1-bit Compressed Sensing et Deep Networks

Joseph Lam Peter Martigny

2017

Résumé

Dans cette recherche, nous nous proposons d'explorer des propriétés théoriques dans le domaine du deep learning. Si cette partie du Machine Learning connaît de nombreux succès dans la pratique, leurs justifications théoriques peinent encore à s'imposer. Nous nous proposons ici d'étudier des propriétés de stabilité et de reconstruction de signaux, subissant des transformations via des couches successives d'un réseau de neurones.

1-Bit Compressed Sensing

I. Introduction au Compressed Sensing

Le but de plusieurs domaines de recherche dont le Compressed Sensing fait partie est la reconstruction d'un signal de grande dimension à partir d'aussi peu de données que possible. Afin de régulariser le problème, une hypothèse est faite sur la structure du signal. En particulier, en compressed sensing, on se concentre sur le traitement de signaux sparses (parcimonieux) ou compressibles, tels les images naturelles. L'idée est qu'il existe une base dans laquelle ces signaux sont représentés de manière sparse (la plupart des coefficiets sont nuls ou très proche de 0). A partir d'un ensemble petit de mesures (projections sur un petit ensemble de vecteurs), il est possible de reconstruire le signal par des méthodes gloutonnes ou un programme linéaire, trouvant la représentation la plus sparse consistante avec les mesures. La qualité de cette reconstruction dépend non seulement de la compressibilité du signal (sparsité intrinsèque), mais aussi du choix de l'algorithme de reconstruction.

Un signal représenté par un vecteur à N dimension est K-sparse si au plus K de ses coefficients sont non-nuls. Un signal est K-sparse dans une base de sparsité induite s'il peut s'écrire comme une combinaison linéaire d'au plus K élements de cette base, que l'on peut écrire $x = \sum_i \alpha_i b_i$ ou de manière compacte $x = B\alpha$. Un signal est K-compressible s'il est bien approximé par les K plus fort coefficients de cette décomposition.

On note les vecteurs de mesures : $y_i =< x, \phi_i >$, que l'on peut réécrire de manière compacte $y = \Phi x = \Phi\alpha$ où y est le vecteur des mesures et Φ est l'opérateur qui modélise le système de mesure.

Comme il sera toujours possible de se ramener à une base induite où les signaux sont sparses, on considère à partir de maintenant que les signaux étudiés le sont, ie $B = I$. Contrairement à la théorie classique de l'échantillonnage, qui demande un échantillon de taille au moins N pour pouvoir reconstruire le signal, un des résultats fondamentaux du compressed sensing et qu'il est possible de reconstruire un tel signal sparse en $M = O(Klog(N/K)) << N$ mesures, en prenant des ϕ_i iid gaussiennes centrées réduites.

Reconstruire y à partir des vecteurs de mesures consiste à trouver le signal le plus sparse possible qui soit consistant avec les vecteurs de mesures. La sparsité se mesure par la pseudo-norme l_0, qui donne le nombre de coefficients non-nuls dans la base idoine. Cependant, cette pseudo-norme n'étant pas convexe, les outils d'optimisation convexe ne s'appliquent pas et donc on va plutôt utiliser la norme l_1, à savoir $||x||_1 = \sum_i |x_i|$ qui, elle, est convexe. Cette approche est validée par le fait que la minimisation avec la norme l_1 est une bonne approximation de la minimisation avec la norme l_0. Ainsi le problème devient le suivant :

$$\hat{x} = argmin_x ||x||_1$$

t.q. $y = \Phi x$.

II. Etude du 1-bit Compressed Sensing

Modèle de mesures

Contrairement au compressed sensing classique, nous n'avons plus accès à des mesures linéaires du signal, mais à des bits individuels.

Chaque mesure représente le signe du produit scalaire entre le signal sparse et un vecteur de mesure :

$$y_i = sign(< \phi_i, x >)$$

Ainsi, on aura toujours :

$$y_i sign(< \phi_i, x >) \geq 0$$

Ce qui s'exprime de manière compacte :

$$y = sign(\Phi x)$$

et

$$Y\Phi x \geq 0$$

où Φ est la matrice représentant le système de mesures, la fonction signe utilisée élement par élement au vecteur Φx, Y est la matrice diagonale des élements de y.

Reconstruction consistante

On remarque que si x est consistant avec les mesures, alors pour $a0 \geq a < 1$ le vecteur ax est également consistant avec les mesures. Ainsi, une procédure qui demande uniquement la consistance avec les mesures obtiendra une solution nulle, ce qui n'est pas souhaitable. Pour pallier à ce problème, on se propose d'imposer le fait que le signal reconstruit appartiennent à la l_2-sphère :

$$||x||_2 = (\sum_i x_i^2)^{1/2} = 1$$

Cette contrainte diminue alors l'espace de recherche des solutions, et aide donc également à la performance de cette recherche. Ainsi le signal le plus sparse sur la sphère unité, consistant avec les mesures, est solution de :

$$\hat{x} = argmin_x ||x||_1, Y\Phi x \geq 0, ||x||_2 = 1$$

Le problème est relaxé en faisant entrer la contrainte dans la fonction d'objectif. On introduit une fonction de coût f qui est positive pour x négatif et nulle sinon, avec un coefficient de relaxation λ :

$$\hat{x} = argmin_x ||x||_1 + \lambda \sum_i f((Y\Phi x)_i), ||x|| - 2 = 1$$

En particulier, dans le cas où f est définie par :

$$f(x) = \begin{cases} x^2/2, & \text{si } x < 0. \\ 0, & \text{sinon.} \end{cases} \quad (1)$$

Cette fonction est convexe et régulière, ce qui en fait une bonne candidate pour les algorithmes de descente de gradient. En notant g la fonction de perte l_1 et $\bar{f}$ la partie de perte quadratique, la fonction de coût devient :

$$Cost(x) = g(x) + \lambda \bar{f}(Y\Phi x)$$

Algorithme de reconstruction

On utilise ici une variation de l'algorithme FPC (Fixed Point Continuation) pour la minimisation de fonctions l_1 régularisées.

Ainsi, si l'on ne contraint pas le problème à la sphère unité, le gradient de la fonction coût au minimum vaut 0, ie :

$$Cost'(x) = g'(x) + \lambda(Y\Phi)^T \bar{f}'(Y\Phi x)$$

et donc

$$g'(x)/\lambda = -(Y\Phi)^T \bar{f}'(Y\Phi x)$$

où l'on considère ici les sous-gradients :

$$g'(x)_i = \begin{cases} -1, & \text{si } x_i < 0. \\ [-1,1], & \text{si } x_i = 0. \\ +1, & \text{si } x_i > 0. \end{cases} \tag{2}$$

$$(\bar{f}'(x))_i = \begin{cases} -x_i, & \text{si } x_i \geq 0. \\ 0, & \text{si } x_i > 0. \end{cases} \tag{3}$$

Lorsque l'on introduit la contrainte de la sphère, le gradient de la fonction de coût est orthogonal à la sphère, ainsi une étape de descente de gradient suivie d'une renormalisation aboutit à un minimum sur la sphère unité comme point fixe.

Algorithm 1: Algorithms de reconstruction pour le 1-bit compressed sensing

1 Initialisation :
  $\hat{x}_0$ t.q. $||\hat{x}_0||_2 = 1$
  Pas de descente = δ
  Compteur : $k = 0$
2 Incrémentation du compteur :
  $k = k + 1$
3 Gradient de la partie quadratique :
  $\bar{f}_k = (Y\Phi)^T \bar{f}'(Y\Phi x_{k-1})$
4 Gradient projeté sur la sphère unité :
  $\tilde{f}_k = \bar{f}_k - < \bar{f}_k, x_{k-1} > x_{k-1}$
5 Descente de gradient pour la partie quadratique :
  $h = \hat{x}_{k-1} - \delta \tilde{f}_k$
6 Régularisation :
  $(u)_i = sign((h)_i) max\{|(h)_i| - \frac{\delta}{\lambda}, 0\}$
7 Normalisation :
  $\hat{x}_k = \frac{u}{||u||_2}$
8 Iteration :
  Répéter à partir de 2 jusqu'à convergence

Définition du gaussian mean width

Nous définissons tout d'abord la largeur de K dans la direction $\eta \in S^{n-1}$ la plus petite largeur du pavé contenant K entre deux hyperplans parallèles avec normales η :

$$\sup_{u \in K} < \eta, u > - \inf_{v \in K} < \eta, v > = \sup_{x \in K-K} < \eta, x >$$

où $K - K = \{x - y : x, y \in K\}$. $\omega(K)^2$ définit la dimension effective de K.

En moyennant sur les η uniformément distribués sur S^{n-1}, on obtient le spherical mean width :

$$\tilde{\omega} = \mathbb{E} \sup_{x \in K-K} < \eta, x >$$

Pour des raisons de simplicité, on utilise plutôt le gaussian mean width $\omega(K)$, où η uniforme sur S^{n-1} est remplacé par g gaussien centré réduit sur $\mathbb{R}^n$.

On montre qu'elles sont proportionnelles de toutes façons. En effet, par invariance en terme de rotation, on peut écrire $\eta = \frac{g}{||g||_2}$.

Donc $\omega(K) = \mathbb{E}||g||_2\tilde{\omega}(K)$.

Or $\mathbb{E}||g||_2 \sim \sqrt{n}$.

D'où $\omega(K) \sim \sqrt{n}\tilde{\omega}(K)$.

Si $K \subset B_2^n$, alors $\omega(K) \leq \sqrt{dim(K)}$, la dimension en algèbre linéaire. Mais il a l'avantage d'être robuste, c'est-à-dire que de petites perturbations de K mènent à uniquement des petits changements de $\omega(K)^2$.

$$\omega(K) = \mathbb{E}(\sup_{x \in K-K} | < g, x > |)$$

Nous allons ainsi nous servir de ce concept afin de traiter de la dimension de $K \subset B_2^n$.

Théorème d'encodage et de décodage du cube de Hamming

Nous considérons ici des mesures 1-bit bruitées que nous modélisons ainsi :

Soit y_i aléatoire tel que :

$$\mathbb{E}(y_i) = \theta(< a_i, x >)$$

où θ est une fonction à valeur dans $[-1, 1]$, non connue telle que $\mathbb{E}(\theta(N(0,1))N(0,1)) = \lambda > 0$ (justifié un peu plus bas). λ est une sorte de ratio signal/bruit.

Puisque l'information sur l'amplitude de x est perdue en 1-bit, on suppose sans perdre de généralité que $||x||_2 = 1$.

On a donc : $< a_i, x > \sim N(0, 1)$.

Finalement $\mathbb{E}(y_i < a_i, x >) = \mathbb{E}(\theta(N(0,1))N(0,1)) = \lambda$.

Nous pouvons alors aborder un théorème central dans la compréhension des résultats en deep learning.

Théorème 1 (Estimation uniforme, bruit adversarial) *Soit $a_1, ..., a_m$ normale centrée réduite indépendante, $\delta > 0$.*

Si $m \geq C\delta^{-6}\omega(K)^2$, alors avec probabilité au moins $1 - 8\exp(-c\delta^2 m)$:

$$||\hat{x} - x||_2^2 \leq \delta\sqrt{\log(e/\delta)} + 11\tau\sqrt{\log(e/\tau)}$$

où la distance de Hamming entre les mesures non bruitées et celles bruitées est donnée par $d_H(\tilde{y}, y) = \sum \mathbf{1}_{\{\tilde{y}_i \neq y_i\}}$, et $\hat{x}$ est la solution au problème d'optimisation (servant à estimer le signal x) :

$$\max \sum_{i=1}^{m} y_i < a_i, x' >$$

où $x' \in K$.

Le bruit adversarial correspond au bruit dans le pire cas. On peut l'interpréter comme le "flip" d'un pourcentage fixe de bits choisis arbitrairement. Ainsi, on peut le mesurer grâce à la distance de Hamming.

Puis nous mettons en contexte ce résultat. Dans [8], il est montré que $K \cap S^{n-1}$ peut être embeddé de façon presque isométrique dans le Hamming cube $\{-1, 1\}^m$:

$$|\frac{1}{\pi}d_G(x, x') - \frac{1}{m}d_H(sgn(Ax), sgn(Ax'))| \leq \delta$$

pour $x, x' \in K \cap S^{n-1}$, et d_G et d_H étant respectivement les distances géodésique dans S^{n-1} et de Hamming dans $\{-1, 1\}^m$.

Donc, l'embedding $K \cap S^{n-1} \to \{-1, 1\}^m$ est donné par le mapping : $x \to sgn(Ax)$ où on considère le signe terme à terme.

De cette façon, afin d'interpréter le résultat en termes de théorie de l'information, un signal x est encodé en une chaine binaire $y = sgn(Ax)$. Et le décodage est donné par la résolution du problème d'optimisation présenté plus haut.

Application au Deep Learning

I. Mise en contexte

L'article de Giryes montre que les réseaux de neurones devraient préserver 3 propriétés clés en classification :

— l'information principale des données d'entrée doit être préservée
— les données d'entraînement contiennent une information généralisable à d'autres données
— les points provenant de classes différentes sont traités différemment

Voici quelques résultats théoriques généraux sur les réseaux de neurones qui ont été montrés dans d'autres articles :

— Hornik et al. [6] et Cybenko [3] : les réseaux de neurones sont des approximateurs unversels de n'importe quelle fonction mesurable borélienne. Mais trouver les poids correspondants est NP-hard.
— Bruna et Mallat [2] : grâce aux wavelet scattering transform (ie des convolutions de transformations d'ondelettes en cascade avec non-linear modulus et opérateurs moyenne), ils montrent qu'avec suffisamment de couches, les features peuvent être rendues invariantes à des groupes de transformations arbitrairement complexes. Ainsi, avec une achitecture profonde, il est possible de capturer plus facilement les propriétés invariantes pour des objets et scènes dans des images.

D'autres travaux ont finalement montré que chaque couche peut potentiellement déformer les données de façon drastique. Ainsi, il est difficile de savoir si l'on peut retrouver l'input d'un network (ou d'une couche) à partir de l'output. La question ici est alors : Comment évolue la distance entre les données suite aux transformations apportées par le réseau de neurones ?

En supposant les poids gaussiens iid, on montre que les DNN préservent la structure métrique des données couche après couche, permettant de retrouver les données originelles à partir des features calculées par le réseau (tant que la taille de la sortie est proportionnelle à la dimension intrinsèque des données en entrée). Chaque couche modifie les distances proportionnellement aux angles entre les points en entrée. Plus l'angle est faible, plus les distances sont réduites. Ainsi, dans une représentation où les angles intra-classes sont petits par rapport aux angles interclasses, les distances intra-classes sont davantage réduites par rapport aux distances inter-classes.

Ainsi, il est possible de retrouver l'input de l'output jusqu'à une certaine précision. Donc, la 1e propriété clé en classification est vérifiée.

II. Stabilité de l'embedding d'une unique couche

Cette section traite des distances entre les métriques d'input et les métriques d'output d'une unique couche d'un réseau de neurones, de la forme $x \to f(Mx)$ où M est une matrice $m \times n$ à poids gaussiens aléatoires et f une fonction linéaire semi-tronquée, comme la fonction ReLU.

$f : \mathbb{R} \to \mathbb{R}$ est semi-tronquée si elle est linéaire sur un intervalle, pouvant être semi-infini, et constant en dehors, telle que $f(0) = 0$, $0 < f(x) \leq x$ pour $x > 0$. Et $0 \geq f(x) \geq x$ pour $x < 0$.

On suppose que les données en entrée appartiennent à un espace K de gaussian mean width $\omega(K)$. On peut par exemple montrer que pour K dans la boule unité de la norme l_2 constitué de L gaussiennes de dimension k, cette largeur vaut $w(K) = O(\sqrt{k + log(L)})$, alors que pour des combinaisons linéaires sparses d'atomes d'un dictionnaire, elle vaut $w(K) = O(klog(L/k))$.

Chaque couche produit un embedding stable des données au sens de Gromov-Hausdorff, ie la couche agit comme une δ-isométrie entre (K, d_K) et $(K', d_{K'})$ où K et K' sont les ensembles où vivent les input et les output, d_K et $d_{K'}$ les métriques associées à ces ensembles. On note qu'une fonction $f : K \to K'$ est une δ-isométrie si :

$$|d_{K'}(h(x), h(y)) - d_K(x, y)| \leq \delta, \forall x, y \in K$$

et

$$\forall x' \in K', \exists x \in K, d_{K'}(x', h(x)) \leq \delta$$

Nous allons appliquer le théorème d'encodage du cube de Hamming vu en 1-bit compressed sensing dans le cadre des réseaux de neurones dans le théorème 1 qui formalise la stabilité d'une unique couche d'un réseau de neurones :

Théorème 1 (Stabilité au sens de Gromov-Hausdorff) *Soit M un opérateur linéaire appliqué à une couche d'un réseau de neurones, f une activation linéaire semi-tronquée et $K \in \mathbb{S}^{n-1}$ l'ensemble où évoluent les données d'entrée pour cette couche. Si $\sqrt{m}M \in \mathbb{R}^{mxn}$ est une matrice aléatoire à poids gaussiens iid, alors l'application $g : (K \in \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \to (g(K), d_{\mathbb{H}^m})$ définie par $g(x) = sign(f(Mx))$ est, avec grande probabilité, une δ-isométrie, ie :*

$$|d_{\mathbb{S}^{n-1}}(x, y) - d_{\mathbb{H}^m}(g(x), g(y))| \leq \delta, \forall x, y \in K$$

avec

$$\delta \leq Cm^{-1/6}w^{1/3}(K)$$

Dans le théorème 1, la distance pour les données d'entrée est la distance géodésique, celle pour les sorties est la distance de Hamming. La fonction *sign* est appliquée élement par élement.

Nous avons ainsi montré que chaque couche crée une partition de l'espace. En sortie d'une couche, nous avons appliqué $f(Mx)$. Or les lignes de M forment un partitionnement de l'espace V en hyperplans. La position de x par rapport à tous ces hyperplans est récupérée de façon binaire terme à terme grâce à la fonction *sign*. Chaque cellule de la partition a un diamètre d'au plus δ d'après le théorème 1. Et il est alors possible de connaître la distance à une petite distortion près entre 2 points x, y de l'espace d'entrée simplement en connaissant le nombre d'hyperplans séparant $f(Mx)$ et $f(My)$, ce qui correspond à l'information donnée par $d_{\mathbb{H}^m}(g(x), g(y))$.

Nous n'avons ici montré que l'effet d'une unique couche. Afin d'itérer sur l'ensemble du réseau de neurones, il est nécessaire d'étendre le résultat à un embedding de S^{n-1} vers S^{m-1}. Cependant l'étude des angles et distances dans le cas d'une fonction d'activation RELU montrera que l'embedding du réseau reste stable de $\mathbb{R}^n$ vers $\mathbb{R}^m$ à chaque étape.

En résumé, chaque couche produit un embedding stable des données d'entrée. Il est donc possible de reproduire les données d'entrée à partir des données de sortie. Le théorème 2 formalise cette intuition.

Théorème 2 (Reconstruction des points en entrée à partir des sorties) *Sous les hypothèses du théorème 1, il existe un programme $\mathbb{A}$ tel que $||x - \mathbb{A}(f(Mx))||_2 \leq \epsilon$, où $\epsilon = O(\frac{w(K)}{\sqrt{m}}) = O(\delta^3)$*

Ainsi, le théorème 1 concerne les embeddings dans un cube de Hamming et le théorème 2 utilise ce fait pour montrer qu'on peut retrouver les input à partir des output.

III. Distorsion des distances et des angles

Dans cette section le papier traite des effets des couches sur les distances euclidiennes et sur les angles, en prenant la fonction ReLu comme activation. Les input étant sur la sphère $\mathbb{S}^{n-1}$, les sorties sont approximatovement sur une sphère également, de rayon moitié (d'après le théorème 3 du papier 31).

Théorème 3 (Effets sur les distances) *Soit M un opérateur linéaire appliqué à une couche donnée, ρ (ReLU) la fonction d'activation et $K \subset \mathbb{B}_1^n$ l'ensemble contenant les données d'entrée de la couche. Si $\sqrt{m}M \in \mathbb{R}^{mxn}$ est une matrice aléatoire à poids gaussiens iid et $m \geq C\delta^{-4}w(K)^4$ alors, avec forte probabilité, on a :*

$$||\rho(Mx) - \rho(My)|||_2^2 - (\frac{1}{2}||x - y||_2^2 + ||x||_2||y||_2\phi(x, y))| \leq \delta$$

où $0 \leq \theta(x, y) = cos^{-1}(\frac{x^T y}{||x||_2||y||_2}) \leq \pi$ et $\phi(x, y) = \frac{1}{\pi}(sin(\theta(x, y) - \theta(x, y)cos(\theta(x, y)))$

Théorème 4 (Effets sur les angles) *Sous les mêmes hypothèses que le théorème 3 et avec $K \subset$*

$$|cos(\theta(\rho(Mx), \rho(My))) - cos(\theta(x, y) + \phi(x, y))| \leq \frac{15\delta}{\beta^2 - 2\delta}$$

où $0 \leq \theta(x, y) = cos^{-1}(\frac{x^T y}{||x||_2||y||_2}) \leq \pi$ et $\phi(x, y) = \frac{1}{\pi}(sin(\theta(x, y) - \theta(x, y)cos(\theta(x, y)))$

On remarque que prendre $\phi = 0$ dans les théorèmes 3 et 4 revient à dire que les distances et angles sont préservées, à un facteur près de l'ordre de 1/2. En effet, on peut visualiser que le fonction ϕ est proche de la fonction $0.5(1-cos(\theta(x,y))$, ainsi pour un angle petit ϕ est minimale et donc les distances sont diminuées de moitié. Ceci peut sembler contradictoire avec le théorème 1, qui stipulait qu'une couche conservait les distances. Or le théorème 1 prend en compte la distance de Hamming, en prenant uniquement le signe de la sortie. En prenant l'amplitude, on garde une stabilité à un facteur 1/2 près. On a ainsi un corollaire aux théorèmes précédents :

Théorème 5 (Effets sur les distances euclidiennes) *Sous les mêmes hypothèses que le théorème* 3*, on a avec grande probabilité :*

$$\frac{1}{2}||x-y||_2^2 - \delta \leq ||\rho(Mx) - \rho(My)||_2^2 \leq ||x-y||_2^2 + \delta$$

Simulations

Les simulations sont réalisées sur une structure de réseaux de neurones qui offre des résultats proches de l'état de l'art en classification d'images : le VGG-net. Associée à cette structure, nous nous intéressons aux résultats données par le réseau pré-entrainé sur le dataset d'imagenet et également au réseau avec des poids aléatoires gaussiens. Le but est de comparer les distances entre les points en entrée du réseau et en sortie de certaines couches (dans notre cas, les couches convolutionnelles 8 et 16). Les distances considérées ici sont celles euclidienne et angulaire. La distance angulaire est donnée pas la cosine similarity définie comme $d_{cos}(x,y) = \frac{<x,y>}{||x||\ ||y||}$.

Pour chaque image, nous calculons la distance intra-classe, définie comme la distance avec l'image la plus éloignée de la même classe. La distance inter-classe est donnée par la distance avec l'image la plus proche d'une autre classe. En appelant x l'image en entrée, $\bar{x}$ est la sortie du réseau (tronqué à la 8e ou 16e couche) en mettant x en entrée. Ainsi en appelant d_{IC} la distance inter-classe par exemple, nous considérons :

$$d_{IC}(\bar{x})/d_{IC}(x)$$

et

$$d_{IC}(\bar{x}) - d_{IC}(x)$$

FIGURE 1 – Rapport des distances intra-classes

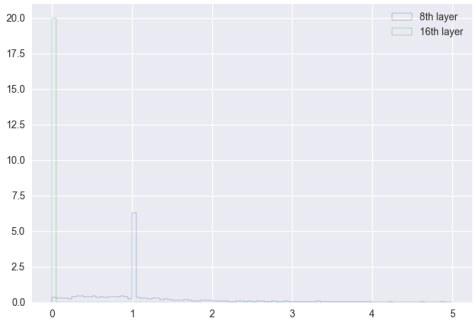

FIGURE 2 – Rapport des distances inter-classes

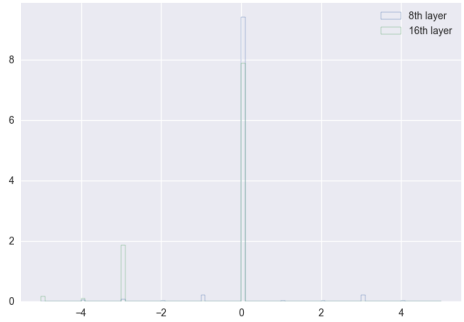

FIGURE 3 – Différence des distances intra-classes

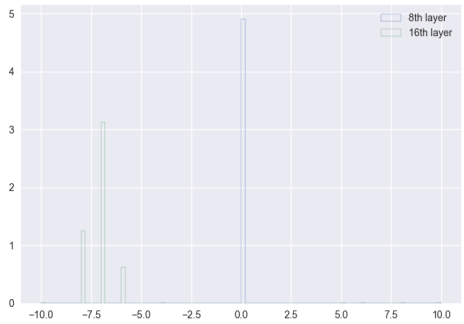

FIGURE 4 – Différence des distances inter-classes

Nous remarquons qu'à mesure que les données traversent les couches du réseau de neurones, toutes les distances se réduisent. Le rôle de l'entraînement apparaît en apprenant à réduire davantage les distances intra-classes par rapport aux distances inter-classes.

Références

[1] Petros T Boufounos and Richard G Baraniuk. 1-bit compressive sensing. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 16–21. IEEE, 2008.

[2] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1872–1886, 2013.

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

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http ://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights : A universal classification strategy. *CoRR, abs/1504.08291*, 2015.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[7] Yaniv Plan and Roman Vershynin. Robust 1-bit compressed sensing and sparse logistic regression : A convex programming approach. *IEEE Transactions on Information Theory*, 59(1) :482–494, 2013.

[8] Yaniv Plan and Roman Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete & Computational Geometry*, 51(2) :438–461, 2014.