_1
B : extrémum suivant D_2



METHODE PARTAN

- Méthode de base : la méthode de Newton

Le choix d'une direction de descente est fourni par le développement en série de TAYLOR au deuxième ordre de la fonction f au voisinage de x_k

$$f(x_k + \delta x_k) = f(x_k) + (\delta x_k)^T f_x(x_k) + 1/2 (\delta x_k)^T F_{xx}(x_k) (\delta x_k)$$

Le problème se ramène à la recherche d'une direction de descente s_k et d'un pas de descente w_k qui minimise la forme quadratique définie ci-dessus :

$$= \underset{\delta x_k}{\operatorname{Min}} \quad f(x_k + \delta x_k)$$

$$\underset{\delta x_k}{\text{Min}} \quad f(x_k + \delta x_k) \Rightarrow \frac{\partial}{\partial(\delta x_k)} f(x_k + \delta x_k) = 0$$

→ $f_x(x_k) + F_{xx}(x_k)(\delta x_k) = 0.0$

Si $F_{xx}(x_k)$ est non singulière (pas de valeurs propres nulles)

→ $\delta x_k = -F_{xx}(x_k)^{-1}f_x(x_k)$

On peut donc choisir comme direction de recherche :

$$s_k = -F_{xx}(x_k)^{-1}f_x(x_k)$$

et pour longueur du pas suivant cette direction : $\omega_k = 1$

Méthode de NEWTON

Remarque :

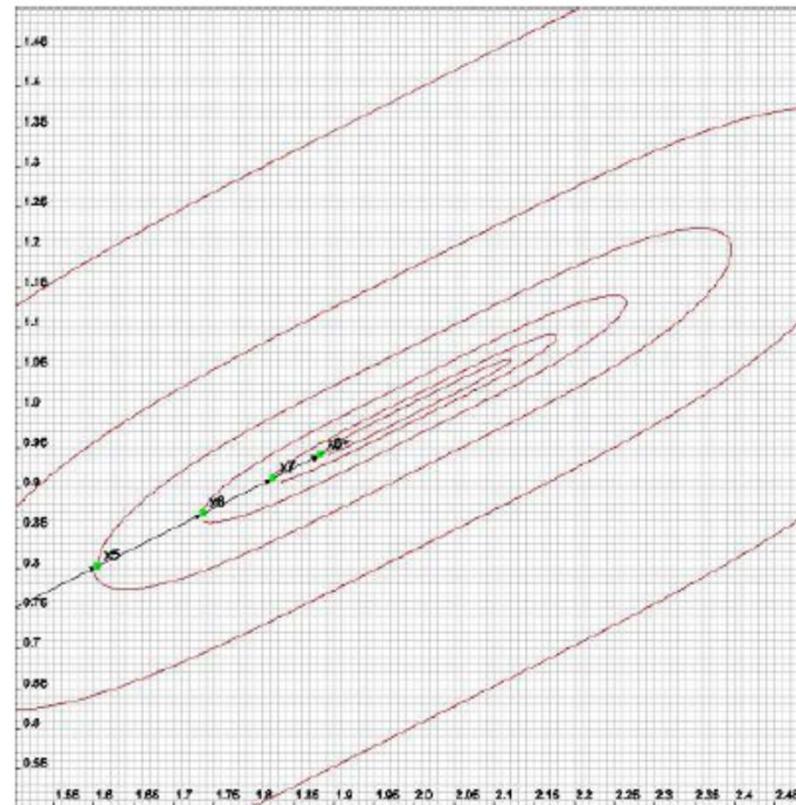
$$s_k^T f_x(x_k) < 0$$

$$-f_x(x_k)[F_{xx}(x_k)^{-1}]^T f_x(x_k) < 0$$

→ Le Hessien $F_{xx}(x_k)$ est défini positif (condition du 2nd ordre)



Minimiser $f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$
Méthode de Newton



→ méthodes de quasi-Newton, du type :

$$\begin{cases} S_k = -B_k f_x(x_k) \\ x_{k+1} = x_k + \omega_k S_k \end{cases} \quad \text{ou} \quad \begin{cases} S_k = -H_k^{-1} f_x(x_k) \\ x_{k+1} = x_k + \omega_k S_k \end{cases}$$

où B_k (respectivement H_k) est une matrice destinée à approcher l'inverse du Hessian de f (respectivement le Hessian de f).

Problème : quelle stratégie adopter pour effectuer cette approximation ?

On peut par exemple prendre $B_0=I$, mais comment ensuite mettre à jour l'approximation B_k au cours des itérations ?

Idée : on sait qu'au point x_k , le gradient et le Hessian vérifie la relation suivante:

$$f_x(x_{k+1}) = f_x(x_k) + F_{xx}(x_{k+1} - x_k) + o(x_{k+1} - x_k)$$



Si on suppose que l'approximation quadratique est bonne :

$$f_x(x_{k+1}) - f_x(x_k) = F_{xx}(x_{k+1} - x_k)$$

Cela conduit à la relation de quasi-Newton :

On dit que les matrices B_{k+1} , H_{k+1} vérifient une relation de quasi-Newton si on a:

$$H_{k+1}(x_{k+1} - x_k) = f_x(x_{k+1}) - f_x(x_k)$$

ou

$$x_{k+1} - x_k = B_{k+1}(f_x(x_{k+1}) - f_x(x_k))$$

Il reste un problème à résoudre :

Comment mettre à jour B_k ou H_k tout en assurant B_k (H_k) > 0 ?



$$B_{k+1} = B_k + \frac{s_k s_k^\top}{s_k^\top y_k} - \frac{B_k y_k y_k^\top B_k}{y_k^\top B_k y_k}$$

$$s_k = x_{k+1} - x_k.$$

$$y_i = \nabla f(x_{i+1}) - f(x_i)$$

Algorithme de Davidon-Fletcher-Powell

1. Choisir x_0 et B_0 définie positive quelconque (par exemple $B_0 = I$)
2. A l'itération k , calculer la direction de déplacement

$$d_k = -B_k \nabla f(x_k),$$

déterminer le pas optimal ρ_k et poser

$$x_{k+1} = x_k + \rho_k d_k.$$

3. Poser $s_k = \rho_k d_k$ et $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ puis calculer

$$B_{k+1} = B_k + \frac{s_k s_k^\top}{s_k^\top y_k} - \frac{B_k y_k y_k^\top B_k}{y_k^\top B_k y_k}.$$

4. Faire $k \leftarrow k + 1$. Retourner en 1 sauf si le critère d'arrêt est vérifié.

Comme critère d'arrêt on retiendra par exemple $\|g_{k+1}\| < \epsilon$.

Cet algorithme a un comportement remarquable dans le cas où f est une forme quadratique :



Méthodes spécifiques pour les problèmes de moindres carrés

Dans les problèmes de moindres carrés non linéaires, la fonction à minimiser prend en général la forme

$$f(x) = \frac{1}{2} \sum_{i=1}^m f_i(x)^2,$$

Quand on applique la méthode de Newton à la minimisation de f , on calcule le gradient de f qui a une forme particulière

$$\nabla f(x) = \sum_{i=1}^m \nabla f_i(x) f_i(x),$$

et le Hessien de f donné par :

$$\nabla^2 f(x) = \sum_{i=1}^m \nabla f_i(x) \nabla f_i(x)^\top + \sum_{i=1}^m f_i(x) \nabla^2 f_i(x).$$

Si on se place près de l'optimum, on supposera que les $f_i(x)$ sont petits et alors on négligera le deuxième terme et le Hessien est alors donné par :

$$H(x) = \sum_{i=1}^m \nabla f_i(x) \nabla f_i(x)^\top$$

On obtient alors la méthode de Gauss-Newton

$$\begin{cases} x_0 & \text{donné,} \\ H_k & = \sum_{i=1}^m \nabla f_i(x_k) \nabla f_i(x_k)^\top \\ x_{k+1} & = x_k - H_k^{-1} \nabla f(x_k). \end{cases}$$



→ Pour assurer la convergence globale de la méthode de Gauss-Newton, on peut remplacer la matrice H_k précédente par la matrice $H_k + lI$ où l est un réel positif
Si l est très grand on retrouve alors la méthode du gradient

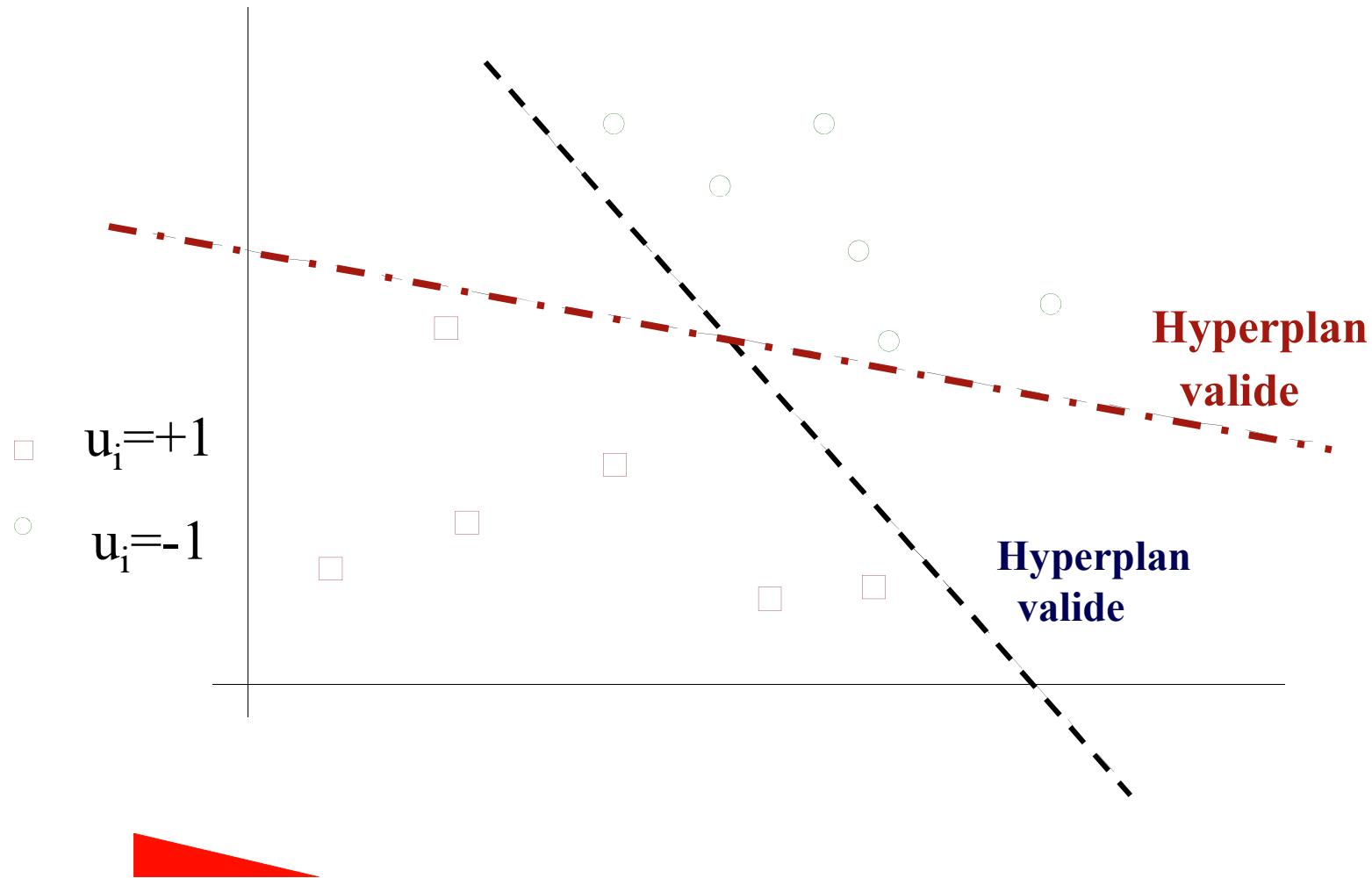
Méthode de Levenberg-Marquardt

$$\begin{cases} x_0 & \text{donné,} \\ H_k & = \sum_{i=1}^m \nabla f_i(x_k) \nabla f_i(x_k)^\top \\ d_k & = -(H_k + \lambda I)^{-1} \nabla f(x_k) \\ x_{k+1} & = x_k + \rho_k d_k, \end{cases}$$

Les SVM :
Machines à Supports Vectoriels ou Machines à Vecteurs de Support
(Support Vector Machines)

Apprentissage supervisé

Cas de deux classes linéairement séparables



- Tâche de classification

Cas de la séparation linéaire

- On cherche h sous forme d'une fonction linéaire : $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$
- La **surface de séparation** est donc l'hyperplan :

$$\mathbf{w} \cdot \mathbf{x} + b = 0 = D(x)$$

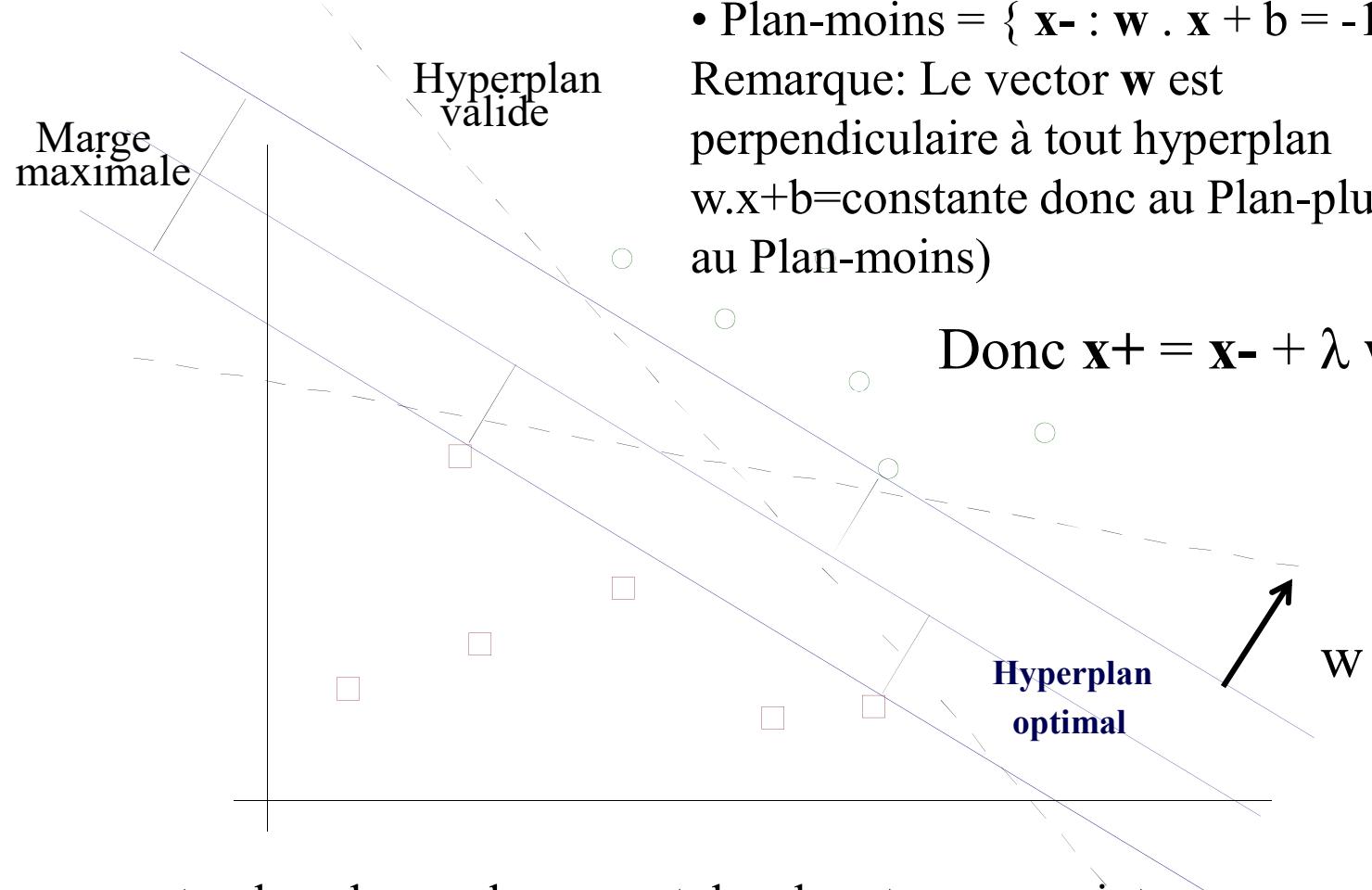
- Elle est valide si $\forall i \quad u_i h(\mathbf{x}_i) \geq 0 \quad (\text{u}_i=1 \text{ ou } \text{u}_i=-1)$

- L'hyperplan est dit sous forme canonique lorsque $\min_i |\mathbf{w} \cdot \mathbf{x}_i + b| = 1$
ou encore $\forall i \quad u_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$

- $\dim(\mathbf{x})$ = nombre des caractéristiques/attributs/informations prises en compte pour distinguer les individus



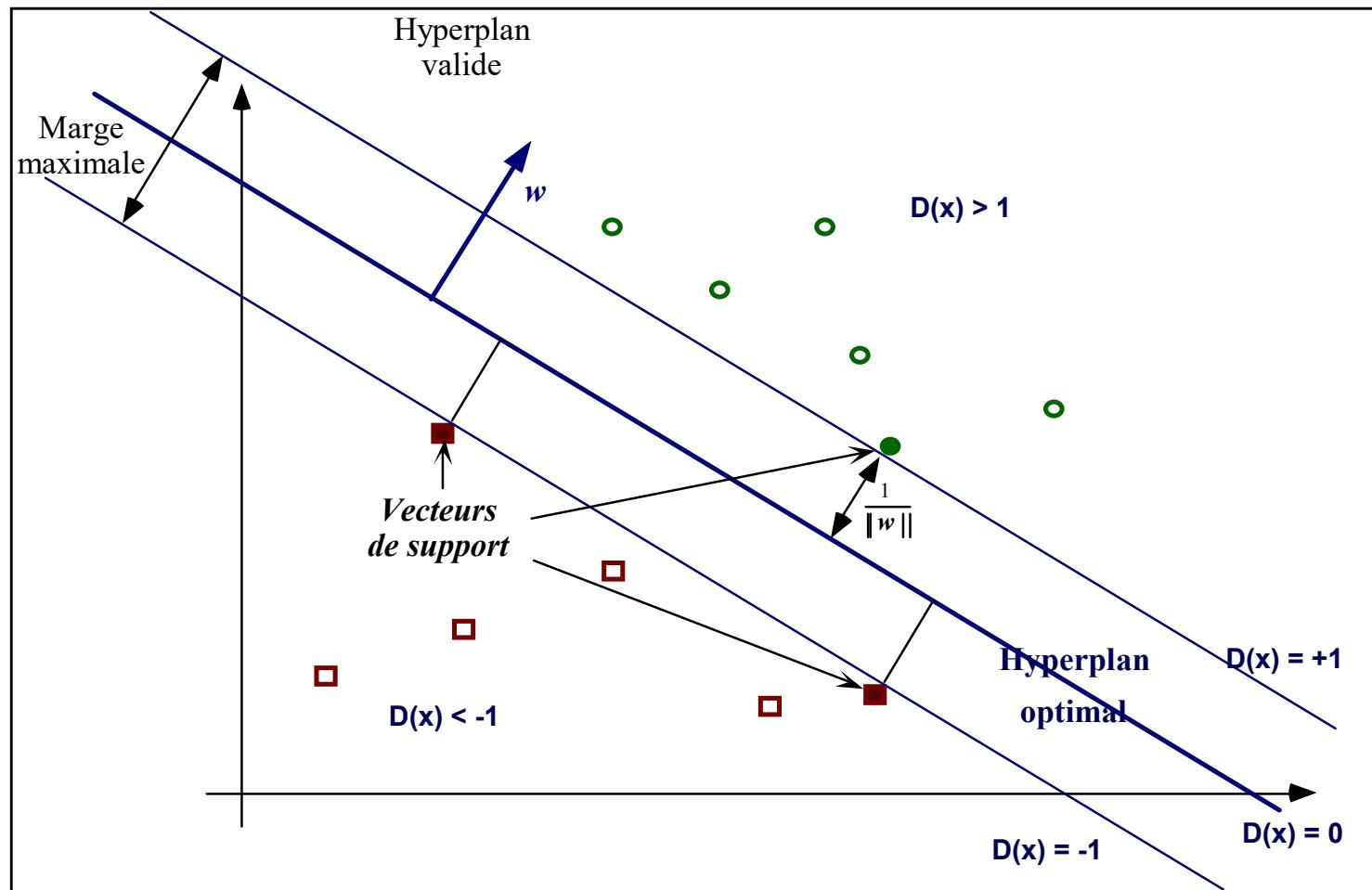
Hyperplan de plus vaste marge



- Plan-plus = { $\mathbf{x}^+ : \mathbf{w} \cdot \mathbf{x} + b = +1$ }
 - Plan-moins = { $\mathbf{x}^- : \mathbf{w} \cdot \mathbf{x} + b = -1$ }
- Remarque: Le vecteur \mathbf{w} est perpendiculaire à tout hyperplan $\mathbf{w} \cdot \mathbf{x} + b = \text{constante}$ donc au Plan-plus (et au Plan-moins)

Hyperplan de plus vaste marge

- Optimisation de la marge



$$\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$

$$\mathbf{w} \cdot \mathbf{x}^+ + b = +1$$

$$\bullet \mathbf{w} \cdot \mathbf{x}^- + b = -1$$

$$\bullet \mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$

$$\bullet |\mathbf{x}^+ - \mathbf{x}^-| = M$$

$$\mathbf{w} \cdot (\mathbf{x}^- + \lambda \mathbf{w}) + b = 1$$

=>

$$\mathbf{w} \cdot \mathbf{x}^- + b + \lambda \mathbf{w} \cdot \mathbf{w} = 1$$

=>

$$-1 + \lambda \mathbf{w} \cdot \mathbf{w} = 1 \Rightarrow \lambda = 2 / (\mathbf{w} \cdot \mathbf{w}) = 2 / \|\mathbf{w}\|^2$$

$$M = \|\mathbf{x}^+ - \mathbf{x}^-\| = \|\lambda \mathbf{w}\| = \lambda \|\mathbf{w}\|$$



- La distance d'un point à l'hyperplan est :
- L'hyperplan optimal est celui pour lequel la distance aux points les plus proches (**marge**) est maximale. Cette distance vaut

$$d(\mathbf{x}) = \frac{|\mathbf{w} \cdot \mathbf{x} + w_0|}{\|\mathbf{w}\|}$$

$$\mathbf{w} \cdot \mathbf{x}_1 + b = +1$$

$$(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$$

$$\text{Donc : } (\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2$$

$$\text{D'où : } (\mathbf{w}/\|\mathbf{w}\| \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2/\|\mathbf{w}\|$$

- Maximiser la marge revient donc à minimiser $\|\mathbf{w}\|$ sous contraintes:

$$\begin{cases} \min \frac{1}{2} \|\mathbf{w}\|^2 \\ \forall i \quad u_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 \end{cases}$$



→ un problème d'**optimisation quadratique**

- Il faut donc déterminer w et w_0 minimisant :

$$\eta(w) = \frac{1}{2} \|w\|^2$$

**EXPRESSION
PRIMAIRE**

(afin de maximiser le pouvoir de généralisation)

- sous les contraintes (hyperplan séparateur) :

$$u_i [(w \cdot x_i) + w_0] \geq 1, \quad i = 1, \dots, n$$

Nombre d'individus pour apprentissage

Résolution de la forme primaire du problème

d : dimension de l'espace d'entrée

Il faut régler $d + 1$ paramètres

- Possible quand d est assez petit
avec des méthodes d'optimisation quadratique
- Impossible quand d est grand ($> \text{qqs } 10^3$)

Transformation du problème d'optimisation

- Méthode des multiplicateurs de Lagrange

$$\begin{cases} L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i \{(\mathbf{x}_i \cdot \mathbf{w} + w_0) u_i - 1\} \\ \forall i \quad \alpha_i \geq 0 \end{cases}$$

- Problème dual

**EXPRESSION
DUALE**

$$\begin{cases} \max_{\alpha} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j u_i u_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \forall i \quad \alpha_i \geq 0 \\ \sum_{i=1}^l \alpha_i u_i = 0 \end{cases}$$



Transformation du problème d'optimisation

- **Propriétés de la forme duale**
- La conversion est possible car les fonctions de coût et les contraintes sont strictement convexes (*Th. de Kuhn-Tucker*)
- La complexité du problème d'optimisation est
 - $\propto m$ (taille de l'échantillon d'apprentissage) et
 - non $\propto d$ (taille de l'espace d'entrée X)

Possible d'obtenir des solutions pour des problèmes impliquant $\approx 10^5$ exemples

Propriété 1 : seuls les α_i correspondant aux points les plus proches sont non-nuls. On parle de ***points de support*** (*exemples critiques*).

Propriété 2 : seuls interviennent les **produits scalaires entre les observations x** dans le problème d'optimisation.

$$\left\{ \begin{array}{l} D(\mathbf{x}) = (\mathbf{w}^* \cdot \mathbf{x} + w_0^*) \\ \text{où : } \mathbf{w}^* = \sum_{i=1}^m \alpha_i^* u_i \mathbf{x}_i \\ \text{et } w_0^* = u_s - \sum_{i=1}^m \alpha_i^* u_i (\mathbf{x}_i \cdot \mathbf{x}_s) \end{array} \right.$$

* : estimé

(x_s, u_s) étant n'importe quel point de support

La majorité des problèmes !!!

Idée :

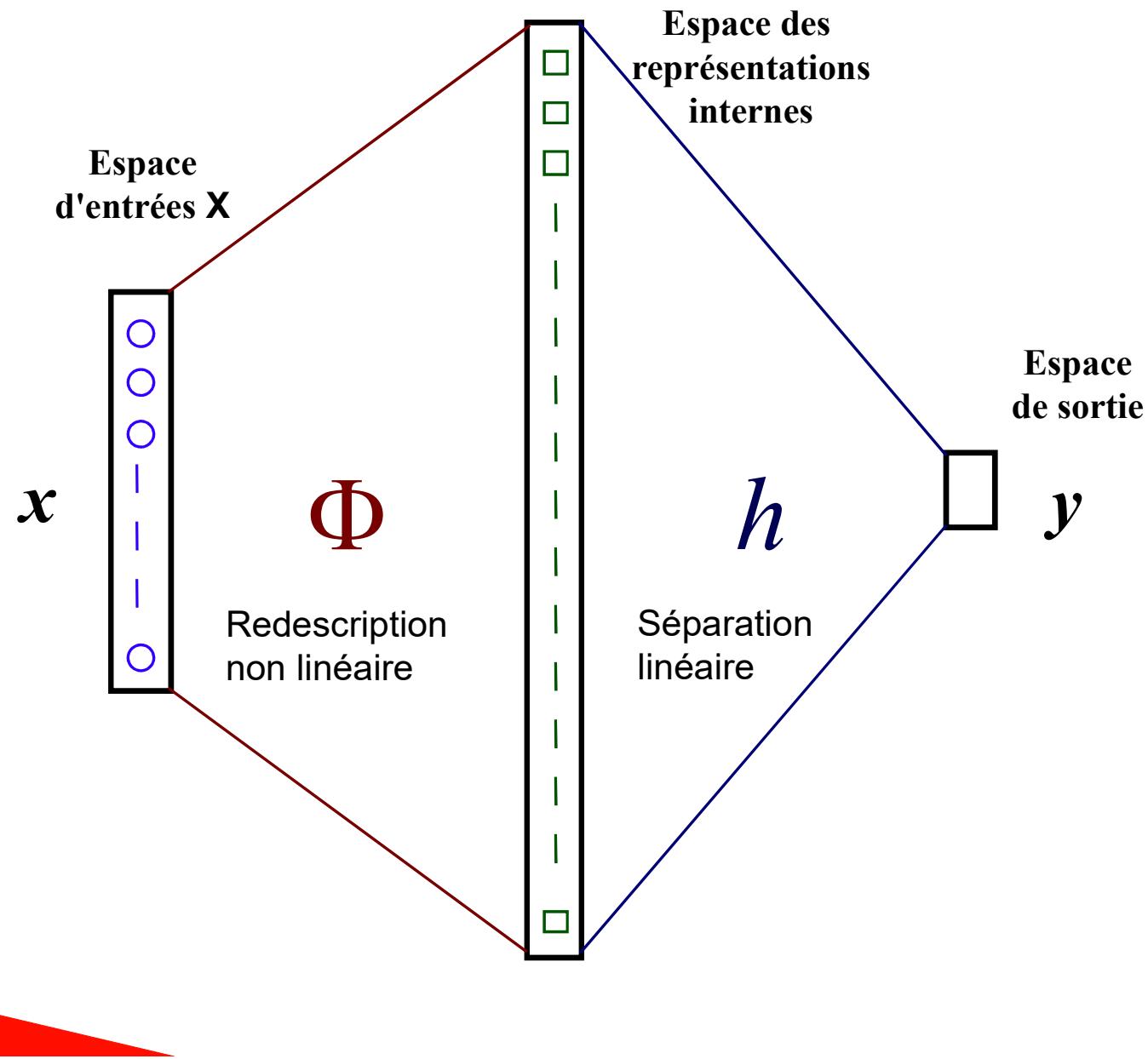
Si on projette dans un espace de redescription de très grande dimension ??

Presque toujours le problème devient linéairement séparable

Mais :

- ❖ Fléau de la dimensionnalité
- ❖ *Le nombre de paramètres explose !!?*

SVM et redescription



- Soit $\Phi : X \rightarrow \Phi(X)$, on peut remplacer partout x par $\Phi(x)$
- Si Φ est bien choisie, $K(x, x') = \Phi(x) \cdot \Phi(x')$ peut être facile à calculer et le problème devient :

$$\begin{cases} \max_{\alpha} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j u_i u_j K(x_i, x_j) \\ \forall i \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^l \alpha_i u_i = 0 \end{cases}$$



Solution du nouveau problème d'optimisation

- La fonction de décision devient :

$$D(\mathbf{x}) = \sum_{j=1}^n w_j g_j(\mathbf{x})$$

n : nb de fcts
de base
(peut être
très grand)

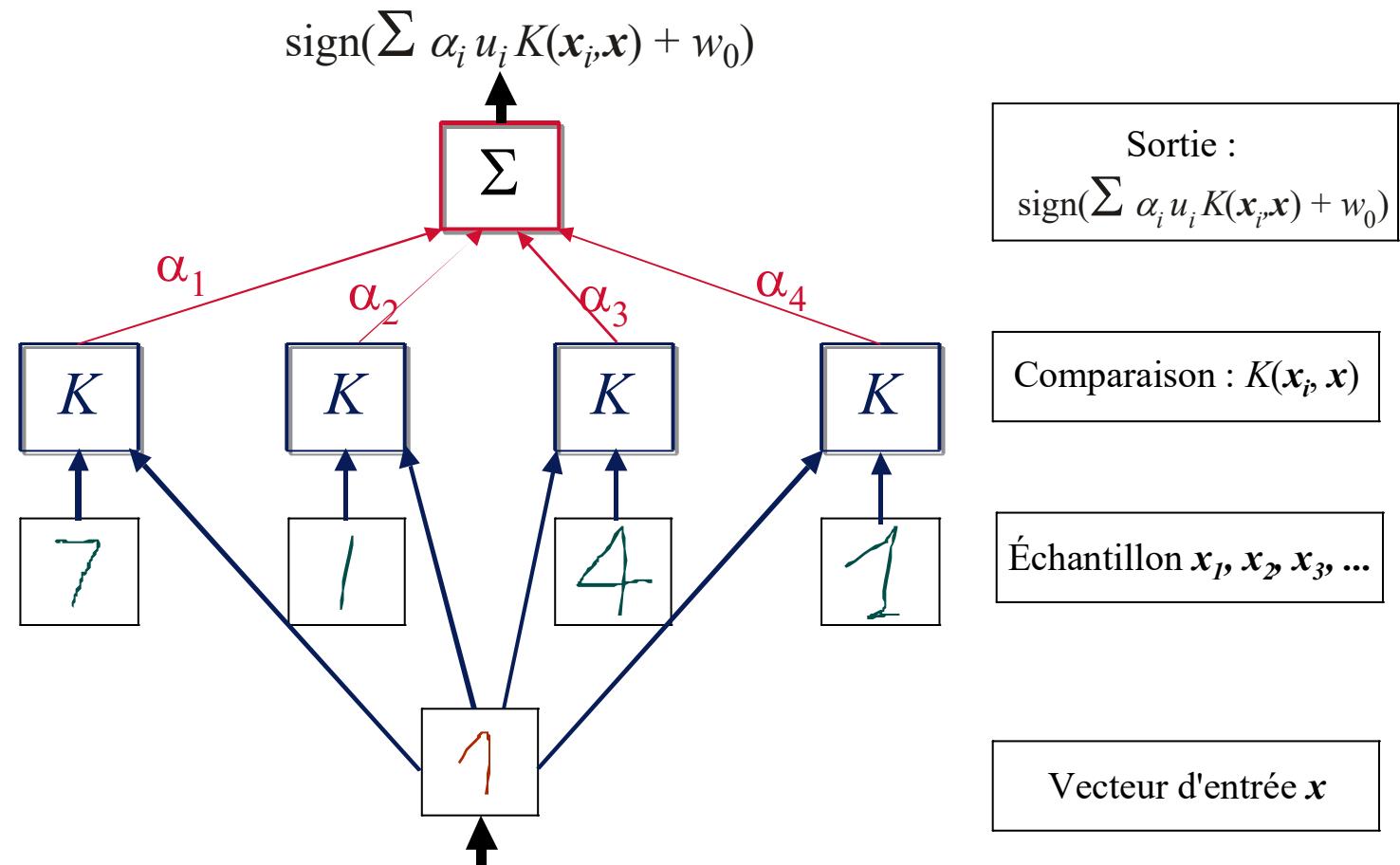
- Soit dans la forme duale :

$$D(\mathbf{x}) = \sum_{i=1}^{m_S} \alpha_i u_i K(\mathbf{x}_i, \mathbf{x}) + w_0$$

m_S : nb de points
de support



Schéma de fonctionnement des SVMs



- Si on prend une fonction K symétrique, **il existe** une fonction Φ tq:

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = \sum_{i=1}^m g_i(\mathbf{x}) \cdot g_i(\mathbf{x}')$$

ssi, pour toute fonction f telle que :

$$\int f(\mathbf{x})^2 d\mathbf{x} \text{ est finie}$$

l'on a :

$$\int K(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

- Si cette condition est vérifiée, on peut appliquer les SVMs
- **MAIS** cela ne dit pas comment construire Φ



- **Polynomiale :**

Les polynomes de degré q ont pour fonction noyau associée :

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^q$$

- **RBF :**

Les fcts à base radiale :

$$h(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^n \alpha_i \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\sigma^2} \right\} \right)$$

ont pour fct noyau associée :

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}}$$

- **Sigmoïde :**

Les réseaux de neurones à fcts d'activation :

$$h(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^n \alpha_i \tanh \{ v(\mathbf{x} \cdot \mathbf{x}_i) + a \} + b \right)$$

ont pour fct noyau associée :

$$K(\mathbf{x}, \mathbf{x}') = \tanh (a \mathbf{x} \cdot \mathbf{x}' - b)$$



- Construction à partir de fonctions noyau de base
(Propriétés de clôture)

$$K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) + K_2(\mathbf{x}, \mathbf{z})$$

$$K(\mathbf{x}, \mathbf{z}) = a K_1(\mathbf{x}, \mathbf{z})$$

$$K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) \cdot K_2(\mathbf{x}, \mathbf{z})$$

...

- Construction de fonctions noyau dédiées

Splines B_m

Expansion de Fourier

Ondelettes

...



- ... encodent :

Une **mesure de similarité** sur les données

$$d(x, y) = \sqrt{K(x - y, x - y)}$$

$$d(x, y) = \sqrt{K(x, x) - 2K(x, y) + K(y, y)}$$

La **forme fonctionnelle des fonctions de décision**

Le **type de régularisation** réalisée

(ex : les fcts gaussiennes favorisent les solutions régulières)

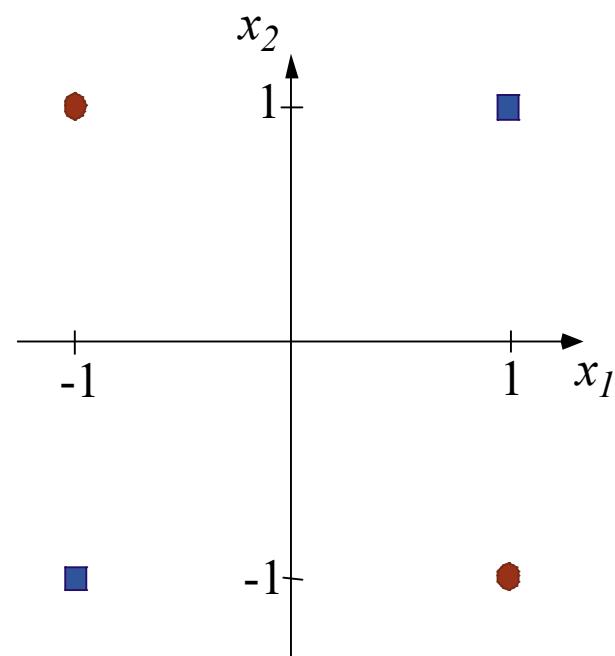
Le **type de covariance** dans l'espace des entrées

(ex : fcts noyau invariantes par rotation)

Sorte de **distribution de probabilité *a priori*** sur l'espace des hypothèses



Illustration : le cas du XOR



Index i	x	u
1	(1,1)	1
2	(1,-1)	-1
3	(-1,-1)	1
4	(-1,1)	-1

Fonction noyau polynomiale de d° 2 :

$$K(\mathbf{x}, \mathbf{x}') = [1 + (\mathbf{x}^\top \cdot \mathbf{x}')]^2$$

soit : $K(x, x_i) = 1 + x_1^2 x_{i1}^2 + 2 x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1}$
 $+ 2x_2 x_{i2}$

correspondant à la projection Φ :

$$[1, x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2} x_1, \sqrt{2} x_2]^\top$$



Illustration : le cas du XOR

Ici :

$$\begin{cases} \max_{\alpha} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j u_i u_j K(x_i, x_j) = Q(\alpha) \\ \forall i \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^l \alpha_i u_i = 0 \end{cases}$$

$$\begin{aligned} Q(\alpha) &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ &\quad - \frac{1}{2} (9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ &\quad + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \end{aligned}$$



Illustration : le cas du XOR

- L'optimisation de $Q(\alpha)$ en fonction des multiplicateurs de Lagrange conduit au système d'équations :

$$\begin{cases} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \\ \alpha_1 - 9\alpha_2 - \alpha_3 + \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - 9\alpha_3 + \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \end{cases}$$

- La valeur optimale des multiplicateurs de Lagrange est :

$$\alpha_1^* = \alpha_2^* = \alpha_3^* = \alpha_4^* = \frac{1}{8}$$



$$\text{soit : } Q^*(\alpha) = \frac{1}{4}$$

$$\frac{1}{2} \|w^*\| = \frac{1}{4} \|w\| = \frac{1}{\sqrt{2}}$$

Illustration : le cas du XOR

- Les 4 exemples sont donc des **exemples critiques** ("support vectors")
 $(\forall i, \alpha_i \neq 0)$
- La fonction de décision s'écrit : $D(\mathbf{x}) = \sum_{i=1}^{m_s} \alpha_i u_i K(\mathbf{x}_i, \mathbf{x}) + w_0$

$$D(\mathbf{x}) = \frac{1}{8} \sum_{i=1}^4 u_i [(x_i \cdot \mathbf{x}) + 1]^2$$

Illustration : le cas du XOR

En revenant dans l'espace d'origine :

Le **vecteur de poids optimal** est :

$$w^* = \frac{1}{8} [-\Phi(x_1) + \Phi(x_2) + \Phi(x_3) - \Phi(x_4)]$$

soit :

$$w^* = \frac{1}{8} \left\{ \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \right\} = \begin{pmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



Illustration : le cas du XOR

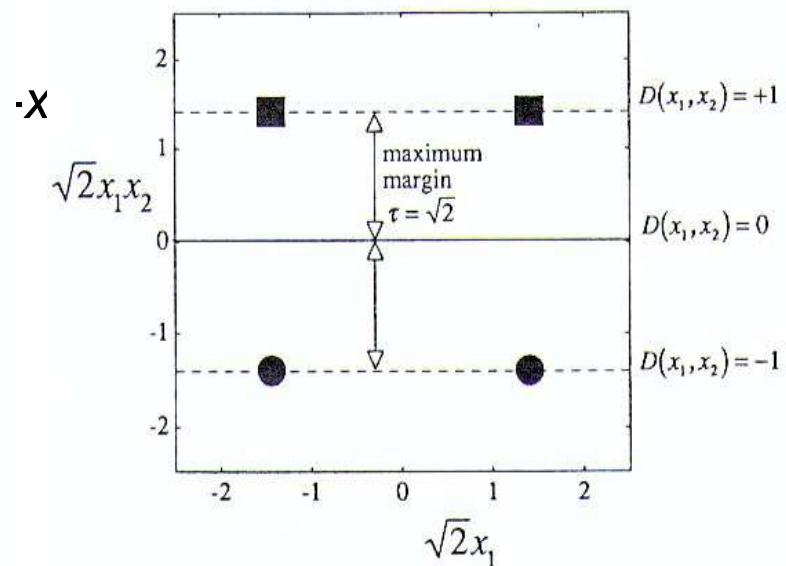
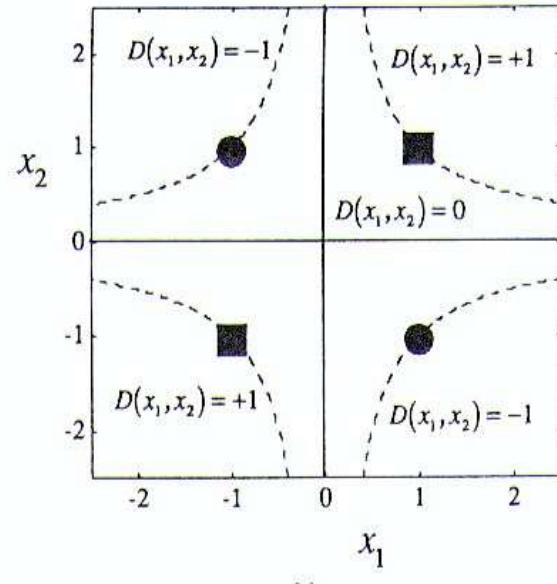
L'hyperplan optimal correspond à :

$$w^{*T} \cdot \Phi(x) = \left(0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0\right) \begin{pmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} = -x_1x_2 = 0$$



Illustration : le cas du XOR

Séparatrice dans l'espace d'entrée



Séparatrice dans l'espace $\Phi(\mathbf{X})$
(espace à 6 dimensions)

$$\sqrt{2} x_1 x_2 = 0$$



Cas du problème non séparable /marges douces

- On introduit des variables “ressort” qui pénalisent l’erreur commise :

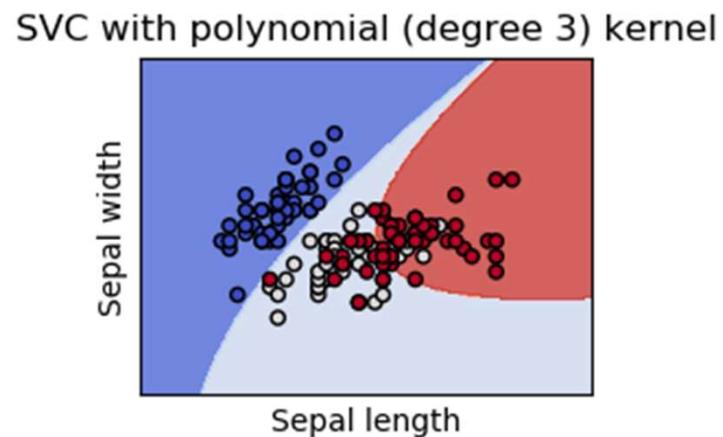
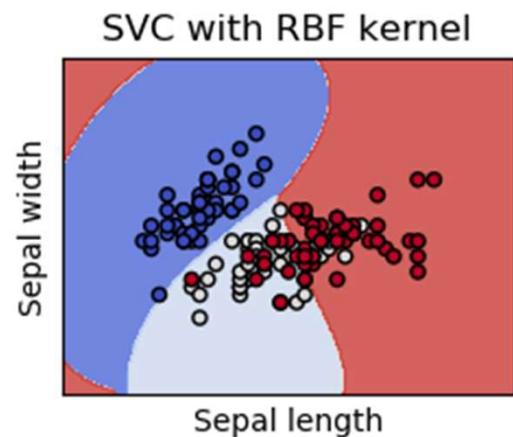
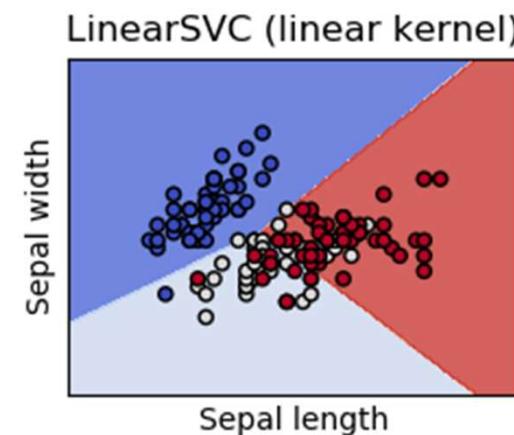
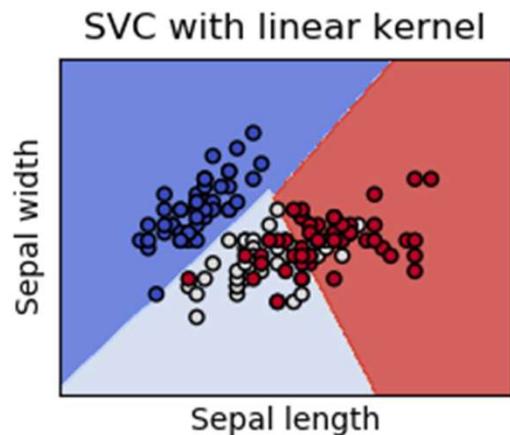
$$\begin{cases} \min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i \\ \forall i \quad u_i(w \cdot x_i + w_0) \geq 1 - \xi_i \end{cases}$$

- Le problème dual a la même forme à l’exception d’une constante C**

$$\begin{cases} \max_{\alpha} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j u_i u_j (x_i \cdot x_j) \\ \forall i \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^l \alpha_i u_i = 0 \end{cases}$$



EXEMPLE



Problème MULTICLASSE :
Deux solutions : 1 contre un ou un contre le reste

- Deux solutions :
- **1 contre un**
- ➔ si n_class est le nombre de classes, alors $n_class * (n_class - 1) / 2$ classifieurs sont construits et chacun d'eux est entraînés sur les données d'entraînement des deux classes.
- **un contre le reste**
- ➔ apprentissage de n classifieurs

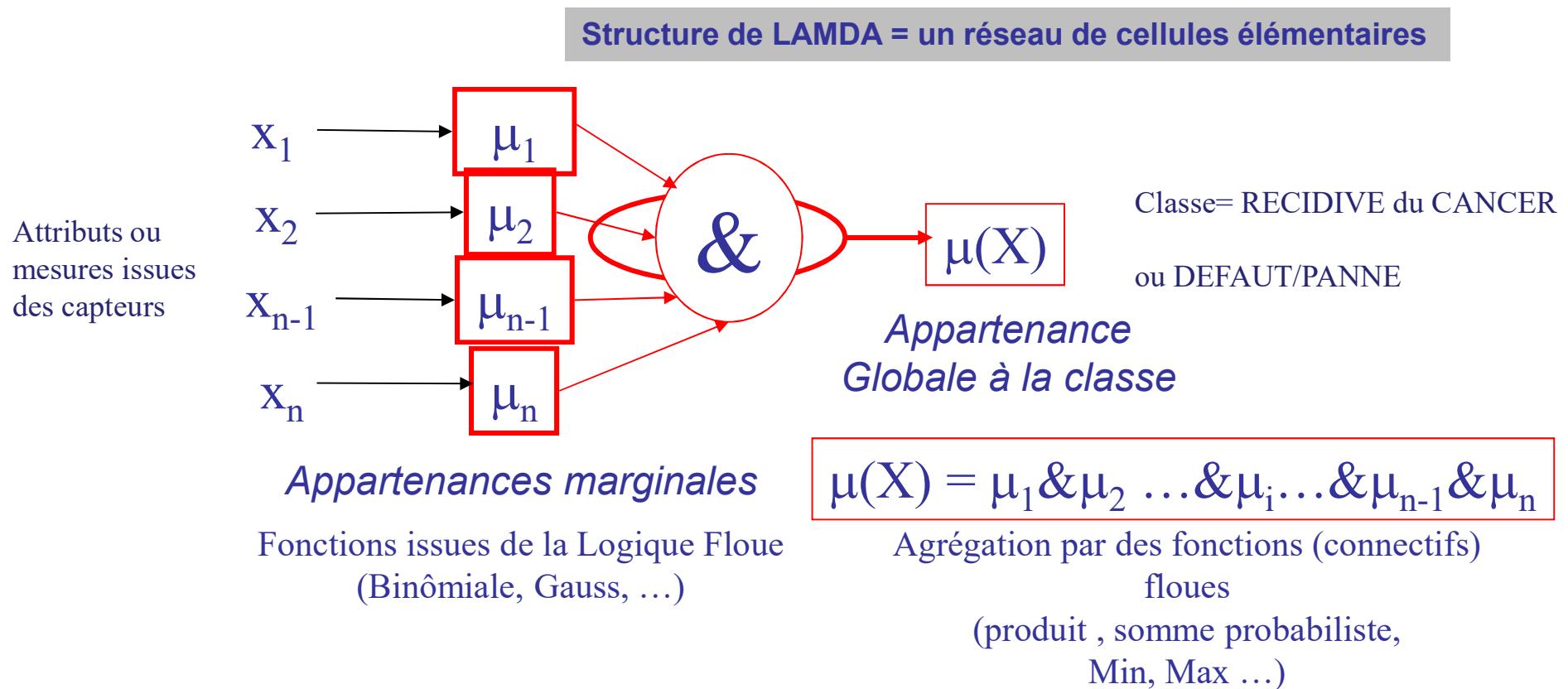
La méthode LAMDA

- ☞ Méthode issue de la Logique Floue basée sur le calcul de l'appartenance d'un individu (décrit par des *attributs*) à chaque classe à partir de l'analyse de similitude.
- ☞ Propriétés:
 - ✚ Possibilité d'un traitement simultané de trois types d'attributs: quantitatifs, qualitatifs (symboliques), intervalaires.
 - ✚ Apprentissage séquentiel Supervisé (avec professeur) et Non Supervisé (pas besoin de connaître le nombre de classes à priori).
 - ✚ Même algorithme d'apprentissage.
 - ✚ Différentes partitions d'un même ensemble de données avec le concept "d'exigence"
- ☞ Procédure: - Pour chaque classe k:
 - 1- Un Degré d'Adéquation Marginal (DAM) est calculé pour chaque attribut .
 - 2- Un Degré d'Adéquation Global (DAG) est calculé pour chaque élément à partir des DAMs.
 - 3- L'attribution de l'élément à la classe de DAG maximum.

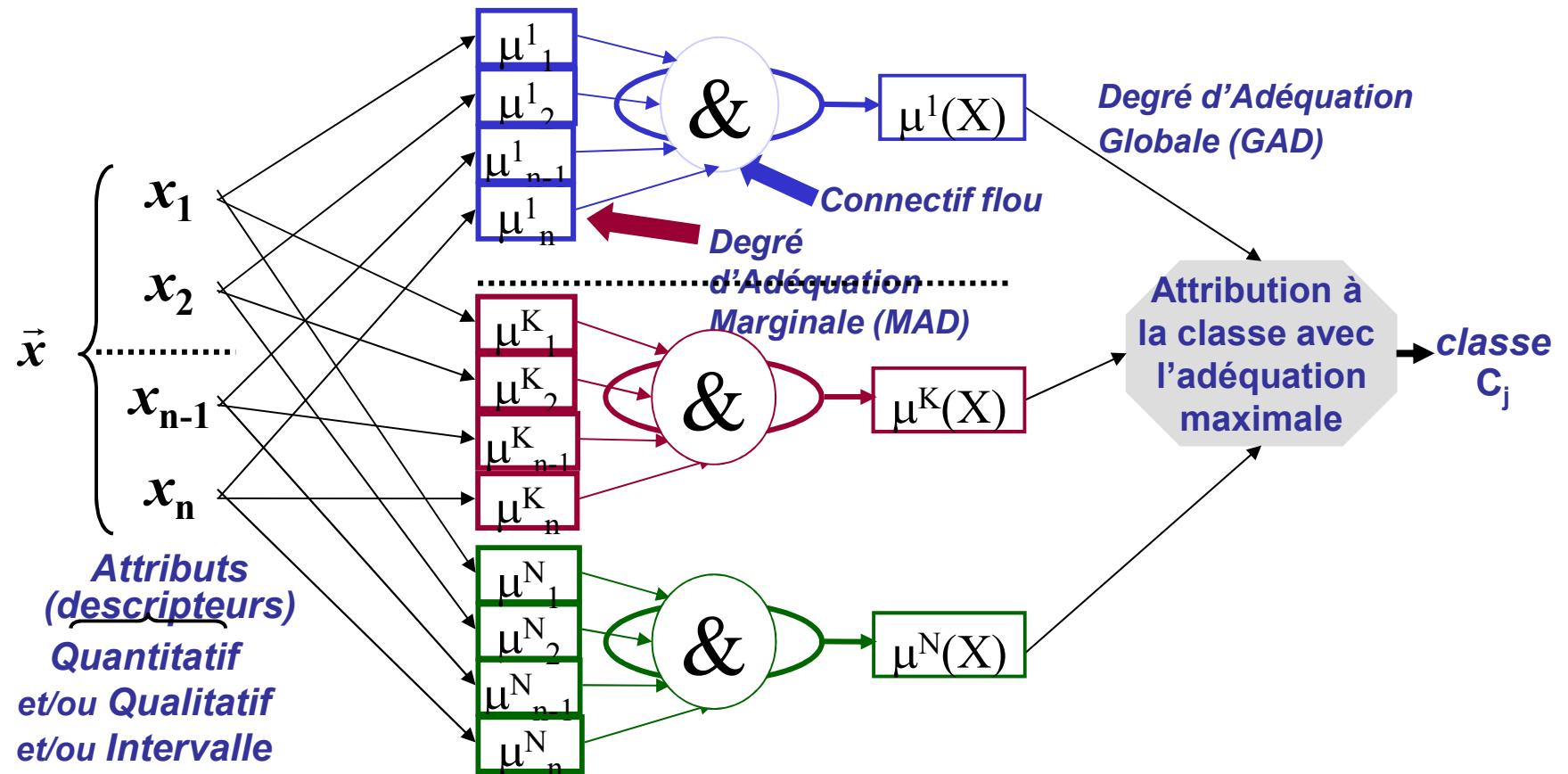


Couplage d'une Méthode par apprentissage et de la Logique Floue

Classification floue par apprentissage:



Couplage d'une Méthode par apprentissage et de la Logique Floue



- **Méthode de classification floue: *LAMDA***
(Learning Algorithm for Multivariable Data Analysis) Aguado et Aguilar , 1994
- **Méthode de classification par apprentissage:**
Capable de calculer l'APPARTENANCE d'un individu (point de mesure) à CHACUNE DES CLASSES
 - Apprentissage supérvisé (classes connues a priori)
 - Apprentissage Non supervisé
- Trois types de donnée traités simultanément:
 - Quantitative
 - Qualitative
 - Interval (nouveau type introduit)

Couplage d'une Méthode par apprentissage et de la Logique Floue

- **Traitement simultané**

Quantitative

Qualitative : *âge, couleur*

Intervalle : mesures des capteurs avec incertitudes, *grade, lymph node status*

- **Apprentissage non supervisé:**

Pas de classe connue a priori

Une seule classe initiale (Non Informative Class) :
auto-organisation

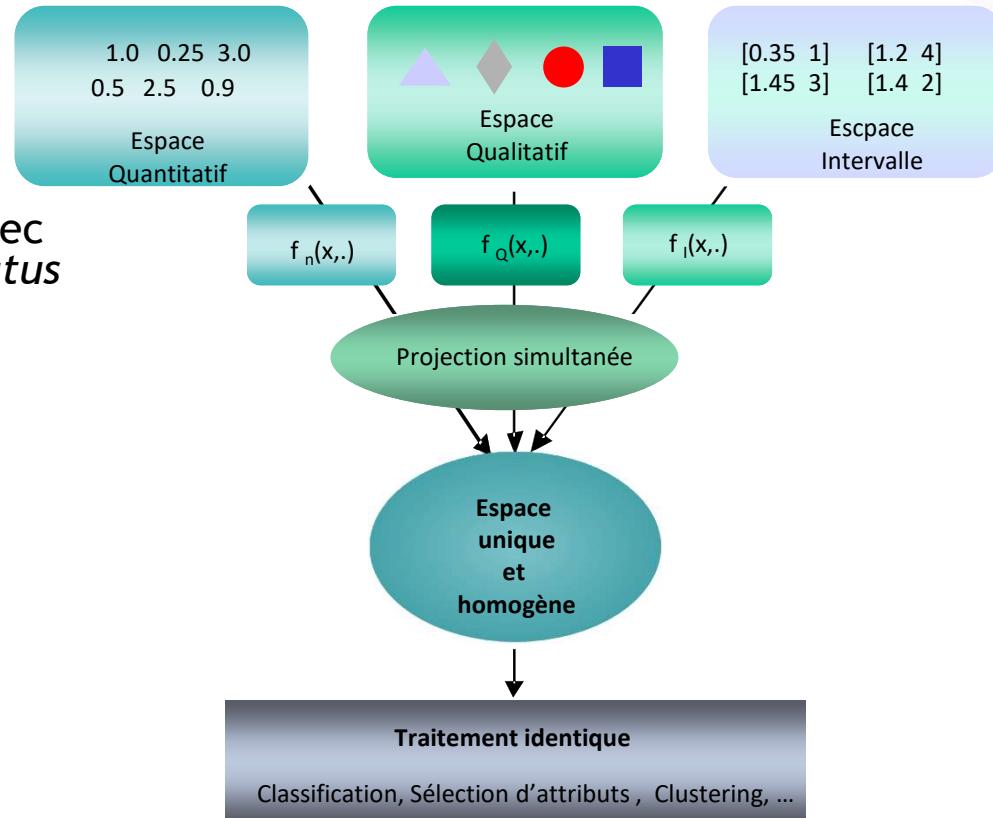
- **Apprentissage supervisé:**

Pré-affectation des classes pour les individus

- **Reconnaissance passive:**

Individus assignés à une des classes prédéfinies

Si **non reconnu** = assigné à la classe NIC



- **Nécessité :**

Introduire la possibilité de traiter un nouveau type de donnée: **intervalle**

Développer une procédure de **sélection d'attribut/capteur intégrée à la procédure de classification**

- **Interêts dans le domaine des procédés**

Le type “**intervalle**” = permet de modéliser les incertitudes de mesure

Procédure intégrée pour la sélection des capteurs

➔ tout type d'information (quantitative, qualitative, intervalle) prise en compte
de manière **simultanée**

- **Optimisation directement des performances de l'apprentissage**

Concept de marges d'appartenance

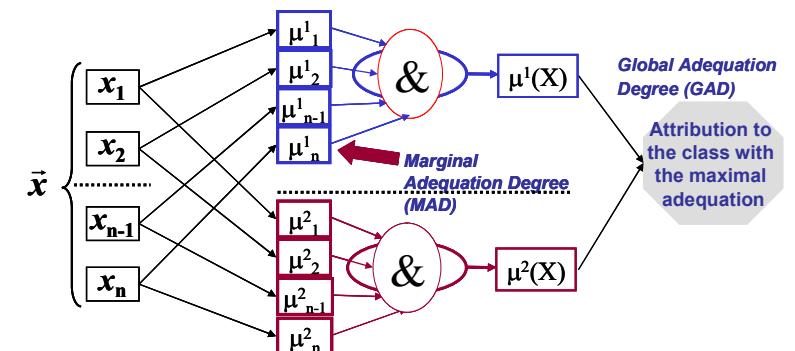
Permet de traiter les corrélations et les redondances entre attributs/capteurs

- MEMBAS (MEmbership Margin Based feAture Selection): [Hedjazi et al., 2010]
Définition d'une Marge d'Appartenance pour chaque $x_n = [x_n^1, x_n^2, \dots, x_n^m]$

$$\beta_n = \Psi(U_{nc}) - \Psi(U_{n\tilde{c}})$$

“bonne classe” “mauvaise classe”

où Ψ = fonction d'agrégation qui calcule la contribution globale de l'ensemble des attributs à l'appartenance à une classe



$$\Psi(U_{n\tilde{c}}) = \sum_{i=1}^d \mu_1^i(x_i) \quad \Psi(U_{nc}) = \sum_{i=1}^d \mu_2^i(x_i)$$

- Element x_n est considéré correctement classé si $\beta_n > 0$.

→ Eviter une recherche heuristique et combinatoire: concept des « poids flous »

$$\Psi(U_{nc_k} / W_f) = W_f^T U_{nc_k} = \sum_i w_{fi} \mu_k^i(x_{ni})$$



- **Procédés**

- Réacteur chimique

- Distillation réactive

- Four

- **Domaine médical**

- Deux principaux challenges**

Haute dimensionnalité des données

Profil génétique = 25 000 données /individu – Diagnostic des procédés : moins de 100 capteurs

Nombre très important d'expressions de gènes non pertinents dans les données des microarrays

Peu d'individus (patients) pour effectuer l'apprentissage- diagnostic des procédés : beaucoup de données disponibles pour l'apprentissage = relevés des capteurs en temps réel (historique sur SCADA) ou résultats de simulation

Hétérogénéité des données (données cliniques : information histopathologiques /analyse des tissus suite à l'ablation chirurgicale de la tumeur + données issues des biopuces)

Quantitative

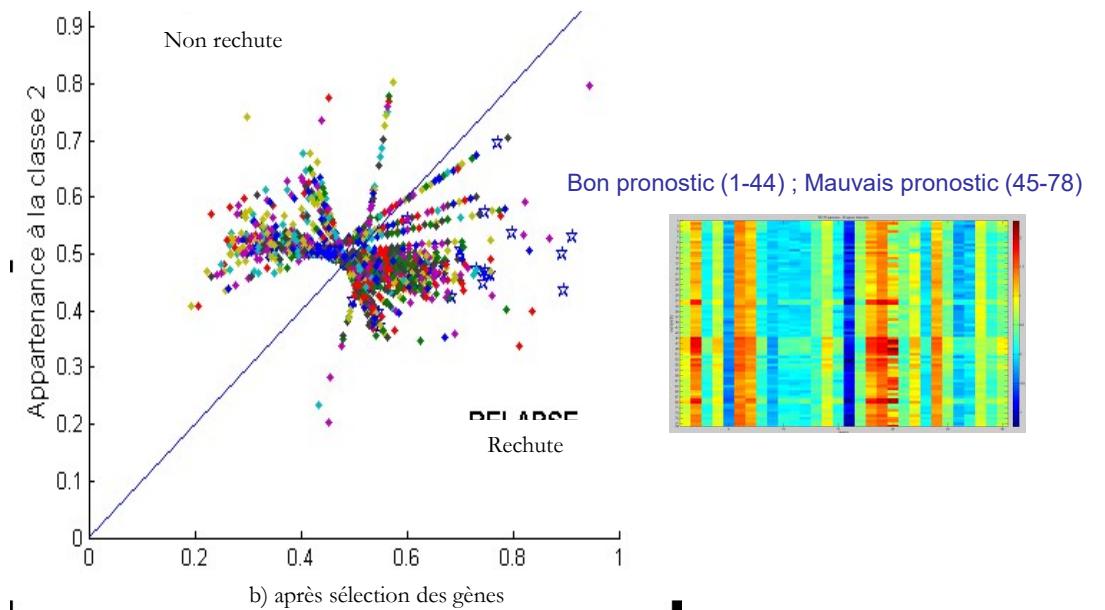
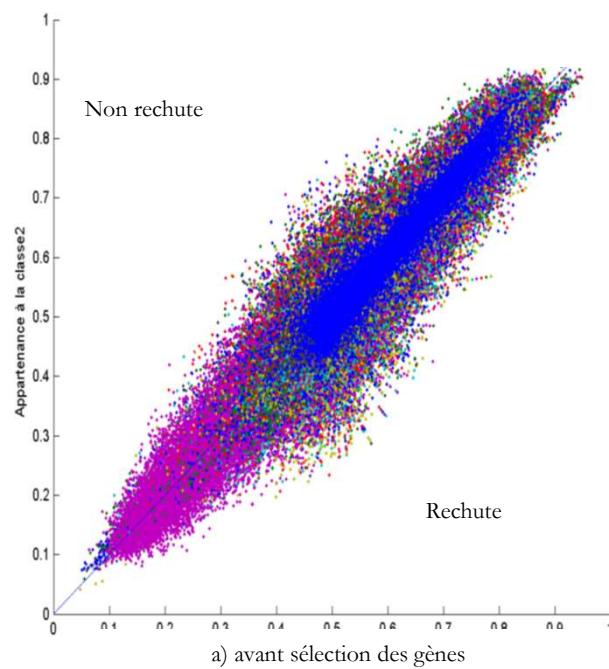
Qualitative : Estrogen Receptor expression (Yes or No)

Interval : Tumor Grade ([3,5]; [6,7]; [8,9])

Application au domaine médical

- Application à l'établissement de "signatures" dans le domaine du cancer diagnostic/pronostic
- Sélection de marqueurs génétiques et/ou cliniques : exemple

Données d'expression de gènes dans l'espace d'appartenance avant et après sélection



Projets INNODIAG/ONCOGRADE



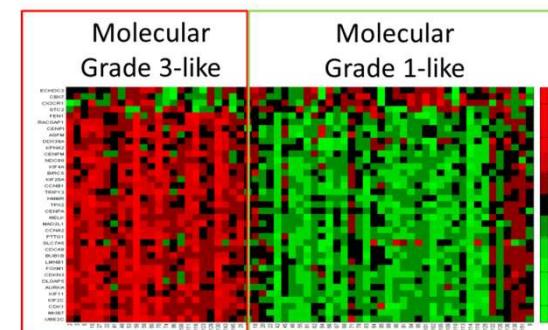
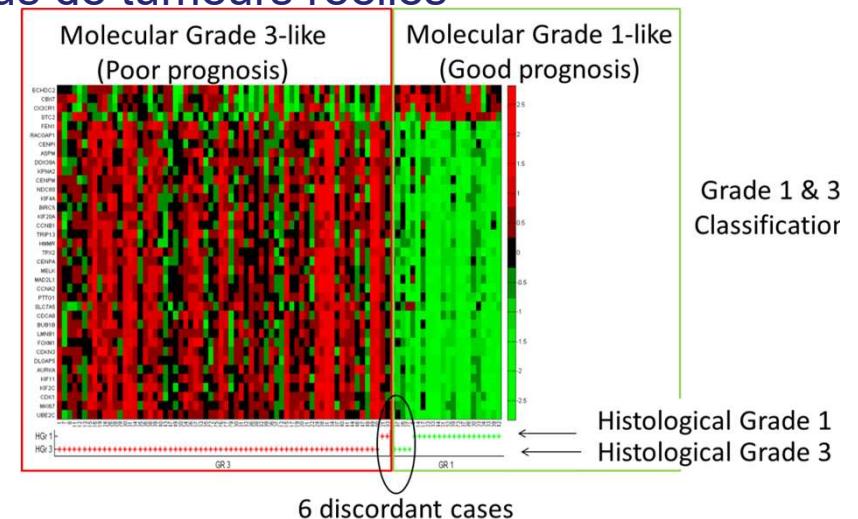
Application de la « liste grade» à des tissus de tumeurs réelles

Objectif : Séparer les « grade 2 »

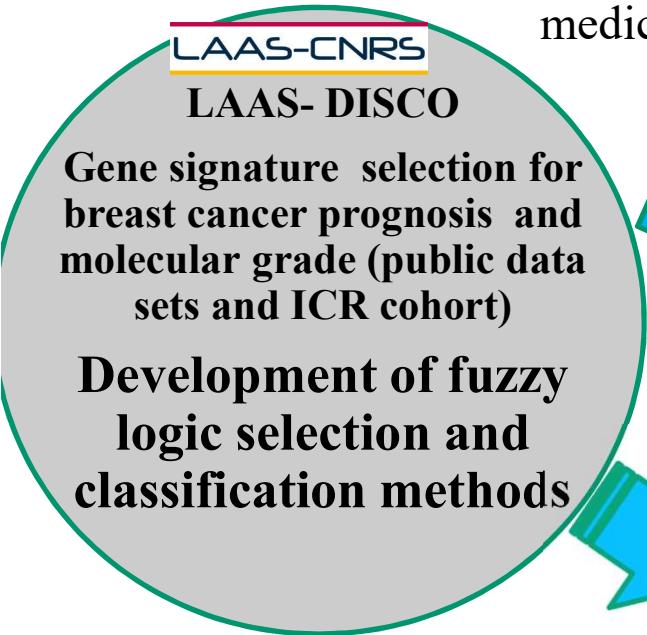
grade1-like grade3-like
(pas-chimio) (chimio)

- 151 tumeurs de cancer du sein congelées
- Cohorte traitée entre 2009 and 2011
- Caractéristiques Clinico-pathologiques semblables à celles utilisées dans des études de routines (pré-ménopausée, node -, ER+, intermediate grade...)

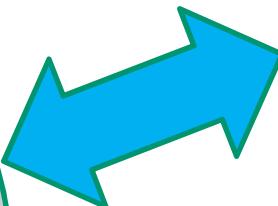
N° LISTE	NB of FINAL GENES	ERROR	SENSITIVITY	SPECIFICITY
36 TK-BD Sotiriou ER+	37	6,82%	94,12%	90%



The **ONCOMATE** and **INNODIAG** projects
the creation of a synergy between different teams



Interactions for medical relevance



33 gene signatures identified by fuzzy logic, including 12 molecular grade signatures

A new project



Claudius Regaud Institute

• Validation cohort (C. Regaud Institute)

- 150 consecutive breast cancers
- RNA extraction from frozen tumour tissue



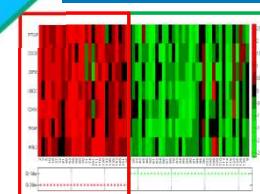
Centre de lutte contre le cancer Toulouse - France

Plateforme Biopuce (INSA)
Design of a Nimblegen in-house microarray for molecular grade signatures test



→ sensitivity 88-100%, specificity 67-97%

Separation of grade 2 into grade 1-like or grade 3-like inducing appropriate treatment



INNOPSYS