Réseaux de neurones

Introduction

- Intelligence Artificielle
  - Cognitivisme (sciences cognitives)
  - Connexionisme (réseaux de neurones)

Le cerveau : réseau très complexe ayant un grand nombre de cellules de base interconnectées. Il y a $\sim$ 100 milliards de neurones et 10^{15} connexions.

Les réseaux de neurones : modèle très simple du cerveau où les unités de calcul élémentaire sont interconnectées. En particulier on ne prends pas en compte de nombreuses caractéristiques biologiques.

Le connexionisme : étudie comment rendre compte des processus cognitifs à partir d'un ensemble d'unités, dotées chacune d'une faible puissance de calcul et interconnectées en réseau ?

C'est une méthode d'apprentissage non symbolique car on produit un résultat difficile à interpréter (contrairement aux arbres de décision).

Différents modèles de réseaux de neurones

- Le perceptron
- Les réseaux multi-couches

Le perceptron

- Un perceptron linéaire à seuil prend en entrée n valeurs $x_1, \ldots, x_n$ et calcule une sortie o.
- Il est défini par $n+1$ constantes :
  - Les coefficients (poids) synaptiques $w_1, \ldots, w_n$
  - Le seuil (ou biais) θ
- La sortie o est calculée par :

$$o = \begin{cases} 1 & \text{si } \sum_{i=1}^{n} w_i x_i > \theta \\ 0 & \text{sinon} \end{cases}$$

- Les entrées peuvent être à valeurs dans $\{0, 1\}$ ou réelles, les coefficients peuvent être entiers ou réels.

Simplification du modèle

On constate :

$$\sum_{i=1}^{n} w_i x_i > \theta \text{ ssi } \sum_{i=1}^{n} w_i x_i - \theta > 0$$

On pose une entrée supplémentaire $x_0 = 1$ et on lui associe un coefficient $w_o = -\theta$. On a donc

$$o = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$$

Simplification du modèle

1. On calcul d'abord de $x = \sum_{i=0}^{n} w_i x_i$ qu'on appelle le potentiel post-synaptique.
2. On applique ensuite au résultat x la fonction d'activation :

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

Exemple : un perceptron qui calcule le OU logique

- Entrées : $x_0 = 1$, x_1 et x_2 arbitraires.
- Coefficients : $w_0 = -0.5$, $w_1 = 1$ et $w_2 = 2$.
- Sortie : $o = x_1 \vee x_2$.

Interprétation géométrique

- Soit S un ensemble d'exemples dans $\mathbb{R}^n \times \{0, 1\}$.
  - Soit $S_0 = \{s \in \mathbb{R}^n \mid (s, 0) \in S\}$
  - Soit $S_1 = \{s \in \mathbb{R}^n \mid (s, 1) \in S\}$.
- S est dit linéairement séparable s'il existe un hyperplan H de $\mathbb{R}^n$ tel que les ensemble S_0 et S_1 soient situés de part et d'autre de cet hyperplan.

Résultats sur les perceptrons

Théorème : Un perceptron linéaire à seuil à n entrées divise l'espace des entrées $\mathbb{R}^n$ en deux sous-espaces délimités par un hyperplan.

Théorème : Tout ensemble linéairement séparable peut être discriminé par un perceptron.

Théorème : Le XOR ne peut pas être calculé par un perceptron linéaire à seuil.

Algorithme d'apprentissage par correction d'erreur

- Problème : Trouver les poids d'un perceptron qui classifie correctement un ensemble S d'apprentissage de $\{0,1\}^n \times \{0,1\}$ ou $\mathbb{R}^n \times \{0,1\}$.
- Notations :
  - $\vec{x}$ une description (dans $\mathbb{R}^n$ ou $\{0,1\}^n$)
  - S est un ensemble de couples $(\vec{x}, c)$
  - $(\vec{x}^s, c^s)$: le s-ième élément de S
  - o^s la sortie du perceptron pour l'entrée $\vec{x}^s$
- Rappel : Il existe une entrée x_0 de valeur 1.

Algorithme par correction d'erreurs

Entrée : un échantillon S

Pour $i = 0 \ldots n$ initialiser aléatoirement les poids w_i

Répéter

Prendre un exemple $(\vec{x}, c)$ dans S

Calculer la sortie o pour l'entrée $\vec{x}$

Pour $i = 0 \ldots n$

$$w_i \leftarrow w_i + (c - o) \times x_i$$

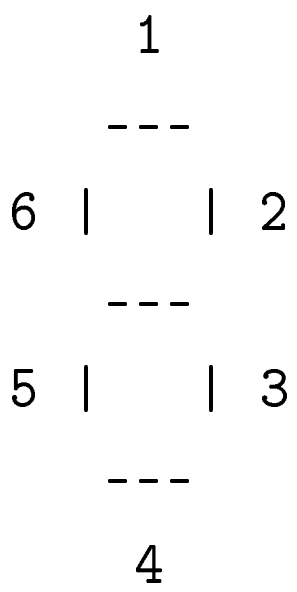
Fin Pour

Fin Répéter

Sortie : Un perceptron P défini par $(w_0, w_1, \ldots, w_n)$

Exemple

On veut apprendre si un chiffre est pair ou impair. Les chiffres est representé sur une retine de 7 leds.



On considère un ensemble complet

$$S = \left\{ \begin{array}{l} (1111110, 0), (0110000, 1), (1101101, 0), \\ (1111001, 1), (0110011, 0), (1011011, 1), \\ (0011111, 0), (1110000, 1), (1111111, 0), (1111011, 1) \end{array} \right\}$$

Trace de l'algorithme

Etape	w	x	c	o
1	$(1,1,1,1,1,1,1,1)$	$(1,1,1,1,1,1,1,0)$	0	1
2	$(0,0,0,0,0,0,0,1)$	$(1,0,1,1,0,0,0,0)$	1	0
3	$(1,0,1,1,0,0,0,1)$	$(1,1,1,0,1,1,0,1)$	0	1
4	$(0,-1,0,1,-1,-1,0,0)$	$(1,1,1,1,1,0,0,1)$	1	0
5	$(1,0,1,2,0,-1,0,1)$	$(1,0,1,1,0,0,1,1)$	0	1
6	$(0,0,0,1,0,-1,-1,0)$	$(1,1,0,1,1,0,1,1)$	1	0
7	$(1,1,0,2,1,-1,0,1)$	$(1,0,0,1,1,1,1,1)$	0	1
8	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,0,0,0,0)$	1	1
9	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,1,1,1,1)$	0	0
10	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,1,0,1,1)$	1	1

Vérification

Coefficients $= (0, 1, 0, 1, 0, -2, -1, 0)$

Entrée	Sortie
$(1, 1, 1, 1, 1, 1, 1, 0)$	0
$(1, 0, 1, 1, 0, 0, 0, 0)$	1
$(1, 1, 1, 0, 1, 1, 0, 1)$	0
$(1, 1, 1, 1, 1, 0, 0, 1)$	1
$(1, 0, 1, 1, 0, 0, 1, 1)$	0
$(1, 1, 0, 1, 1, 0, 1, 1)$	1
$(1, 0, 0, 1, 1, 1, 1, 1)$	0
$(1, 1, 1, 1, 0, 0, 0, 0)$	1
$(1, 1, 1, 1, 1, 1, 1, 1)$	0
$(1, 1, 1, 1, 1, 0, 1, 1)$	1

Remarques sur l'algorithme par correction d'erreurs

- Dans quel ordre présente-on les exemples ?
  aléatoirement ? en suivant un ordre prédéfini ? on les présente tous ?
- Critères d'arrêt de la boucle ?
  après un nb d'étapes ? après traitement de tous les exemples ? lorsque les poids ne sont plus modifiés ?

Résultat sur l'algorithme par correction d'erreurs

Théorème : Si S est linéairement séparable et si les exemples sont présentés équitablement, la procédure d'apprentissage par correction d'erreur converge vers un perceptron linéaire à seuil qui calcule S.

Problèmes et Questions

- Qu'est-ce que se passe si S n'est pas linéairement séparable ? L'algorithme ne converge pas.
- Est-ce qu'on peut borner la valeur des poids et le seuil ? Oui, mais par un nombre très grand, donc pas d'application pratique.
- Combien de pas faut-il pour converger ? Exponentiel.
- Est-ce que la solution trouvée est robuste ? Non.
- Est-ce que l'algorithme est tolérant aux bruits ? Non.

Algorithme par descente du gradient

– Idée : plutôt que de classifier correctement tous les exemples, on définit une fonction d'erreur qu'on essaie de minimiser.

– Un perceptron linéaire prend en entrée un vecteur $\vec{x}$ de n valeurs (on élimine θ) et calcule la sortie o à l'aide d'un vecteur $\vec{w}$ de constantes comme suit :

$$o = \vec{x}.\vec{w} = \sum_{i=1}^{n} w_i x_i$$

Erreur d'un perceptron

L'erreur d'un perceptron P défini par les coefficients $\vec{w}$ sur un échantillon S d'exemples est donnée par :

$$E(\vec{w}) = 1/2 \sum_{(\vec{x},c)\in S} (c - o)^2$$

où o est la sortie calculée par P sur l'entrée $\vec{x}$.

Remarque : $E(\vec{w}) = 0$ ssi P classifie correctement tout l'ensemble S.

But : Detérminer un $\vec{w}$ qui minimise $E(\vec{w})$. On utilise la méthode du gradient.

Méthode du gradient

- Étant donné une fonction f on construit une suite (x_n)
- Cette suite devrait s'approcher du minimum.
- On part de x_0 quelconque et on définit $x_{n+1} = x_n - \epsilon f'(x_n)$, où ϵ est une valeur "bien choisi".
- Le choix de ϵ est empirique (si trop petit, alors nb d'itérations élevé, si trop grand, alors peut ne pas converger).
- Rien ne garantit que le miminum trouvé est un minimum global.

Concepts mathématiques pour la descente du gradient

- $E(\vec{w})$ est une fonction de n variables. La méthode du gradient peut être étendue à n variables.
- On calcule la dérivée partielle de E par rapport à w_i :

$$\frac{\partial(\vec{w})}{\partial w_i} = \sum_{(\vec{x},c)\in S} (c - o) \times (-x_i)$$

- On définit

$$\Delta w_i = -\epsilon \times \frac{\partial(\vec{w})}{\partial w_i} = \epsilon \times \sum_{(\vec{x},c)\in S} (c - o) \times x_i$$

L'algorithme d'apprentissage par descente de gradient

Entrée : un échantillon S de $\mathbb{R}^n \times \{0,1\}$ et ϵ

Pour $i = 0..n$ initialiser aléatoirement les poids w_i ;

Répéter

  Pour tout i $\Delta w_i \leftarrow 0$;

  Pour tout exemple $(\vec{x}, c)$ de S :

    Calculer la sortie o ;

    Pour tout i $\Delta w_i \leftarrow \Delta w_i + \epsilon \times (c - o) \times x_i$;

  Pour tout i $w_i \leftarrow w_i + \Delta w_i$;

Fin Répéter

Sortie : Un perceptron P défini par $(w_1, \ldots, w_n)$

Remarques sur l'algorithme par descente de gradient

- L'algorithme converge pour un ϵ bien choisi suffisamment petit même si l'ensemble d'entrée n'est pas linéairement séparable.
- Si ϵ est trop grand, on risque d'osciller autour du minimum.
- La convergente peut être lente, beaucoup de calcul à chaque itération.

Vers une variation

– On modifie les poids à chaque présentation d'exemple avec la formule :

$$\Delta w_i = \epsilon \times (c - o) \times x_i$$

L'algorithme d'apprentissage de Widrow-Hoff

Entrée : un échantillon S de $\mathbb{R}^n \times \{0,1\}$ ou $\{0,1\}^n \times \{0,1\}$

Pour $i = 0..n$ initialiser aléatoirement les poids w_i ;

Répéter

Prendre un exemple $(\vec{x}, c)$ dans S

Calculer la sortie o pour l'entrée $\vec{x}$

Pour $i = 1..n$ $\quad w_i \leftarrow w_i + \epsilon \times (c - o) \times x_i$

Fin Répéter

Sortie : Un perceptron P défini par $(w_1, \ldots, w_n)$

Remarques sur l'algorithme de Widrow-Hoff

- En général on parcourt S dans un ordre prédéfini.
- Critère d'arrêt : pour un passage complet de S toutes les modifications de poids sont $\leq$ à un seuil prédéfini.
- Il n'y a pas correction d'erreur, mais modification dans presque tous les cas.
- L'algorithme converge vers une solution optimale (si on a bien choisi ϵ).
- Meilleure tolérance aux bruits.