Cours IFT6266,
Auto-encodeurs et architectures profondes

Auto-encodeurs

On peut entraîner des réseaux de neurones à reproduire leur entrée x, en minimisant par exemple $||f(x) - x||^2$ ou $-\log P(x; f(x))$ (avec $f(x)$ les paramètres d'une loi qui modélise x, comme la log-vraisemblance discrète quand x est discret et $f(x)$ le vecteur des probabilités pour chacune des valeurs possible).
Pour quoi faire?
Minimiser une erreur de reconstruction est une manière de construire une **variété** qui modélise les zones de haute densité de la vraie loi génératrice $P(X)$. Par exemple, si $f(x) = b + U(Vx + c)$ (réseau de neurones avec une couche cachée linéaire) avec $U \in \mathcal{R}^{d \times h}$, $V \in \mathcal{R}^{h \times d}$ et $x \in \mathcal{R}^d$, on peut modéliser une variété affine: V sera dans la base des vecteurs propres principaux de la matrice de covariance des x.
Avec un réseau à une couche cachée non-linéaire

$$f(x) = b + U \tanh(c + Vx)$$

on obtient dans $h(x) = \tanh(c+Vx)$ une transformation non-linéaire des x qui cherche à extraire des facteurs non-linéaires explicatifs des variations des x. Avec plus de couches cachées on peut obtenir des transformations beaucoup plus non-linéaires et complexes, et donc modéliser des variétés plus complexes. Plus généralement

$$f(x) = g(h(x))$$

avec $h(.)$ un réseau de neurones qui transforme x en ses coordonnées $h(x)$ sur la variété et $g(.)$ un réseau de neurone qui des coordonnées u sur la variété en coordonnées $\hat{x} = g(u)$ dans l'espace des x.
Traditionnellement on prend $\dim(h(x)) < \dim(x)$, avec cette interprétation. Quand l'auto-encodeur a beaucoup de couches on choisit les tailles des couches décroissantes dans h et croissantes dans g. On peut interpréter $h(x)$ comme une **compression** de x, qui garde le "signal" (les directions majeures de variation) et jette le "bruit" (les variations mineures et moins prévisibles).
Mais des travaux récents montrent qu'on obtient de bons résultats même avec $\dim(h(x)) > \dim(x)$, quand on utilise l'apprentissage de l'auto-encodeur comme une initialisation d'un réseau de neurones supervisé.
Ne pas apprendre la fonction identité
En théorie, si $\dim(h(x)) \geq \dim(x)$, le réseau peut apprendre une transformation $h(x)$ triviale et inintéressante qui fait que $f(x) \approx x$. Par exemple dans le cas à une couche cachée, il suffit de prendre $V = \alpha I$ et $U = \frac{1}{\alpha} I$, en laissant $\alpha \to 0$. Donc les poids de la couche cachée tendent vers 0 et ceux de la couche de sortie vers l'infini. Ainsi la tanh() opére dans sa zone linéaire où $\tanh(a) \approx a$ et $f(x) \approx x$. Pour éviter ce genre de solution, on peut soit prendre $\dim(h(x)) < \dim(x)$, ou bien s'arranger pour empêcher les poids de devenir très grands, e.g. par la régularisation ou l'utilisation d'un algorithme

d'apprentissage comme le gradient stochastique, qui ne peut pas faire des grandes modifications rapidement.

Malédiction de la dimensionalité et machines à noyau

Plusieurs résultats théoriques suggèrent fortement que les machines à noyau utilisant un noyau *local* (comme le noyau Gaussien) ne sont pas efficaces (en terme d'exemples nécessaires) pour apprendre des fonctions qui *varient beaucoup*. Voir (Bengio, Delalleau, & Le Roux, 2006) pour plus de détails.
Théorème
Suppose that the learning problem is such that in order to achieve a given error level for samples from a distribution P with a Gaussian kernel machine with predictor $f(.)$, then f must change sign at least $2k$ times along some straight line (i.e., in the case of a classifier, the decision surface must be crossed at least $2k$ times by that straight line). Then the kernel machine must have at least k bases (non-zero α_i's).

Il y a aussi des théorèmes pour des classes de fonctions spécifiques:
Théorème
Let $f(x) = b + \sum_{i=1}^{2^d} \alpha_i K_\sigma(x_i, x)$ be an affine combination of Gaussians with same width σ centered on points $x_i \in X_d$. If f solves the parity problem, then there are at least 2^{d-1} non-zero coefficients α_i.
Des résultats similaires ont été prouvés pour les algorithmes d'apprentissage semi-supervisés et non-supervisés basés sur la construction d'un graphe de voisinage (ils essaient de couvrir la variété avec ce graphe).

Architectures Profondes

La *profondeur* d'une architecture (classe de fonction) est informellement le nombre de *niveaux* d'opérations de base qui sont composées afin de produire la sortie en fonction de l'entrée. Par exemple un réseau de neurones à N couches cachées a $N + 1$ niveaux, une machine à noyau à 2 niveaux (calcul des $K(x, x_i)$ puis combinaison linéaire), etc.
Dans (Bengio & Le Cun, 2007) (disponible sur le site du cours) on discute plusieurs exemples et théorèmes qui montrent que les architectures peu profondes ne sont pas *efficaces* en terme de représentation, par rapport à des architectures profondes. Plusieurs exemples et la théorie sont basés sur les *circuits logiques*. Par exemple pour implanter la parité avec 2 niveaux il faut $O(2^d)$ portes logiques, mais il en faut seulement $O(d \log d)$ pour l'implanter avec $\log d$ niveaux. Même chose pour implanter une multiplication binaire entre deux nombres entiers à d bits chacun. On sait aussi que si on essaie d'implanter avec k niveau une fonction réalisée de manière compacte par un circuit à $k + 1$ niveau, on a généralement besoin d'un nombre exponentiel de portes logiques.
Il faudrait donc développer des algorithmes d'apprentissage non-paramétrique *qui apprennent le nombre de niveaux*, et qui permettent d'optimiser des *architectures profondes*.
Les résultats sur la malédiction de la dimensionalité et sur la profondeur des circuits suggèrent donc que pour l'apprentissage de fonctions complexes comme celles surement

nécessaires dans l'intelligence artificielle, les machines à noyau et les réseaux de neurones à une ou deux couches seront insuffisants. Une analyse similaire permet d'éliminer la plupart des algorithmes d'apprentissage actuellement en vogue...
Jusqu'à très récemment, on ne savait pas vraiment entraîner des réseaux de neurones profonds (sauf pour un type de problème très particulier, avec les réseaux à convolution), car l'entraînement des réseaux de neurones profonds semblent rester pris dans des minima locaux ou plateaux médiocres (moins bons que ceux qu'on obtient avec un réseau à une ou deux couches cachées).
Cependant un principe simple semble grandement atténuer cette difficulté d'optimisation: l'initialisation vorace couche-par-couche non-supervisée.

Initialisation vorace non-supervisée, couche-par-couche

Voir (Bengio, Lamblin, Popovici, & Larochelle, 2007) pour plus de détails. L'article et les slides de la présentation de cet article sont aussi disponibles sur le site du cours.

References

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Saul, L., Weiss, Y., & Bottou, L. (Eds.), *Advances in Neural Information Processing Systems 19*. MIT Press.

Bengio, Y., Delalleau, O., & Le Roux, N. (2006). The curse of highly variable functions for local kernel machines. In Weiss, Y., Schölkopf, B., & Platt, J. (Eds.), *Advances in Neural Information Processing Systems 18*, pp. 107–114. MIT Press, Cambridge, MA.

Bengio, Y., & Le Cun, Y. (2007). Scaling learning algorithms towards AI. In Bottou, L., Chapelle, O., DeCoste, D., & Weston, J. (Eds.), *Large Scale Kernel Machines*. MIT Press.