Première partie

Classifieur de Bayes naïf

Utilité et risque I

Prendre une décision = faire un acte utile

Lorsqu'un agent intelligent prend une décision c'est pour atteindre un but. La décision la plus appropriée est celle qui est la plus utile pour atteindre le but.

Utilité

On associe une **utilité** $u(h|f)$ à toute décision associant l'hypothèse h à l'événement e alors que l'hypothèse correcte était f.
L'utilité moyenne ou bayésienne associée à la classification de l'événement e est alors donnée par :

$$U(h|e) = \sum_f u(h|f)P(f|e)$$

Utilité et risque II

Prendre une décision = prendre un risque

Toute prise de décision dont les conséquences sont inconnues constituent une prise de risque. La décision la plus appropriée est souvent considérée comme la moins risquée.

Risque

On associe un **coût** $c(h|f)$ à toute décision associant l'hypothèse h à l'événement e alors que l'hypothèse correcte était f.
Le risque moyen ou bayésien associé à la classification de l'événement e est alors donné par :

$$R(h|e) = \sum_f c(h|f)P(f|e)$$

Utilité et risque III

Décision

La décision optimale est donc donnée par :

$$h^* = \underset{h}{\text{argmax}}\, U(h|e) \; si \max_h U(h|e) > \sigma_U$$

$$h^* = \underset{h}{\text{argmin}}\, R(h|e) \; si \min_h R(h|e) < \sigma_R$$

Sinon : Rejet !

rem : on définit un seuil d'utilité σ_U ou de risque σ_R auquel est associé une décision de rejet.

Utilité et risque IV

Utilisation de la règle de Bayes

$$h^* = \underset{h}{\operatorname{argmax}}\, U(h|e) = \underset{h}{\operatorname{argmax}} \sum_f u(h|f)\frac{P(e|f)P(f)}{P(e)}$$

$$h^* = \underset{h}{\operatorname{argmax}}\, U(h|e) = \underset{h}{\operatorname{argmax}} \sum_f u(h|f)P(e|f)P(f)$$

Théorie de la décision

L'introduction de la notion d'utilité (ou de risque) donne lieu à la **théorie de l'utilité**. L'association de cette théorie et de celle des probabilités donne lieu à la **théorie de la décision**.

$$\text{Decision theory} = \text{Probability theory} + \text{Utility theory}$$

Règle du Maximum A Posteriori : MAP

Utilité identique et nulle pour toutes les hypothèses erronées.
Soit $u(h|f) = 0$ si $h \neq f$ et $u(h|f) = 1$ sinon.

$$h^* = \underset{h}{\text{argmax}}\, U(h|e) = \underset{h}{\text{argmax}} \sum_h u(h|f)P(e|f)P(f)$$

Règle du Maximum a Posteriori - MAP

$$h^* = \underset{h}{\text{argmax}}\, P(e|h)P(h) = \underset{h}{\text{argmax}}\, P(h|e)$$

On choisit donc l'hypothèse qui a la probabilité *a posteriori* maximale

Remarque

- *Cela fonctionne de la même manière avec le risque*
- *L'hypothèse reste forte (même coût pour toutes les erreurs, faux en diagnostic médical) mais est souvent utilisée.*

Maximum de vraisemblance ou ML

Si les hypothèses ont toutes la même probabilité *a priori* : $P(h_i) = P(h) \forall h_i$.

Règle du Maximum de Vraisemblance - ML

$$h^* = \underset{h}{\text{argmax}}\, P(e|h)$$

- On choisit l'hypothèse pour laquelle la vraisemblance de l'événement est maximale.
- in English : Maximum Likelihood (**ML**)
- Rem : on remplace tout simplement $P(h|e)$ par $P(e|h)$!
- ex : reconnaissance de chiffres écrits

Classificateur naïf de Bayes I

Souvent, on dispose de plusieurs observations en même temps.
On cherche alors $P(h|e_1, e_2, ..., e_n)$

Règle de Bayes : $P(h|e_1, e_2, ..., e_n) = \frac{P(e_1,e_2,...,e_n|h)P(h)}{P(e_1,e_2,...,e_n)}$

Si les événements e_i sont conditionnellement indépendants sachant l'hypothèse h on : $P(e_1, e_2, ..., e_n|h) = \prod_i P(e_i|h)$

Classificateur naïf de Bayes

$$h^* = \underset{h}{\text{argmax}}\, P(h) \prod_i P(e_i|h)$$

Classificateur naïf de Bayes II

ex : diagnostic de la grippe

Symptômes : fièvre, mal de tête, douleur à la gorge.

$$P(grippe|fievre, tete, gorge) = \frac{P(fievre, tete, gorge|grippe)P(grippe)}{P(fievre, tete, gorge)}$$

$$= \frac{P(fievre|grippe)P(tete|grippe)P(gorge|grippe)P(grippe)}{P(fievre, tete, gorge)}$$

Ces valeurs sont beaucoup plus simples à mesurer, mais il faut faire l'hypothèse que la fièvre n'est pas la cause du mal de tête. C'est la raison pour laquelle il est appelé *classificateur naïf*.

Deuxième partie

Réseaux de neurones

Historique I

Pourquoi ?

L'être humain réalise les tâches que nous cherchons à reproduire. Donc, pourquoi ne pas s'inspirer de la machinerie qui lui sert à réaliser ces tâches : le cerveau !

Connexionnisme

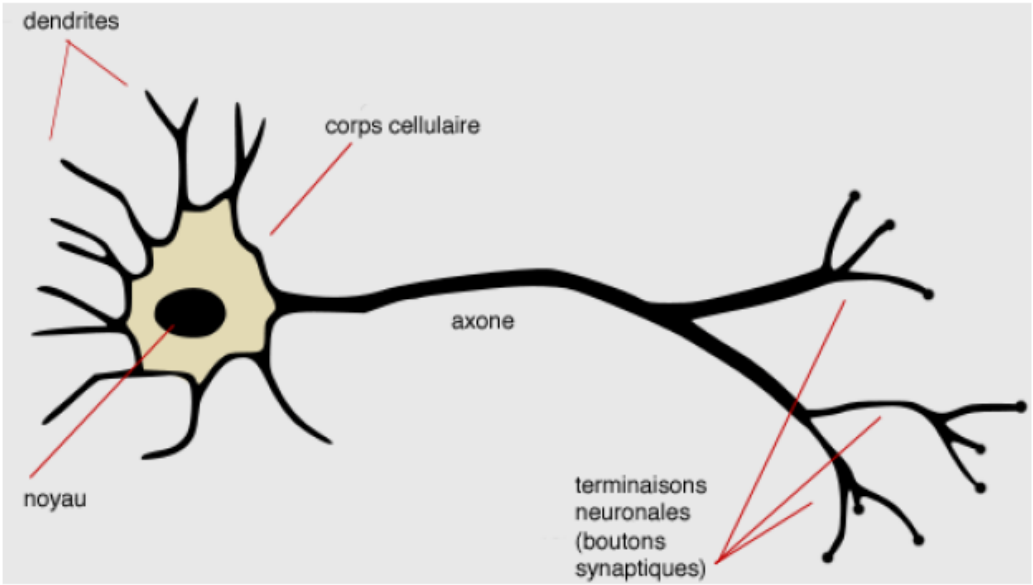
On modélise le cerveau comme un ensemble d'unités de calculs (les neurones) connectées entre elles. On apprend les paramètres du réseau (les poids des connexions) pour reproduire le même comportement.

Historique II

Histoire

- Premier neurone formel par McCulloch et Pitts en 1943
- Etude systématique des neurones par Hebb en 1949 (loi de Hebb : deux neurones activés en même temps voient leur connexion renforcée, sinon inhibition)
- Perceptron par Rosenblatt en 1958 et première méthode d'apprentissage
- Travaux de Widrow et Hoff à la même époque (apprentissage en ligne)

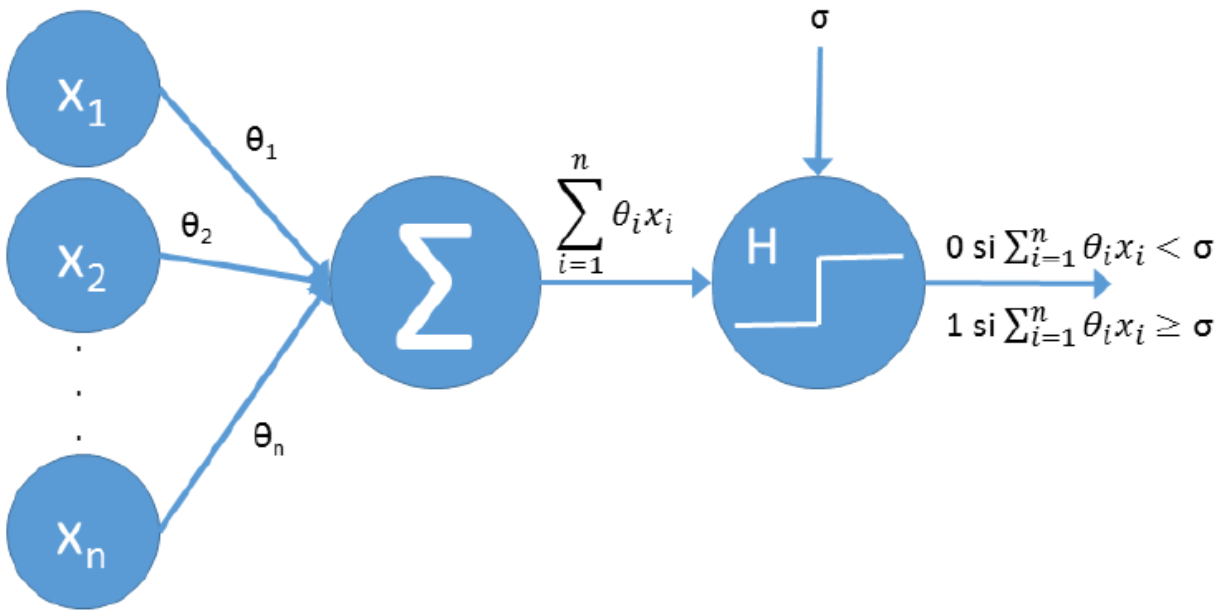
Neurone biologique



Modèle

L'information se transmet le long de l'axone sous la forme d'impulsions électriques. L'activation répond à un effet de seuil et dépends de l'information (neurotransmetteurs) reçue via les synapses des autres neurones. Les neurotransmetteurs peuvent avoir des effets excitateurs ou inhibiteurs.

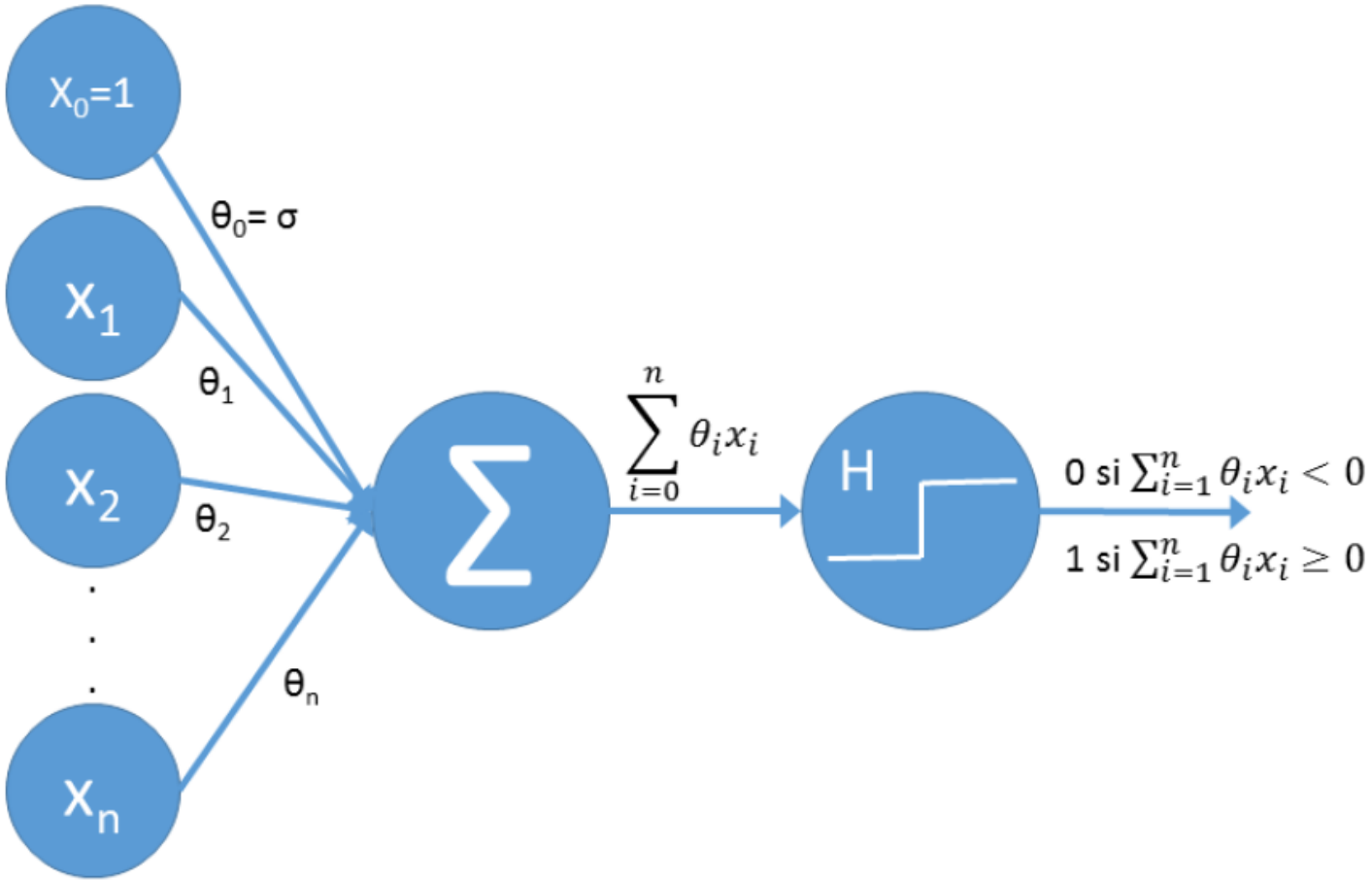
Neurone formel ou perceptron I



Nomenclature

- θ_i = poids synaptiques σ = biais
- H = fonction d'activation (ici Heaviside)

Neurone formel ou perceptron II



Apprentissage des poids I

Règle de Hebb

$$\theta_j^{(i+1)} = \theta_j^{(i)} + \alpha_i x_j^{(i)} y^{(i)}$$

- Inspiration biologique
- En général, données binaires dans {-1,1}
- Deux neurones qui s'activent en même temps voient leur connexion synaptique renforcée
- Deux neurones qui s'activent en sens opposés voient leur connexion synaptique inhibée
- N'apprend pas de l'erreur ...

Apprentissage des poids II

Correction d'erreur

$$\theta_j^{(i+1)} = \theta_j^{(i)} + \overbrace{\left(y^{(i)} - H\left(\sum_j \theta_j^{(i)} x_j^{(i)} \right) \right)}^{Erreur} x_j^{(i)}$$

- Ici les sorties sont dans $\{0,1\}$.
- Si on prend une décision correcte, on ne change pas les poids
- Si on se trompe, on ajoute ou on retire l'entrée.
- On apprend des erreurs.

Ca ne vous rappelle rien ? I

Régression

$$\theta_j^{(i+1)} = \theta_j^{(i)} + \alpha_i \left(y^{(i)} - \overbrace{\not{H} \sum_j \theta_j^{(i)} x_j^{(i)}}^{\text{Potentiel Présynaptique}} \right) x_j^{(i)}$$

- Règle de Widrow-Hoff !
- On ajoute un pas d'apprentissage.
- On conserve le seuil mais après apprentissage (donc, pas régression).

Ca ne vous rappelle rien ? II

Régression Logistique

$$\theta_j^{(i+1)} = \theta_j^{(i)} + \alpha_i \left(y^{(i)} - \overbrace{\sigma\left(\sum_j \theta_j^{(i)} x_j^{(i)} \right)}^{\text{Activation Sigmoïde}} \right) x_j^{(i)}$$

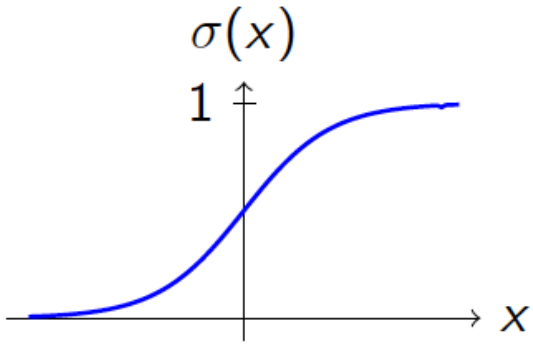
- On remplace H par σ
- idée : on peut construire des perceptrons avec plusieurs types de fonction d'activation

Vous faisiez donc des réseaux de neurones sans le savoir !

Différentes fonctions d'activation standards

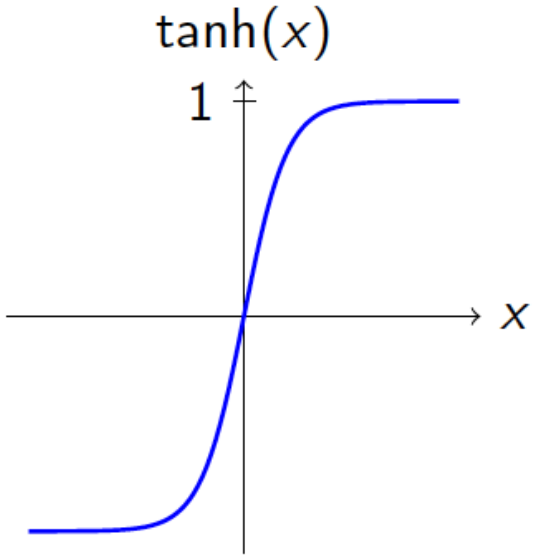
Sigmoïde

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

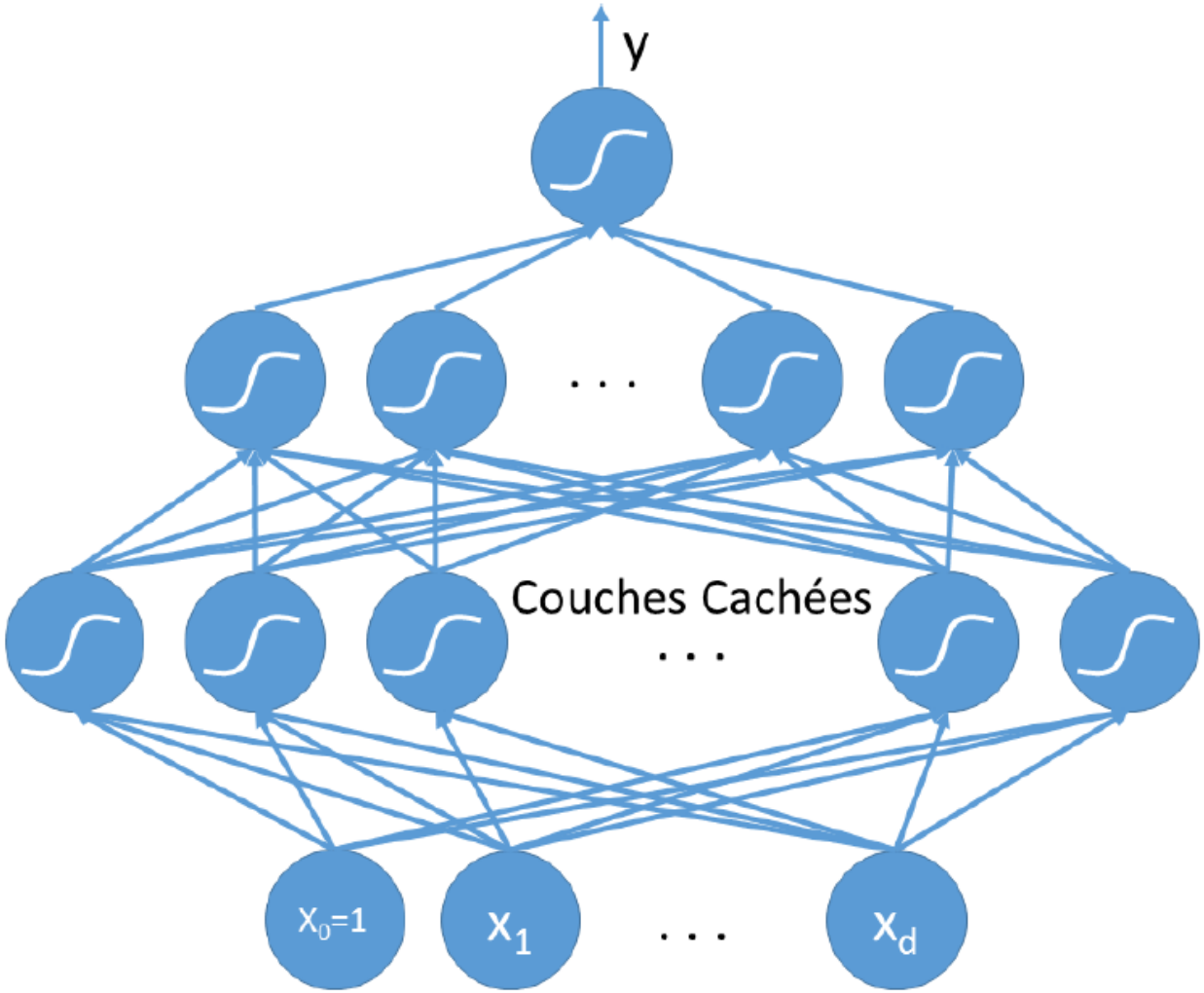


Tangente Hyperbolique

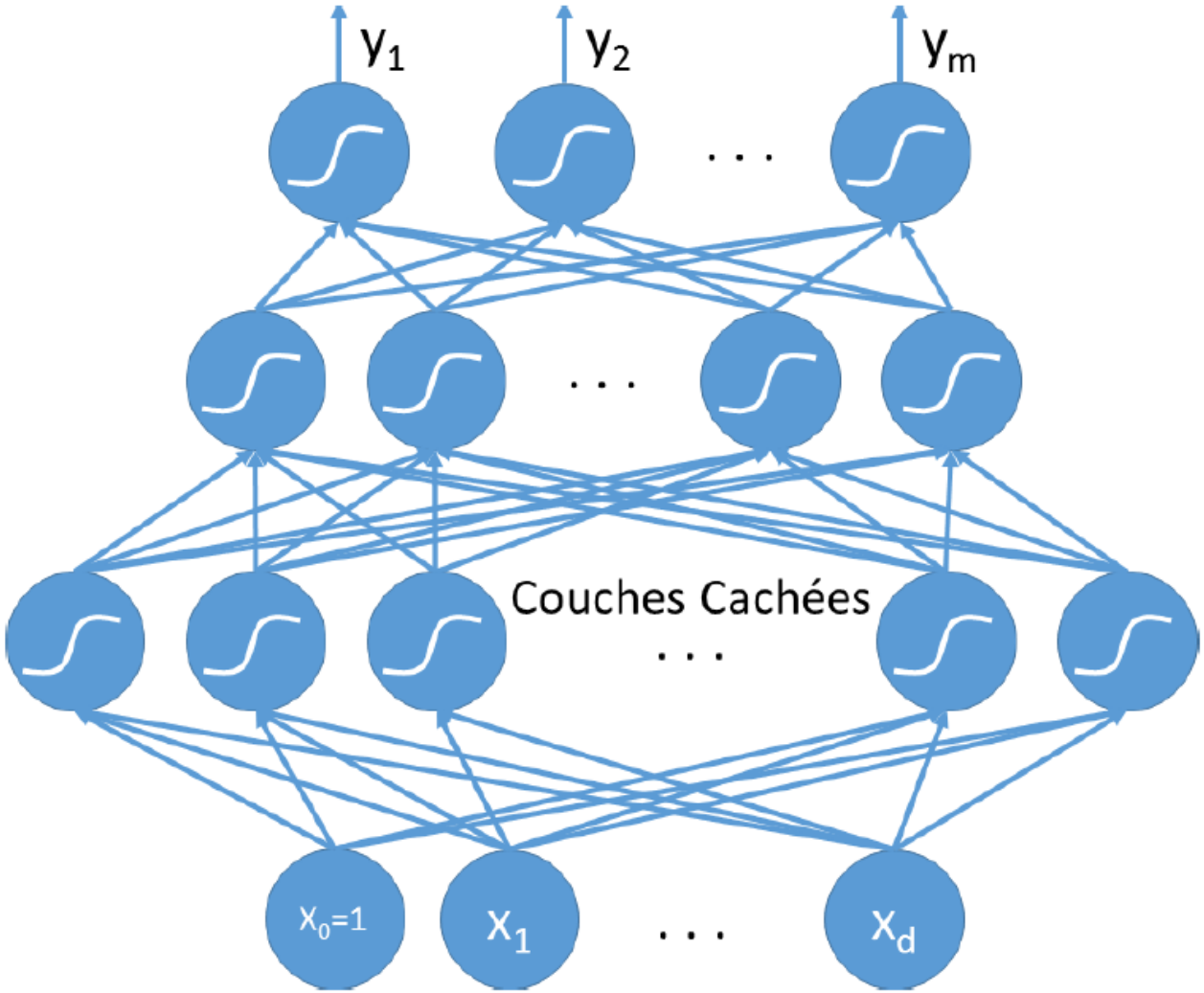
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



Perceptron multicouche (MLP)



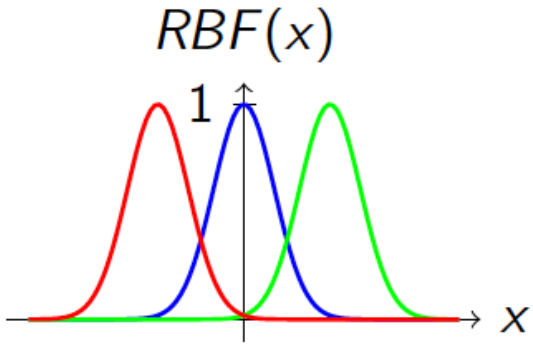
Multiclasse

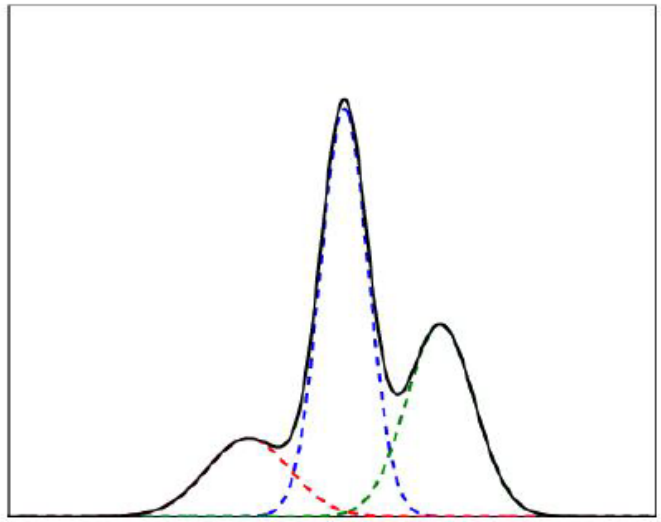


Régression I

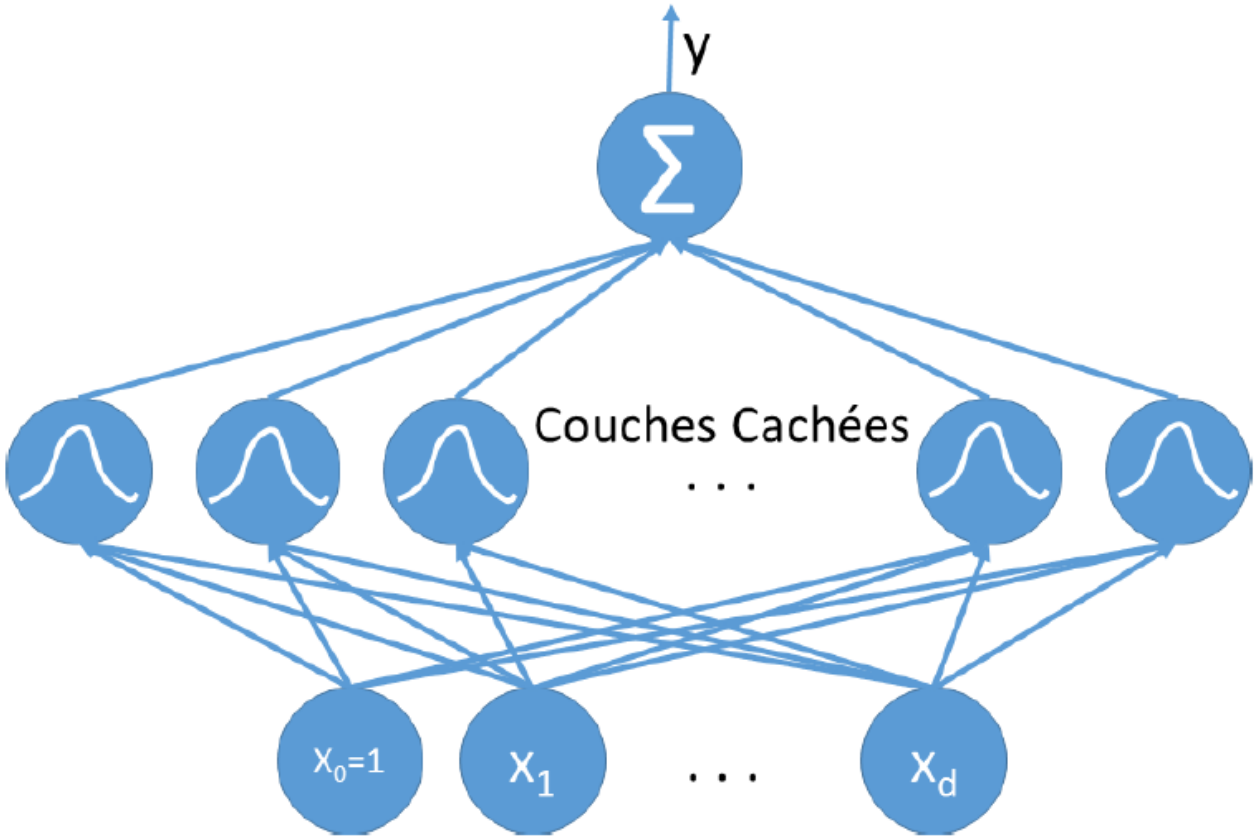
Radial Basis Function

$$RBF(x) = e^{-\frac{(x-\mu)^2}{\sigma}}$$





Régression II



Rétropropagation du gradient I

Problème

$\nabla_\theta R_n(f_\theta(x))$ n'est plus calculable facilement car chaque poids d'une couche influe sur les couches supérieures. Et il y a des fonctions non-linéaires qui séparent les couches.

Idée

- On indice par k les valeurs de la couche k (x_k, θ_k).
- Erreur pour l'exemple i : $E = \frac{1}{m}\sum_{j=1}^{m}(y_j^{(i)} - f(\theta_{j,k}^T x_k^{(i)}))^2$
- On cherche comment corriger les poids $\theta_{j,k}$ pour minimiser l'erreur. On cherche $\frac{\partial E}{\partial \theta_{j,k}} = 0$
- On pose $\nu_k = \theta_{j,k}^T x_k$ et $\delta_k = \frac{\partial E}{\partial \nu_k}$.
- En omettant i, on a $\frac{\partial E}{\partial \theta_{j,k}} = \frac{\partial E}{\partial \nu_k}\frac{\partial \nu_k}{\theta_{j,k}} = \delta_k x_k$

Rétropropagation du gradient II

Généralisation de la descente de gradient stochastique

- Pour la sortie :
  $\delta_k = \frac{\partial E}{\partial \nu_k} = \frac{\partial}{\partial \nu_k}(y_j - f(\nu_k))^2 = f'(\nu_k)(y_l - f(\nu_k))$
- Pour les couches cachées on a : $\delta_k = f'(\nu_k) \sum_j \theta_{j,k+1} \delta_{k+1}$

Algorithme

- Initialiser les poids aléatoirement
- Propager un exemple $x^{(i)}$ dans le réseau
- Calculer la sortie $y^{(i)}$
- Mettre à jour : $\theta_{j,k}^{(i+1)} = \theta_{j,k}^{(i)} + \alpha_i \delta_k x_j^{(i)}$

Rétropropagation du gradient III

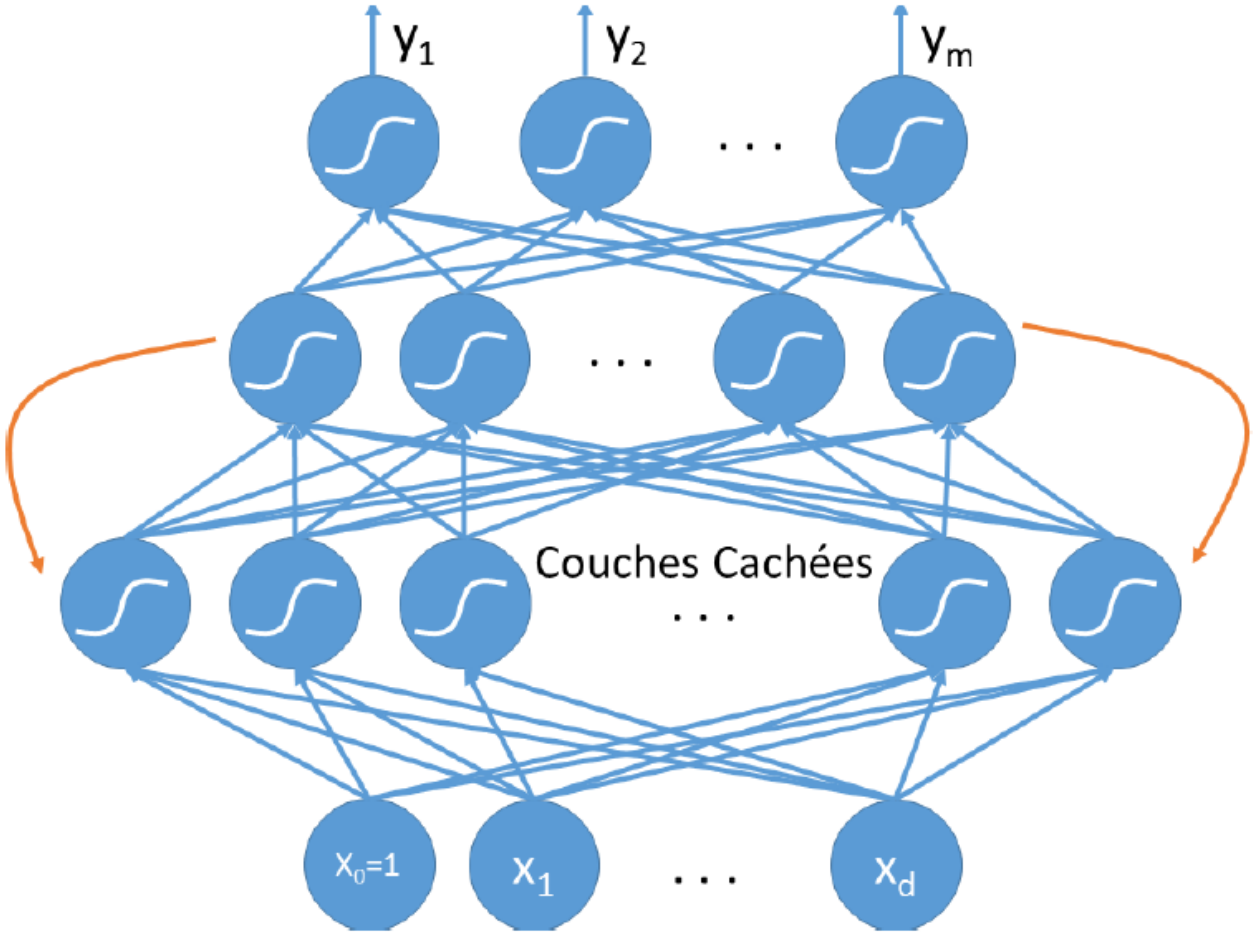
Avantages

- Approximateur universel
- Représentation compacte d'une fonction

Problème

- Minimum locaux
- Sur-apprentissage
- Accumulation de l'erreur
- Limitation du nombre de couches
- Difficile de fixer le nombre de couches, le nombre de neurones dans les couches cachées

Réseaux Récurrents (RNN)



Réseaux profonds (DNN)