Projet MDI343
Apprentissage profond, réseau de neurones, *etc.*

Antoine BIARD & Vincent BODIN

Résumé

Ce projet s'attelle à la question de la représentation des données en *machine learning*. Ces représentations, utilisées ensuite par l'algorithme de décision - régression logistique, SVM... - jouent un rôle primordial car porteuses de l'information. A la méthode classique de création de *features* par des connaissances *a priori*, on oppose des méthodes d'extraction de représentation par des structures dites profondes. Nous étudierons quelques algorithmes et comparerons les résultats.

Table des matières

1 Introduction : de l'importance de la représentation des données

Ces dernières années se sont imposées massivement des méthodes dites d'apprentissage profond - *deep learning* en anglais. La communauté scientifique a accru son intérêt en la matière au vue de leurs résultats, dans des domaines aussi variés que la vision par ordinateur ou la reconnaissance vocale. Atteignant souvent des performances au moins équivalentes à l'état de l'art, de nombreux algorithmes de type profond ont été créés pour tenter d'améliorer encore la représentation.

La représentation est une clef fondamentale dans l'élaboration d'un processus d'intelligence artificielle, en ceci qu'elle concentre l'information des données. On distingue en effet une première étape d'apprentissage de la représentation, puis une seconde d'apprentissage, à partir de cette représentation, d'une décision - pour de la classification par exemple. Les algorithmes effectuant cette dernière tâche sont hautement dépendants des données fournies en entrée. Une amélioration de la représentation entraîne de fait une meilleure performance dans la prise de décision. Nous renvoyons à la deuxième section de [1] pour des quantifications d'amélioration de la performance avec des représentations plus élaborées.

Traditionnellement, les descripteurs - ou encore *features* en anglais - sont extraits de façon dite supervisée à partir des données. Ceci nécessite une connaissance *a priori* dans le domaine, puisque cela requiert souvent une importante étape de traitement des données pour en extraire l'information - citons parmi tant d'autre en vision les points de Harris et les SIFT ou encore le spectrogramme et les *Mel-frequency cepstrum coefficients* en audio. Cela repose sur l'idée que l'homme a la capacité d'extraire et d'organiser l'information intrinsèque des données. Pour autant, dans un cadre d'intelligence artificielle pure, il serait désirable de s'affranchir de cette hypothèse et d'implémenter des algorithmes capables d'apprendre une représentation convenable sans supervision - non-supervisé. En outre, les types de données ayant tendance à devenir de plus en plus variés - image, son, web... - une telle connaissance dans chaque domaine s'avère illusoire et le manque de généralité de ces méthodes se paye dans la nécessité d'expertise dans chaque domaine. Il s'agit d'implémenter des méthodes extrayant des données une représentation abstraite.

Une manière d'atteindre ce but est l'extraction d'une représentation par apprentissage profond - *deep learning*. Une telle représentation s'obtient par l'ajout de couches successives. Une première représentation est obtenue par un processus à partir des données pures, puis cette représentation est elle-même raffinée, et ainsi de suite. Dans l'espoir d'obtenir une représentation abstraite, la non-linéarité devient nécessaire puisque sans elle, l'ajout de deux couches linéaires serait équivalent à une unique couche - le concept de profondeur perdrait sens.

Nous allons par la suite détailler les algorithmes d'apprentissage profond les plus classique, puis nous comparerons les performances de certains d'entre eux.

2 Revue élémentaire des méthodes de *deep learning*

2.1 Réseau de neurones

Un réseau de neurone peut être vu comme l'utilisation en série et parallèle plusieurs neurones simples, cf. Fig.(1). Un neurone est un noeud, avec un certain nombre d'entrée, et une fonction d'activation, disons σ - souvent tangente hyperbolique ou une sigmoïde, l'idée étant d'avoir une fonction proche de la fonction seuil, qui donne une activation *soft*. Pour un neurone ayant p entrée, chacun pondéré par un poids w_i, et un *offset* b, la sortie du neurone est :

$$h_{w,b}(x) = \sigma(w^T x) = \sigma\left(\sum_{i=1}^{p} w_i x_i + b\right) \tag{1}$$

Dans le cas d'un réseau de neurones, en utilisant une formule de chaîne où chaque sortie de neurone intermédiaire devient l'entrée du neurone suivant, on obtient un réseau du type de la Fig.(1). On voit ici apparaître naturellement l'idée de profondeur dans l'apprentissage par accumulation de couche de neurone, bien qu'il s'agisse du modèle le plus basic d'apprentissage profond. La fonction de sortie dépend de tous les paramètres w, b à chaque couche et chaque noeuds. La difficulté majeure dans l'apprentissage profond réside en l'estimation des paramètres optimaux - dans le sens où la sortie est la représentation la plus adéquate.

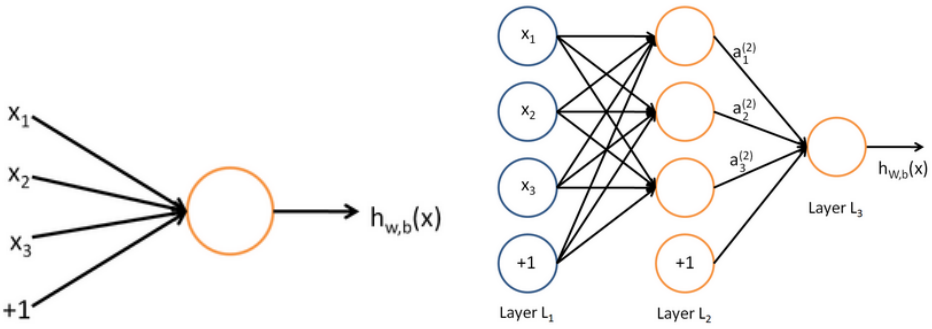

FIGURE 1 – (gauche) Un neurone avec trois entrées (x_1, x_2, x_3) et un *offset* ; (droite) un réseau de neurones de taille $(3, 3, 1)$ avec des *offset* aux deux premières couches.

2.2 Multilayer Perceptron (MLP)

Le second modèle classique en apprentissage profond est le perceptron multicouche (MLP). Il s'agit du premier exemple d'algorithme utilisant la méthode dite de *backpropagation*. Bien que peu utilisé seul en pratique, car la fonction d'optimisation n'est pas convexe et il y a de grand risque de trouvé un optimum local et non global, le MLP correspond aux prémisses du *deep learning* et sera réutilisé pour des modèles plus complexes par la suite.

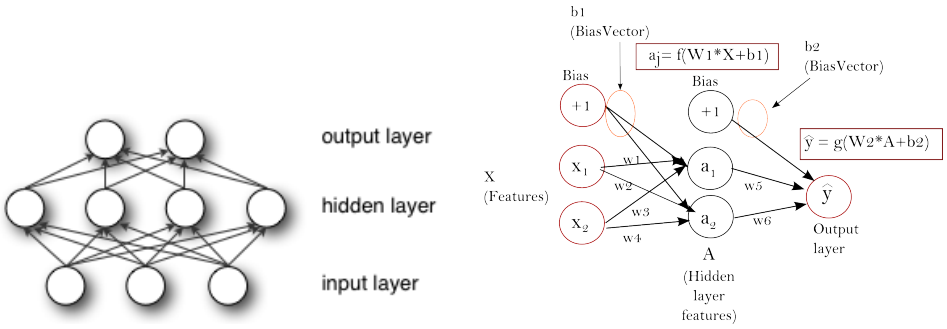

FIGURE 2 – (gauche) Un MLP avec une seule couche de variables cachées ; (droite) structure de MLP avec l'ajout de biais et les notions de poids.

Le MLP possède une fonction d'activation σ - tangente hyperbolique ou la fonction logistique souvent - des noeuds de biais b et des poids W associés à chaque arrête. Pour chaque couche, le paramètre est donc $\theta = (W, b)$, et pour une entrée x de dimension d - *i.e.* d noeuds - la sortie de cette couche est :

$$s(x) = \sigma(Wx + b) \tag{2}$$

Ceci forme un vecteur de taille disons m - sur la Fif.(2) par exemple pour la première couche $d = 3$ et $m = 4$ - qui est envoyé comme entrée à la couche suivante.

La nouveauté du MLP comparé au réseau de neurone réside en la manière d'apprendre l'hyper-paramètre $\theta = (W_1, \cdots, W_p, b_1, \cdots, b_p)$ - p est le nombre de couches. L'algorithme en question sera développé plus tard mais nous en donnons un aperçu. L'initialisation s'effectue classiquement de manière aléatoire - bien que dans le cas du DBN on lui envoie des poids définis en entrée. Dans [2], il est expliqué comment tirer les poids initialement. L'algorithme contient ensuite deux étapes :

Forward pass. On part des variables visibles v et on remonte le graphe avec les poids de la structure. On en déduit une représentation.

Backpropagation. A partir de cette représentation, on effectue le chemin inverse en descendant le graphe et en comptabilisant les erreurs commises. En comparant les résultats de la descente avec les variables visibles, on peut mettre à jour les poids.

Cela est répété un certain nombre de fois et cette mise à jour par *backpropagation* permet d'obtenir une représentation convenable. Toutefois ceci est très dépendant de l'initialisation.

Notons que ceci ne fournit qu'une représentation. Pour effectuer une classification, on utilise souvent une base de données étiquetée. Chaque représentation est associée à une étiquette et on se sert des représentations comme *feature* de l'algorithme de décision - régression logistique, SVM...

2.3 Restricted Boltzmann Machine (RBM)

2.3.1 Machine de Boltzmann

Une machine de Boltzmann est un réseau de variables aléatoires couplées. On se restreint au cas où les données sont binaire, on note les variables visibles v et les cachées h. Une machine de Boltzmann est représentée en Fig.(3) à gauche.

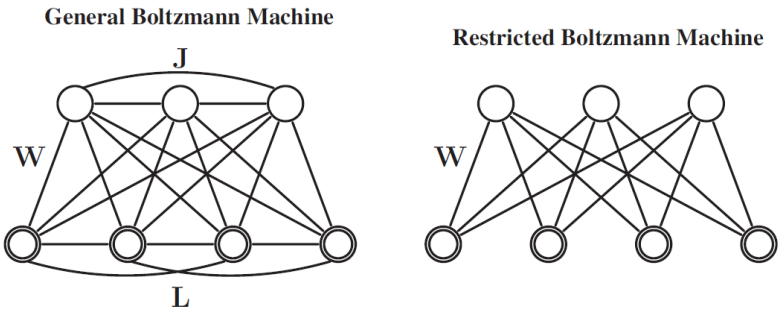

FIGURE 3 – (gauche) Une machine de Boltzmann générale ; (droite) machine de Boltzmann restreinte (RBM).

L'énergie des états (v, h) prend alors la forme :

$$E(v, h; \theta) = -v^T W h - \frac{1}{2} v^T L v - \frac{1}{2} h^T J h \tag{3}$$

(où l'on ne prend pas en compte ici les termes de biais qui correspondent à un recentrage des (v, h)). Il s'agit d'un cas de *energy-based models* (EBM). Dans ce cas, la probabilité d'un état v s'écrit :

$$p(v; \theta) = \frac{1}{\mathcal{Z}} \sum_h e^{-E(v,h;\theta)} \tag{4}$$

où $\mathcal{Z} = \sum_{x,h} e^{-E(v,h;\theta)}$ est la fonction de partition. Ces modèles ont la propriété que le gradient de $\log p(v)$ - log-vraisemblance - se décompose en deux termes, une phase positive et une négative. Pour autant la vraisemblance n'est pas calculable car exponentielle à la fois en le nombre de variables cachées et visibles.

2.4 Restricted Boltzmann Machine

Comme montré dans la Fig.(3), un RBM est une machine de Boltzmann où les interactions inter-couches sont gelées. De par la forme de son graphe, le RBM a des propriétés d'indépendances conditionnelles qui rendent l'inférence facile en pratique :

$$\begin{aligned} p(h|v) &= \prod_i p(h_i|x) \\ p(v|h) &= \prod_j p(x_j|h) \end{aligned} \tag{5}$$

Par suite on a :

$$\begin{aligned} p(h_i = 1|v) &= \sigma\left(\sum_j W_{ji}x_j + d_i\right) \\ p(x_j = 1|h) &= \sigma\left(\sum_i W_{ji}h_i + b_j\right) \end{aligned} \tag{6}$$

où σ est la fonction d'activation, et les (b, d) sont les biais. Pour autant, comme la plupart du temps, la fonction de partition reste incalculable en pratique, mais ces formules de factorisation permettent d'effectuer de l'inférence sans avoir à la calculer.

Nous en reparlerons plus en détail dans la partie d'implémentation, mais l'entraînement d'un RBM nécessite d'effectuer des étapes d'échantillonnage - de Gibbs - souvent par des méthodes MCMC. Il existe plusieurs méthodes pour accélérer le processus, nous parlerons de la *Contrastive Divergence* (CD) et du *Stochastic Maximum Likelihood* (SML).

Références

[1] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning : A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[2] Yoshua Bengio and Xavier Glorot. Understanding the difficulty of training deep feedforward neural networks. 9 :249–256, May 2010.