Rapport de Projet Long
Équipe HAL9000

Hamelain Christian, Hasan Pierre-Yves, Gaspart Quentin, Trestour Fabien

11 février 2016

Table des matières

Première partie

Introduction

0.1 Contexte du projet

CentraleSupélec propose à ses étudiants en deuxième année de participer à des projets longs qui se réalisent en équipe, tout au long de l'année sur un thême mêlant plusieurs compétences que l'ingénieur se doit de posséder.

Dans notre cas, c'est l'élaboration de réseaux neuronaux et les méthodes de Deep Learning que 12 éléves ont étudié, répartis en 3 groupes, dont HAL9000.

Ce rapport présente les avancement de ce dernier groupe sur le sujet tout au long de l'année.

0.2 Présentation du sujet

Ce projet aborde le sujet du Deep Learning. Cette méthode spécifique de machine learning est dérivée du concept de réseau de neurones.

Les réseaux de neurone sont une modélisation simple du fonctionnement cérébral à l'échelle cellulaire. Cette approche a vu le jour avec les études de McCulloch et Pitts dès la fin des années 50. Malgré certains écueils dans les années 70, l'approche connexioniste des réseaux de neurones a su se développer et devenir un sujet de recherche populaire dans les dernières années, notament grâce aux capacités d'adaptabilité et de généralisation de ces réseaux qui en font d'excellents candidats pour des applications telles que la reconnaissance d'image ou la classification.

0.3 Déroulement du projet

Tout d'abord, il s'agissait pour nous de comprendre le fonctionnement de ces structures et commencer à coder des structures élémentaires afin de comprendre les enjeux du machine learning. Nous avons pour cela travaillé avec la base de données MNIST de Yves LeCun et en JAVA. Afin de travailler en groupe de manière efficace, nous avons aussi utilisé l'outil GIT.

Par la suite nous nous sommes interessés aux architectures profondes, chaque groupe se penchant sur une architecture spécifique. Notre équipe s'est en particulier intéressée aux machines de Boltzmann, en commençant par les machines de Boltzmann restreintes (RBM), puis la structure de Deep Learning associée.

Cette étude a été encadrée par Joanna Tomasik et Arpad Rimmel, enseignants à CentraleSupélec, campus de Gif-sur-Yvette.

Deuxième partie

Première approche du problème : le Perceptron

Chapitre 1

Architecture d'un perceptron

1.1 Introduction

Ici, l'objectif était la reconnaissance des caractères de la base de données MNIST grâce à un perceptron.

Le perceptron fait partie des architectures de réseaux neuronaux les plus simples. Son étude nous a donc permis de s'introduire à la problématique du machine learning avant d'approfondir en étudiant des structures plus complexes.

1.2 La brique de base : le neurone

Le neurone est le composant élémentaire des réseaux neuronaux. Il est une modélisation du fonctionnement des neurones du système nerveux humains.

Chaque neurone reçoit un signal via une entrée, qui correspond aux dendrites des systèmes biologiques. Le neurone prend en compte la valeur de toutes ses entrées et en déduit la valeur de sortie. Cette sortie est ensuite propagée par le biais d'un axone vers un autre neurone.

La sortie du $j^{ième}$ neurone est donnée par la formule :

$$s_j(x) = f(\sum_{k=1}^{N} w_{j,k} * x_k + w_0) \tag{1.1}$$

où :
- s est la valeur de la sortie
- f est la fonction d'activation.
- N est la dimension du vecteur d'entrée.
- $w_{j,k}$ est la $k^{ième}$ composante du vecteur de poids w_j du $j^{ième}$ neurone. w_0 est le biais du neurone. Cette valeur correspond au poids d'une entrée fictive valant toujours 1.
- x_k est la $k^{ième}$ composante du vecteur d'entrée x.

L'entrée x est un vecteur défini par un ensemble de caractéristiques que l'on choisit pour représenter les données d'entrées. Dans le cas d'une image par exemple, ce vecteur d'entrée peut être composé de la valeur de tous les pixels

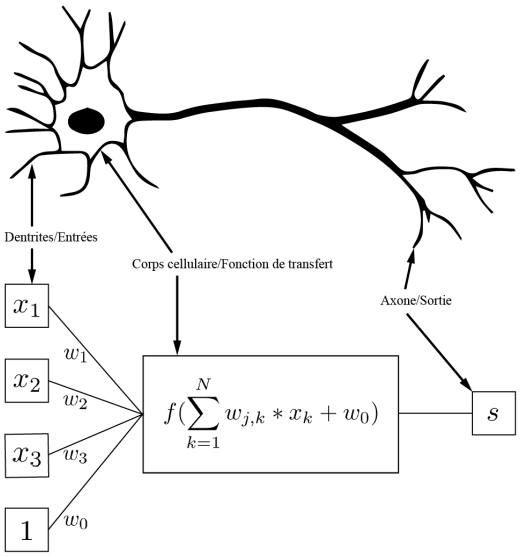

FIGURE 1.1 – Analogie entre neurone biologique et neurone formel.

de l'image. D'autres représentations moins triviales peuvent être choisies afin de réduire la quantité d'information à traiter et de s'approcher au mieux des données signifiactives de la problématique traitée.

La fonction d'activation permet de définir le comportement du neurone. Selon la définition de cette fonction on aura une sortie à valeurs discrètes ou continues, centrées en 0 ou en 0,5. Le choix de la fonction est donc étroitement lié avec le problème à traiter. Il existe différents types de fonctions d'activation. Parmis les plus utilisées figurent :

— La fonction échelon

$$f(x) = \mathbf{1}_{\mathbb{R}_+^*} \tag{1.2}$$

— Les fonctions linéaires

$$f(x) = \alpha * x + \beta \tag{1.3}$$

— La fonction sigmoïde

$$f(x) = \frac{1}{1 + e^{-\lambda * x}} \tag{1.4}$$

— La fonction tangente hyperbolique

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.5}$$

Le vecteur de poids du j$^{\text{ième}}$ neurone représente la pondération de chacune des entrées de ce neurone. C'est ce paramètre qui permet de modifier le réseau de neurone sans avoir à modifier son architecture. Toutes les propriétés d'un réseau neuronal sont donc issues de ce vecteur de pondération et de ses variations.

1.3 Organisation des couches

Il existe de nombreuses architectures de réseau de neurone. Elles peuvent être récurrentes ou non, entièrement connectées ou seulement partiellement, organisées en couches, ... De nombreuses propriétés permettent de caractériser une architecture de réseau de neurone.

Le Perceptron est un modèle assez élémentaire de réseau. Il est constitué en couches totalement connectées.

On peut distinguer trois types de couches : la couche d'entrée, les couches cachées et la couche de sortie.

La couche d'entrée est assez élémentaire. C'est tout simplement une couche dont la valeur sera le vecteur d'entrée x. Cette couche doit donc être constituée d'autant de neurones que le vecteur d'entrée a de dimensions. Les neurones de cette couche ont pour fonction d'activation l'identité.

Les couches cachées sont celles qui effectuent les calculs. Les principales propriétés du réseau sont héritées de cet empilement de couches. On comprend alors mieux l'intérêt actuel pour le Deep Learning, c'est-à-dire l'apprentissage des réseaux avec un grand nombre de couches cachées. Le nombre de neurones dans chaque couche et les fonctions d'activation utilisées par les neurones sont choisies par le concepteur du réseau selon le problème traité par le réseau.

Afin d'expliciter l'importance de l'architecture du réseau, abordons l'exemple usuel du ou exclusif.

Essayons de réaliser avec un unique neurone la fonction logique XOR. Ce neurone aura deux entrées x_1 et x_2 à valeurs dans $\{0\,;1\}$ et une sortie s, elle aussi à valeurs dans $\{0\,;1\}$. On prendra comme fonction d'activation la fonction de Heaviside, c'est à dire $\mathbf{1}_{\mathbb{R}_+^*}$. Ce neurone aura par conséquent un fonctionnement totalement binaire. Il s'agit donc ici de déterminer les pondérations w_1 et w_2, respectivement associées aux entrées x_1 et x_2, qui conviennent pour obtenir en sortie la valeur $x_1 \oplus x_2$.

Pour un tel problème, le neurone agit comme un séparateur linéaire de l'espace des entrées. L'équation de la droite séparant le demi-espace définit par $s = 0$ de celui définit par $s = 1$, découle alors directement de l'équation 1.1 :

$$w_2 * x_2 + w_1 * x_1 + w_0 = 0 \tag{1.6}$$

On constate ici l'intérêt du biais, qui permet de réaliser des séparations affines de l'espace des entrées et pas seulement des fonctions linéaires.

Les limites d'un neurone seul sont alors évidentes : toutes les séparations de l'espace des entrées ne sont pas affine, et le cas du XOR en est déjà une illustration.

Il est alors nécessaire d'introduire la structure de réseau. En prennant une couche d'entrée, une couche cachée et une couche de sortie, respectivement composées de deux, deux et un neurone, on peut contourner le problème sus-cité. Cette structure permet en fait de générer deux séparations de l'espace des entrées.

La séparation que l'on cherche à effectuer est en effet de la forme :

$$f(x_1, x_2) = 0 \Leftrightarrow \begin{cases} x_1 + x_2 - 0,5 & < & 0 \\ x_1 + x_2 - 1,5 & > & 0 \end{cases} \tag{1.7}$$

On obtient ainsi directement les valeurs de poids désirées grâce à ces équations :

$$blablabla \tag{1.8}$$

Ce premier exemple simpliste manifeste donc la nécessité d'adapter la structure du réseau au problème traité.

De manière plus générale, le nombres de neurones du réseau fait varier le nombre de poids du réseau, et donc la capacité du réseau à séparer des ensembles multiples et complexes. Ainsi, un réseau sous-dimensionné mène à des résultats pas assez précis et un réseau sur-dimensionné mène à du sur-apprentissage. On a donc environ la loi suivante :

$$N < \frac{1}{10} * T * dim(s) \tag{1.9}$$

— N est le nombre de neurones dans le réseau.
— T est le nombre de vecteurs de la base d'apprentissage.
— s est le vecteur de sortie.

De plus, le nombre de neurones dans une couche doit être de moins de trois fois celui de la couche précédente.

Ces deux approximations donnent donc une première idée de la structure à adopter pour un réseau de neurones.

Chapitre 2

Algorithmes d'apprentissage

2.1 salut

123

2.2 c'est cool

Chapitre 3

La base MNIST

Chapitre 4

Influence des paramètres sur l'apprentissage

Troisième partie

Approfondir le sujet : Restricted Boltzmann Machine