Table des matières

1 Présentation

Les voitures autonomes sont aujourd'hui au coeur de l'actualité, tant en terme d'innovation que de débats au sujet de leurs fiabilitées ou de leur autonomie. Ces dernières années, le développement de nouveaux algorithmes couplés à la puissance de calcul phénomenale fournit par des ordinateurs toujours plus compact ont permis la démocratisation de nouvelles façons de programmer. Les algorithmes dédié aux voitures autonome se basent aujourd'hui en partie sur un modèle appelée *deeplearning* (ou apprentissage profond) qui comprend la structure algorithmique du réseau neuronal. Ce système se démarque des simples algorithmes impératifs classiques par une étape dite d'apprentissage. Nous reviendrons sur ce concept plus en details par la suite.

Nous nous sommes alors posé la question suivante : **Comment fonctionne ce genre d'algorithmes et est-il possible, à notre échelle, d'en implémenter un afin d'automatiser une voiture radio-commandée ?**

Nous traiterons dans un premier temps du fonctionnement de base d'un système d'aprentissage profond, nous aborderons ensuite le cas des structures neuronales plus complexe, discutant des avantages et des limites propres à chacunes. Nous détaillerons enfin la conception de notre voiture radio-commandée autonome.

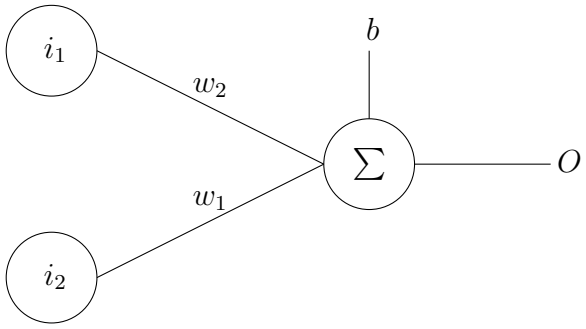
2 Les réseaux neuronaux

Les réseaux neuronaux sont des outils algorithmiques permettant principalement de catégoriser des données selon des règles pré-établies. Ils peuvent être utilisés pour des taches diverses et variées telles que les prévisions météorologiques ou, pour ce qui nous intéresse, la reconnaissance d'image. Mais finalement, qu'est-ce qu'un réseau de neurone ?

2.1 Le perceptron

Le principe du perceptron a été inspiré par le fonctionnement des neurones humains. Il génère une sortie paramamétrée par des variables d'entrée. Algorithmiquement, le perceptron effectue la somme pondérée des valeurs d'entrées par des *poids* (notés w par la suite) à laquelle on ajoute un biais (noté b). Le résultat obtenu passe à travers une fonction d'activation ($\mathbb{R} \to \mathbb{R}$) afin de générer une sortie exploitable. Nous reparlerons plus en détail de ces fonctions par la suite.

Le plus simple des perceptrons (ou neurones) est consitué de deux entrés et d'une sortie. On peut le représenter selon le graphique ci-dessous :



Si l'on note σ la fonction d'activation, on obtient alors :

$$O = \sigma(i_1 \times w_1 + i_2 \times w_2 + b) \tag{1}$$

On peut réécrire matriciellement cette expression en considérant les matrices :

$$I = (i_1 \ i_2) \quad P = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \quad B = (b)$$

On obtient alors :

$$O = \sigma\left(I \cdot P + B\right) \tag{2}$$

On définit la fonction d'activation pour une matrice :

Soit $A = (a_{ij})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant p}} \in \mathcal{M}_{np}(\mathbb{R})$

$$\sigma[A] = (\sigma(a_{ij}))_{1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant p} \tag{3}$$

Nous allons maintenant étudier l'application $f_{2,1}$ qui correspond à un perceptron à deux entrés. C'est le plus petit système présentant un interêt. Il comprend deux entrées (i_1, i_2), un neurones (n_1), deux réels (w_1, w_2) appellés poids et le biais du neurone (b_1). On a donc :

$$P = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \; et \; B = (b)$$

$$n_1 = f_{2,1}((i_1 \; i_2)) = \sigma\left[(i_1 \; i_2).P + B\right] \tag{4}$$

On peut alors schématiser un perceptron de la manière suivante :

Le perceptron permet la classification d'ensemble linéairement séparable. C'est a dire que pour un espace E de dimension n $\in$ donné, si l'on peut classer les éléments de E de *part et d'autre* d'un hyper-plan de E alors E est dit linéairement séparable.

2.2 Définition

Un réseau de neurones (à 0 couche (ou étage) pour commencer) peut etre rerésenté par l'application suivante :

Soient $P \in \mathcal{M}_{np}(\mathbb{R}), B \in \mathcal{M}_{1,p}(\mathbb{R})$

$$\begin{array}{rccc} f_{pn} : & \mathcal{M}_{1,n}(\mathbb{R}) & \longrightarrow & \mathcal{M}_{1,p}(\mathbb{R}) \\ & \left(x_1 \; ... \; x_n\right) & \longmapsto & \sigma\left[\left(x_1 \; ... \; x_n\right).P - B\right] \end{array}$$

Détaillons cette application. Elle prend en argument n scalaires vu comme un matrice ligne et effectue le produit matriciel de cette matrice par une matrice de $\mathcal{M}_{n,p}(\mathbb{R})$ fixée dite de *poids* afin de retourner p scalaires toujours sous forme d'une matrice ligne. On donne enfin p réels appellés biais que l'on retranche au vecteurs précédent. C'est biais sont un degré de liberté suplémentaire qui permet de donner de l'importance ou non un neurone en particulier. Les valeurs ainsi obtenue ne sont pas satisfaisantes, elles sont dispérées sur un domaine de définition trop large pour être interpretées. Il convient donc, à l'instar des neurones traditionels, d'utiliser une fonction d'activation qui restreint le domaine de définition des valeurs. Les plus courantes sont la fonction sigmoid (σ) et tangente hyperbolique.

2.3 Exemples

Pour un premier exemple, on peut considérer la porte logique AND. Vérifions que l'on peut représenter cette porte logique avec un perceptron.