Note de synthèse

Dell'Aiera Clément, Prévosteau Clément, David Wahiche

Ce projet a pour objectif de présenter une introduction au réseau de neurones artificiels, et à certaines nouvelles méthodes qui sont récement appliquées en ce domaine, à savoir les méthodes de la géométrie de l'information, ainsi que l'apprentissage profond, ou deep learning.

1 Problématique

Notre but, tout au long du GT, a été de comprendre et d'implémenter des méthodes récentes d'apprentissage statistique supervisé. Depuis quelques années, certaines équipes, celle de Geoffrey Hinton en tête, ont réussi à obtenir des performances très élevées en appliquant des classes d'algorithmes répondant au nom de *deep learning* (réseaux de neurones profonds), notamment en reconnaissance de caractères manuscrits . Plus récemment, Yann Ollivier a mis en ligne au début de l'année 2013 un preprint où il applique des méthodes de géometrie de l'information sur les réseaux de neurones.

Notre objectif a été triple : étudier et implémenter des réseaux de neurones, comprendre et appliquer des idées géométriques à des algorithmes d'apprentissage, et implémenter des méthodes de réseaux profonds.

2 Synthèse des travaux

2.1 Réseaux de neurones

Les réseaux de neurones peuvent être rapidement présentés commedes modèles de régression. Ils consistent en plusieurs couches de neurones empilées les une au-dessus des autres, reliées par des connexions dont la réponse dépend d'un poids et d'une fonction d'activation. Ces derniers sont les paramètres du modèle dont l'expérimentateur va faire varier la valeur en entraînant le réseau sur une base de données via un apprentissage supervisé.

Formellement, un réseau de neurones est la donnée d'un entier N, le nombre de couches, de $N-1$ matrices $W_j,\ j = 1, N-1$, les matrices de poids, et d'autant

de fonctions d'activations f_j. Alors, si l'entrée est une donnée $x \in \mathcal{D}$, le réseau agit récursivement :

$$\begin{cases} x_0 = x \\ x_{j+1} = f_j(W_j x_j) \end{cases}$$

Dans la formule ci-dessus, la fonction d'activation est appliquée à chaque terme du vecteur $W_j x_j$. Les fonctions d'activation sont à choisir parmi celles connues des praticiens : *tanh*, sigmoïde,... Les matrices de poids constituent les paramètres du modèle.

Ces réseaux s'utilisent en apprentissage supervisé, c'est-à-dire que l'on lui fournit un entrée x, et la réponse attendue t. Le réseau compare alors l'étiquette t à la réponse y, et altère alors ses paramètres (les poids des connexions) selon un algorithme prédéfini, à savoir une descente de gradient sur une fonction de perte.

2.2 Méthodes de géométrie de l'information

C'est ici qu'intervient la géométrie de l'information. L'idée est de voir l'espace des paramètres comme une variété riemannienne, c'est-à-dire un espace muni d'un système de coordonnées et d'une métrique qui permet de donner une «longueur» à une variation des paramètres (intuitivement l'espace tangent est une variation infinitésimale de paramètres). On peut alors choisir des métriques «invariantes» ou «intrinsèques» [10] au sens où elles dépendent que de ce que fait le réseau et non de son paramétrage. Par exemple, Ollivier décide d'utiliser la métrique qui est donnée par la matrice de Fisher : elle est symmétrique définie positive, c'est donc bien en chaque paramètre du modèle une métrique. Ollivier montre qu'elle est invariante. Nous avons donc testé des descentes de gradients riemanniens et observé une convergence plus rapide.

La géométrie de l'information utilise des outils plus compliqués. Dans le livre d'Amari sur le sujet [12] figure une introduction aux connexions affines. Toutefois, bien qu'ayant commencé à les étudier, nous avons décidé de nous limiter au cadre de la descente de gradient adaptée, et ce pour plusieurs raisons. La première est que le niveau technique demandé, en géométrie notamment, est élevé, plus que ce que nous nous sentions capables de faire. Ensuite, il apparaît dans la littérature que ces méthodes plus complexes n'aboutissent pas des résultats statistiques probants, en tout cas pour le moment.

2.3 Apprentissage profond ou *deep learning*

Les méthodes de *deep learning* incorporent, en plus de réseaux de neurones à plusieurs couches, des techniques de préparations des poids des réseaux. Au moyen de se que l'on appelle des machines de Boltzman restreintes, qui sont des modèles issus de la physique statistique voisins par exemple du modèle d'Ising,

on peut entraîner chaque couche avant d'effectuer la descente de gradient.

Ces technisques sont très actuelles, et bénéficient d'un intérêt croissant de la part des entreprises qui traitent beaucoup de données. Google par exemple, qui vient de racheter une société spécialisée dans ces techniques, Deepmind.

3 Résultats

Nous avons codé une classe python entièrement à la main pour manier les réseaux de neurones à pluiseurs couches. Nous avons alors pu comparer ces méthodes avec celles plus classiques vu en cours à l'ENSAE, comme les SVM par exemple.

Nous avons aussi pu implémenter des méthodes de géométrie de l'information, ainsi que des techniques d'apprentissage profond.

Du point de vue des difficultés, nous en avons surtout rencontrées au niveau de la littérature. Le domaine est en effet très récent (certains articles datent de 2013), et il est difficile de trouver des exposés clairs et concis, ni d'harmonisation des concepts ou notations.

Les algorithmes ont été testé sur le *benchmark* de l'apprentissage : la base d'images de chiffres écrits à la main MNIST, disponible sur le site de Hinton, qui contient 70000 images, toutes au même format de 28×28 pixels. Nous avons testés les méthodes imbriquées les unes dans les autres de manière de plus en plus complexes : au départ, un simple classifieur logistique sur MNIST, puis un réseau de neurones à 2 couches dont la dernière est un classifieur logistique, enfin une préparation des données avec des machines de Boltzman, puis la même technique. Nous avons réussi à atteindre des taux d'erreurs sur l'échantillon de test de MNIST de, respectivement, $7,489\%$, $1,65\%$ et $1,34\%$. Pour comparer, le lecteur intéressé peut aller sur la page `http://yann.lecun.com/exdb/mnist/`, où il trouvera le score d'équipes de recherches.

Nous avons aussi testé les différences entre descente de gradient riemannien et descente de gradient classique, sur des exemples simulés tels que des ensembles aléatoires non linéairement séparables (XOR dans le rapport). Nous aurions voulu adapter et faire le lien entre ces techniques et le cadre riemannien mais n'avons pas aboutit.

Références

[1] G.Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4) :303–314, 1989.

[2] Yee-Whye Teh Geoffrey E.Hinton, Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7) :1527–1554, 2006.

[3] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 2(4) :251–257, 2006.

[4] Shun ichi Amari. Information geometry of the em and em algorithm for neural networks. *Neural Networks*, 8(9) :1379–1408, 1994.

[5] Shun ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2) :251–276, 1998.

[6] Patrick Kenny. Notes on boltzmann machines. 2012.

[7] Jacques Lafontaine. *Introduction aux variétés différentielles*. EDP Sciences, 1996.

[8] Herbert Lee. *Bayesian Nonparametrics via Neural Networks*. Society for industrial and applied mathematics, 2004.

[9] Seymour Papert Marvin Minsky. *Perceptrons*. MIT Press, 1969.

[10] Yann Ollivier. Riemanian metrics for neural networks. *Preprint*, 2013.

[11] Frank Rosenblatt. *Principles of neurodynamics : perceptrons and the theory of brain mechanisms*. Washington, Spartan Books, 1962.

[12] Hiroshi Nagaoka Shun-ichi Amari. *Methods of information geometry*. Oxford University Press, 1993.