Réseaux de neurones

Jean-Michel Marin

Université de Montpellier
Institut Montpelliérain Alexander Grothendieck (IMAG)

HMMA303

Introduction

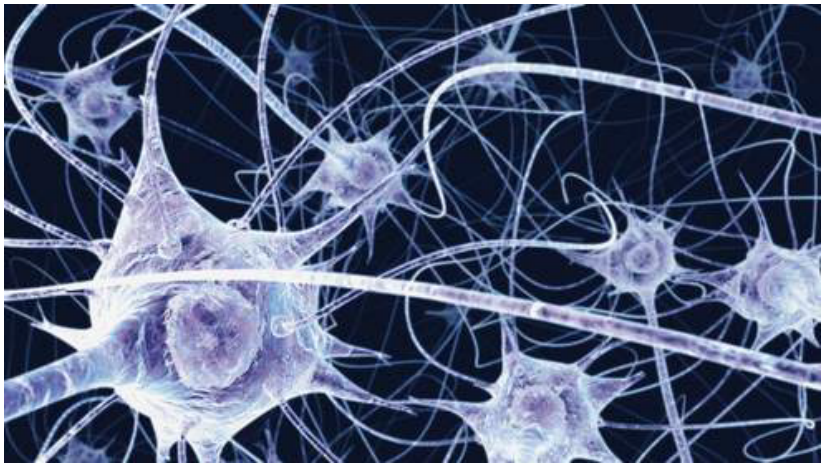
Combiner de nombreuses fonctions élémentaires pour former des fonctions complexes

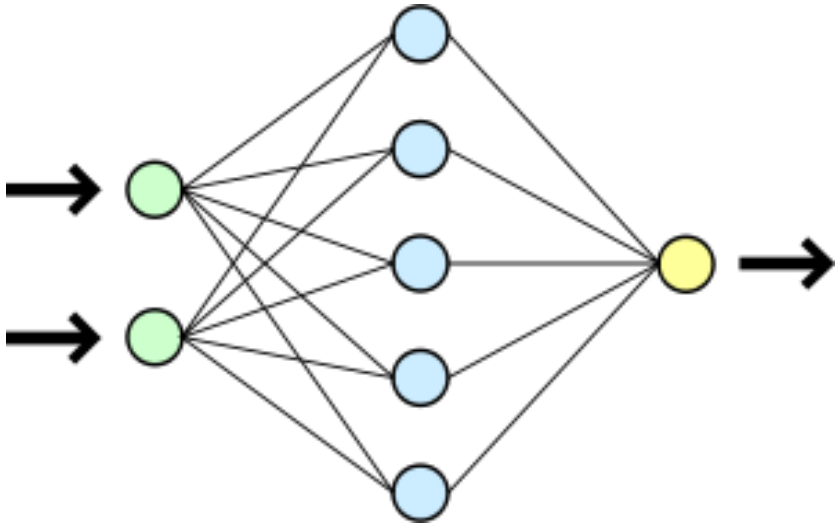
Apprendre les liens entre ces fonctions simples
à l'aide de l'échantillon d'apprentissage

Analogie (un peu commerciale) avec le cerveau

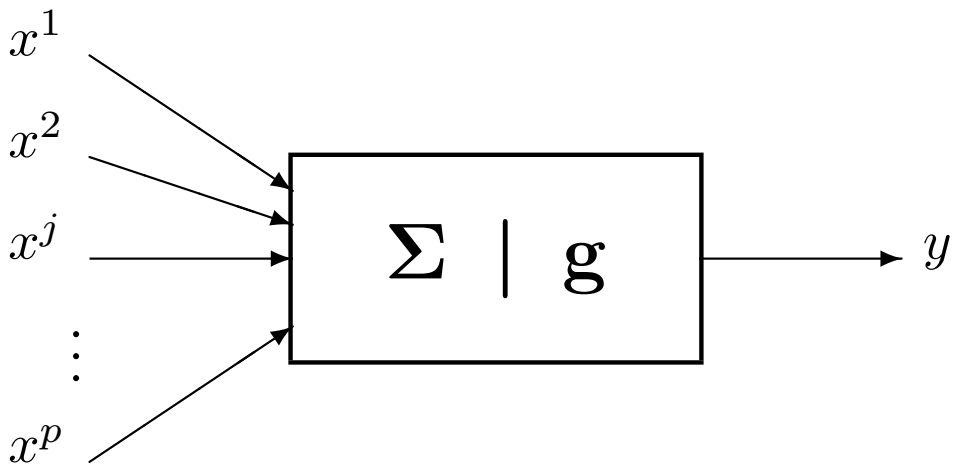
- fonctions élémentaires = neurones
- connexions = synapses
- apprentissage des connexions = la connaissance

Introduction





Introduction
Neurone formel



Introduction
Neurone formel

Le neurone formel est un modèle qui se caractérise par un état interne s, des signaux d'entrée $x^1, \ldots, x^p$ et une fonction d'activation

$$s = h(x^1, \ldots, x^p) = g\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x^j\right)$$

Introduction
Fonction activation

La fonction d'activation opère une transformation d'une combinaison affine des signaux d'entrée, α_0, terme constant, étant appelé le biais du neurone

Cette combinaison affine est déterminée par un vecteur de poids $[\alpha_0, \ldots, \alpha_p]$ associé à chaque neurone et dont les valeurs sont estimées dans la phase d'apprentissage

Ils constituent la mémoire ou connaissance répartie du réseau

Introduction
Fonction activation

- linéaire $g(x) = x$
- seuil $g(x) = \mathbf{1}_{[0,+\infty[}(x)$
- sigmoïde $g(x) = 1/(1 + exp(-x))$
- ReLU (Rectified Linear Unit) $g(x) = \max(0, x)$
- softmax $g(x)_{(j)} = \frac{\exp(x^j)}{\sum_{k=1}^{K} \exp(x^k)}$

Introduction
Fonction activation

Les modèles linéaires, sigmoïdaux, ReLU, softmax sont bien adaptés aux algorithmes d'apprentissage impliquant une rétro-propagation du gradient car leur fonction d'activation est différentiable ; ce sont les plus utilisés

Le modèle à seuil est sans doute plus conforme à la réalité biologique mais pose des problèmes d'apprentissage

Le modèle stochastique est utilisé pour des problèmes d'optimisation globale de fonctions perturbées

Perceptron multicouche
Architecture

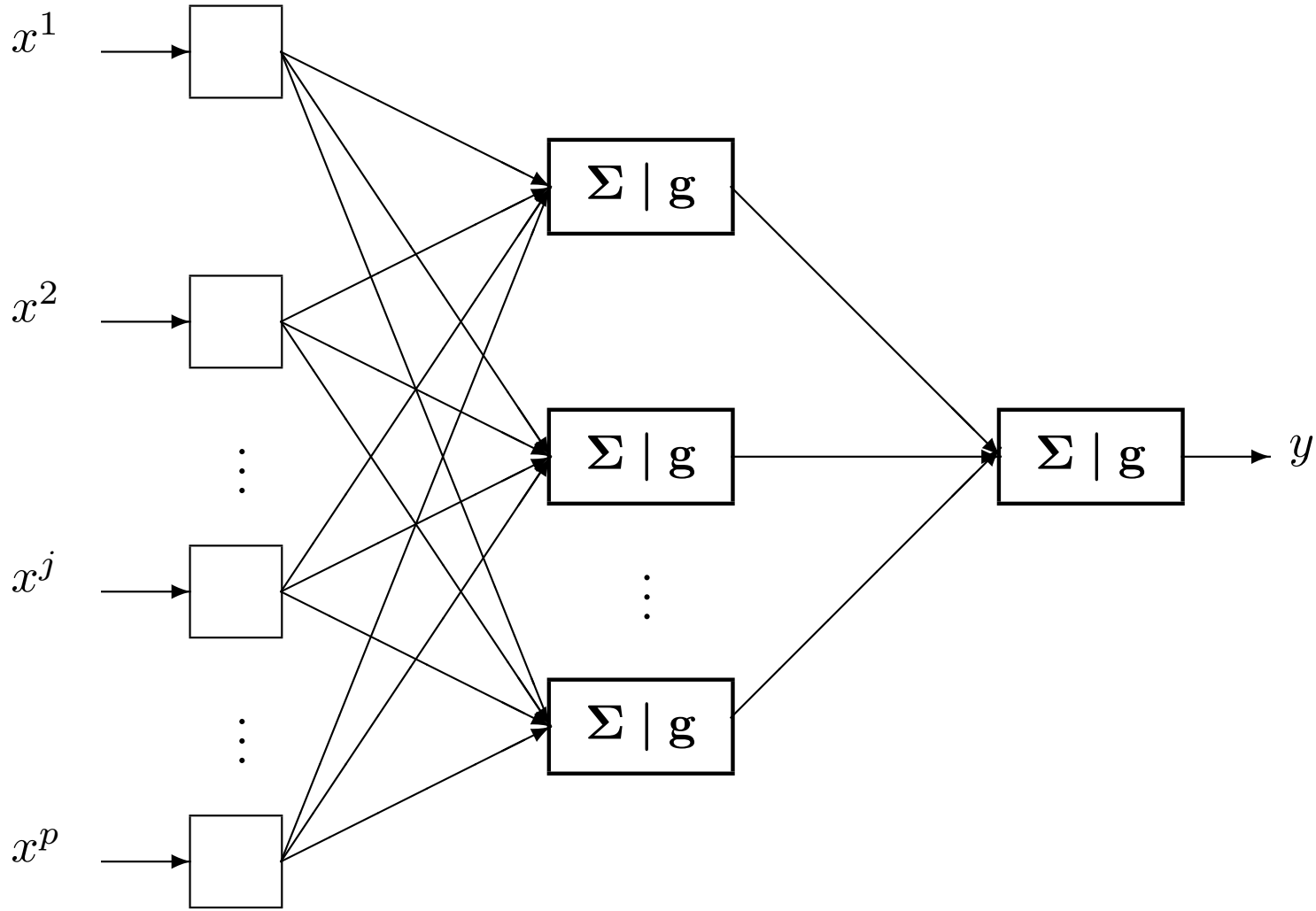
Le perceptron multicouche (PMC) est un réseau composé de couches successives

Une couche est un ensemble de neurones n'ayant pas de connexion entre eux

Une couche d'entrée lit les signaux entrant, un neurone par entrée x_j, une couche en sortie fournit la réponse du système

Dans un perceptron, un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante

Perceptron multicouche
Architecture



Perceptron multicouche
Fonction de transfert

Les entrées du réseau sont les variables explicatives $x^1, \ldots, x^p$

Les poids sont des paramètres α à estimer lors de la procédure d'apprentissage

la sortie est la variable y à expliquer

Perceptron multicouche
Fonction de transfert

Un théorème dit d'approximation universelle montre qu'une structure élémentaire à une seule couche cachée est suffisante

Toute fonction régulière peut être approchée uniformément avec une précision arbitraire et dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini possédant tous la même fonction d'activation et un neurone de sortie linéaire

Attention, ce résultat, qui semble contradictoire avec les structures d'apprentissage profond, est théorique, il masque des difficultés d'apprentissage et de stabilité pour des problèmes complexes en très grande dimension

Perceptron multicouche
Fonction de transfert

En régression la dernière couche est constituée d'un seul neurone muni de la fonction d'activation identité

En classification binaire, le neurone de sortie est muni de la fonction d'activation sigmoïde

En classification à m classes, le neurone de sortie intègre une fonction d'activation softmax à m valeurs dont la somme est égale à 1, ces m valeurs sont assimilables à des probabilités d'appartenance aux classes

Perceptron multicouche
Apprentissage

Considérons le cas simple de la régression avec un réseau constitué d'un neurone de sortie linéaire et d'une couche cachée à q neurones

$$f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{k=1}^{q} \beta_k g\left(\sum_{j=1}^{p} \alpha_{kj} x^j\right)$$

Les termes de biais sont omis

Perceptron multicouche
Apprentissage

On dispose d'une base d'apprentissage de taille n, les paramètres sont optimisés par moindres carrés

Estimation des paramètres α et β par minimisation de

$$Q(\alpha, \beta) = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} (y_i - f(x_i; \alpha, \beta))^2$$

Différents algorithmes d'optimisation sont proposés,
ils sont généralement basés sur une évaluation du gradient par rétro-propagation

Perceptron multicouche
Apprentissage

Rétro-propagation de l'erreur

Il s'agit donc dévaluer la dérivée de la fonction coût en une observation et par rapport aux différents paramètres

On note

$$\widehat{y}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}_i; \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Perceptron multicouche
Apprentissage

Nous avons

$$\frac{\delta Q_i}{\delta \beta_k}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -2\left[y_i - \widehat{y}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})\right] g\left(\sum_{j=1}^{p} \alpha_{kj} x_i^j\right)$$

$$\frac{\delta Q_i}{\delta \alpha_{kj}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -2\left[y_i - \widehat{y}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})\right] \beta_k g'\left(\sum_{j=1}^{p} \alpha_{kj} x_i^j\right) x_i^j$$

Perceptron multicouche
Apprentissage

Une passe avant dans le réseau, en partant de l'entrée $\mathbf{x_i}$ permet de calculer les valeurs à chaque neurone et au final $\widehat{\mathbf{y}}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$

Tirant profit de l'évaluation des termes à chaque neurone lors de la passe avant la passe retour permet de calculer tous les gradients

Perceptron multicouche
Apprentissage

Sachant évaluer les gradients, différents algorithmes, plus ou moins sophistiqués, sont implémentés.

Le plus élémentaire est une utilisation itérative du gradient : en tout point de l'espace des paramètres, le vecteur gradient de Q pointe dans la direction de l'erreur croissante.

Pour faire décroître Q, il suffit donc de se déplacer en sens contraire

Perceptron multicouche
Apprentissage

Il s'agit d'un algorithme itératif modifiant les poids de chaque neurone selon

$$\beta_k^{(t+1)} = \beta_k^{(t)} - \tau \sum_{i=1}^{n} \frac{\delta Q_i}{\delta \beta_k}(\boldsymbol{\alpha}^{(t)}, \boldsymbol{\beta}^{(t)})$$

$$\alpha_{kj}^{(t+1)} = \alpha_{kj}^{(t)} - \tau \sum_{i=1}^{n} \frac{\delta Q_i}{\delta \alpha_{kj}}(\boldsymbol{\alpha}^{(t)}, \boldsymbol{\beta}^{(t)})$$

Perceptron multicouche
Apprentissage

Le coefficient de proportionnalité τ est appelé le taux d'apprentissage

Il peut être fixe, à déterminer par l'utilisateur, ou encore varier en cours d'exécution selon certaines heuristiques

Il paraît en effet intuitivement raisonnable que, grand au début pour aller plus vite, ce taux décroisse pour aboutir à un réglage plus fin au fur et à mesure que le système s'approche d'une solution

Perceptron multicouche
Apprentissage

Une version accélérée de l'algorithme fait intervenir à chaque itération un ensemble (batch) d'observations pour moyenner les gradients et mises à jour des poids

Bien d'autres méthodes d'optimisation ont été adaptées à l'apprentissage d'un réseau : méthodes du gradient avec second ordre utilisant une approximation itérative de la matrice hessienne

La littérature sur le sujet propose quantités de recettes destinées à améliorer la vitesse de convergence de l'algorithme ou bien lui éviter de rester collé à une solution locale défavorable.

Contrôle de la complexité
Régularisation

Dans les réseaux élémentaires, une option simple pour éviter le sur-apprentissage consiste à introduire une terme de pénalisation dans le critère à optimiser :

$$Q(\theta) + \gamma ||\theta||_1$$

Plus la valeur de γ est importante et moins les poids des entrées des neurones peuvent prendre des valeurs chaotiques contribuant ainsi à limiter les risques de sur-apprentissage.

Contrôle de la complexité
Choix des paramètres

L'utilisateur doit donc déterminer

- les variables d'entrée et la variable de sortie ; transformations, normalisations ;
- l'architecture du réseau :
  1. le nombre de couches cachées qui correspond à une aptitude à traiter des problèmes de non-linéarité,
  2. le nombre de neurones par couche cachée.

  Ces deux choix conditionnent directement le nombre de paramètres (de poids) à estimer et donc la complexité du modèle.

Contrôle de la complexité
Choix des paramètres

- 3 autres paramètres interviennent également sur ce compromis : le nombre maximum d'itérations, l'erreur maximum tolérée et un terme éventuel de régularisation lasso
- le taux d'apprentissage ainsi qu'une éventuelle stratégie d'évolution de celui-ci
- la taille des ensembles ou batchs d'observations considérés à chaque itération.

En pratique, tous ces paramètres ne peuvent être réglés simultanément par l'utilisateur

Introduction à l'apprentissage profond

Années 90, début des années 2000 : le développement de l'apprentissage machine s'est focalisé sur les SVM et l'agrégation

Renouveau de la recherche sur les neural nets impulsé par

- Geoffrey Hinton (google depuis 2013)
- Yoshua Bengio (Element IA)
- Yan le Cun

Yan Le Cun a tenu à jour un célèbre site dédié à la reconnaissance des caractères manuscrits de la base MNIST

`http://yann.lecun.com/exdb/mnist/`

Introduction à l'apprentissage profond

La liste des publications listées sur ce site témoigne de la lente progression de la qualité de reconnaissance, de 12% avec un simple perceptron à 1 couche jusqu'à moins de 0,3% en 2012 par l'introduction et l'amélioration incrémentale d'une couche de neurones spécifique appelée convulational neural network (Conv- Net).

L'étude de ces données qui ont servi de benchmark pour la comparaison de très nombreuses méthodes sert maintenant de données jouet pour beaucoup de tutoriels des environnements dédiés (tensorFlow, Keras, pyTorch, caffe...)

Introduction à l'apprentissage profond

Trois grandes familles de réseaux d'apprentissage profond sont développées avec des ambitions industrielles en profitant du développement des cartes graphiques (GPU) pour paralléliser massivement les calculs

Introduction à l'apprentissage profond

Trois grandes familles de réseaux d'apprentissage profond sont développées avec des ambitions industrielles en profitant du développement des cartes graphiques (GPU) pour paralléliser massivement les calculs

- **CONVolutional neural NETworks** essentiellement pour l'analyse d'images

Introduction à l'apprentissage profond

Trois grandes familles de réseaux d'apprentissage profond sont développées avec des ambitions industrielles en profitant du développement des cartes graphiques (GPU) pour paralléliser massivement les calculs

- **CONVolutional neural NETworks** essentiellement pour l'analyse d'images
- **Long-Short Term Memory** des propriétés d'autocorrélation sont prises en compte (signal, analyse du langage naturel...)

Introduction à l'apprentissage profond

Trois grandes familles de réseaux d'apprentissage profond sont développées avec des ambitions industrielles en profitant du développement des cartes graphiques (GPU) pour paralléliser massivement les calculs

- **CONVolutional neural NETworks** essentiellement pour l'analyse d'images
- **Long-Short Term Memory** des propriétés d'autocorrélation sont prises en compte (signal, analyse du langage naturel...)
- **autoEncoder decoder** en apprentissage non supervisé (débruitage d'images ou signaux...)

Couches pour l'apprentissage profond

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques

Couches pour l'apprentissage profond

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques

- **fully connected** couche classique de perceptron et dernière couche d'un réseau profond

Couches pour l'apprentissage profond

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques

- **fully connected** couche classique de perceptron et dernière couche d'un réseau profond
- **convolution** opère une convolution sur le signal d'entrée en associant une réduction de dimension

Couches pour l'apprentissage profond

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques

- **fully connected** couche classique de perceptron et dernière couche d'un réseau profond
- **convolution** opère une convolution sur le signal d'entrée en associant une réduction de dimension
- **pooling** réduction de dimension en remplaçant un sous-ensemble des entrées par une valeur

Couches pour l'apprentissage profond

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques

- **fully connected** couche classique de perceptron et dernière couche d'un réseau profond
- **convolution** opère une convolution sur le signal d'entrée en associant une réduction de dimension
- **pooling** réduction de dimension en remplaçant un sous-ensemble des entrées par une valeur
- **drop out** supprimer des neurones d'une couche afin de réduire la dimension
- ...

Couches pour l'apprentissage profond

Sans bases de données très volumineuses et moyens de calcul substantiels il est illusoire de vouloir apprendre un réseau profond impliquant l'estimation de millions de paramètres

https://www.deeplearningbook.org/