Learning and Memorization

Théo Cornille
Mirwaisse Djanbaz
Luc Gibaud
Cédric des Lauriers

Abstract

Dans l'article *Learning and Memorization* (Chatterjee, 2018), une étude sur le lien entre généralisation et mémorisation est menée. Il est mis en lumière le fait qu'il est possible de généraliser à partir seulement de mémorisation, avec une méthode présentant des caractéristiques proches de celles des réseaux de neurones.

Dans ce rapport de projet, nous allons donc décrire les idées et le problème étudié dans l'article, présenter les résultats de nos expérimentations et de notre réimplémentation, puis analyser ces résultats et comprendre les limites du modèle étudié.

1. Introduction

Une question importante dans le domaine de l'apprentissage automatique est l'importance de la mémorisation et son lien avec la généralisation. Il a été montré par exemple que les réseaux de neurones pouvaient mémoriser totalement les données d'entraînement, même si elles sont aléatoires. Par exemple, en changeant les labels du dataset ImageNet, Zhang et al. ont réussi à obtenir un taux d'erreur durant la phase d'apprentissage presque nul (Zhang et al., 2017). Mais en plus de pouvoir mémoriser, les réseaux de neurones ont également la capacité de généraliser à de nouvelles données. Ainsi le but de Chatterjee dans son papier a été de déterminer les capacités d'apprentissage d'un modèle qui ne peut que mémoriser et voir s'il était possible de généraliser dans ce cas de figure.

2. Learning and Memorization

2.1. Lookup table

L'idée principale introduite dans le papier est le concept de "lookup table" ("lut").

Master A.I.C. 2018-2019
Cours d'apprentissage avancé - OPT6.
Sous la supervision de M. Antoine Cornuéjols.

Dans le papier, seul les fonctions binaires sont considérées (entrées décrites par k bits, sorties binaires). En posant $\mathbb{B} = \{0, 1\}$, nous considérons alors seulement les fonctions $f : \mathbb{B}^k \to \mathbb{B}$. Les données d'entraînement sont constituées de paires (x, y), x étant décrit par k bits et y étant une sortie binaire.

Ensuite, nous construisons une table avec 2^k lignes correspondant à chaque combinaison possible de k bits (1024 lignes si $k = 10$ par exemple) et deux colonnes y^0 et y^1. Pour une combinaison de bits p, la valeur à sa ligne pour la colonne y^0 (noté c_{p0}) est le nombre de fois où p est associé à la sortie 0 dans le dataset d'entraînement. De même pour la colonne y^1, mais avec la sortie 1. Nous appelons cette table un k-lut.

Ensuite, nous associons une fonction booléenne $\hat{f} : \mathbb{B}^k \to \mathbb{B}$ à notre lut de la manière suivante :

$$\hat{f}(p) = \begin{cases} 1 & c_{p1} > c_{p0} \\ 0 & c_{p1} < c_{p0} \\ random\{0,1\} & c_{p1} = c_{p0} \end{cases}$$

Cette fonction va en fait simplement associer une combinaison p à la classe la plus fréquente qui lui est associée dans le dataset, ou alors à une classe aléatoire s'il y a égalité.

Voici un exemple repris du papier original, avec un dataset, le lut et la fonction $\hat{f}$ apprise :

Table 1. Exemple d'un lut et de la fonction apprise associée

$x_0x_1x_2$	y
000	0
000	1
000	1
001	1
100	0
110	0
110	1

$x_0x_1x_2$	y^0	y^1
000	1	2
001	0	1
010	0	0
011	0	0
100	1	0
101	0	0
110	1	1
111	0	0

$x_0x_1x_2$	y
000	1
001	1
010	0*
011	1*
100	0
101	1*
110	1*
111	0*

(l'astérisque indique que la valeur a été déterminée aléatoirement)

La fonction apprise $\hat{f}$ est Bayes-optimal, c'est-à-dire qu'il n'y a pas de fonction avec une erreur plus petite sur le dataset d'entraînement.

Elle est aussi monotone, c'est-à-dire que si nous parvenions à avoir plus de bits d'information sur chacun des exemples d'entraînement, alors une nouvelle fonction $\hat{g}$ apprise sur ce dataset augmenté ne peut pas avoir une erreur à l'entraînement plus élevée.

2.2. Réseau de lookup tables

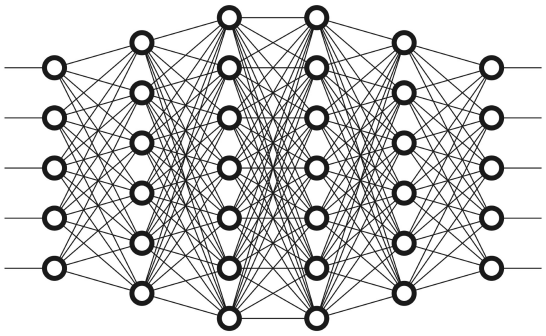

Figure 1. Réseau de neurones profond

Le principe de profondeur a prouvé son importance dans l'apprentissage des réseaux de neurones. Chatterjee a donc construit un réseau de "lookup tables" ou "luts" tels que chacun des luts est arrangé en couche successives comme un réseau de neurones.

Cependant, afin de modéliser un réseau qui ne fait que mémoriser, le modèle ne contient pas de backpropagation, ni d'algorithme d'optimisation tel que la descente du gradient.

A la place, lors de la construction du réseau, un nombre limité de luts (features pour le rapprocher du machine learning) de la couche précédente est sélectionné de manière aléatoire pour alimenter chaque lut de la nouvelle couche. Ces connexions établies aléatoirement entre couche précédente et couche suivante sont fixées et conservées ensuite lors du reste de l'apprentissage et aussi de la prédiction. Pour l'apprentissage, chaque petite lookup table est entraînée couche par couche où la sortie de la couche correspond à la sortie du finale du réseau entier : c'est à dire la valeur binaire à prédire.

L'apprentissage est réalisée par inférence : nous prédisons la valeur binaire ayant le plus grand nombre d'apparition dans toutes nos données d'entraînement.

Le réseau de look-up tables va donc apprendre pour chaque lut la valeur à prédire pour tous les bits possibles de taille k. Nous aurons donc différentes tables de stockage qui correspond aux différentes fonctions "d'activations" du réseau.

La limitation de la taille de combinaison de luts ("k" dans l'expérience) permet de contrôler la complexité de la fonction apprise par un lut.

Voici un exemple repris de l'article original avec le même dataset que l'exemple précédent qui illustre le fonctionnement d'un réseau de luts, avec les fonctions apprises :

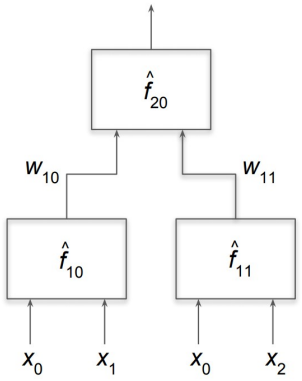

Figure 2. Réseau de luts

Table 2. Exemple de la première couche d'un réseau de lut et des fonctions apprises associées

x_0x_1	y^0	y^1	$\hat{f_{10}}$
00	1	3	1
01	0	0	1*
10	1	0	0
11	1	1	1*

x_0x_2	y^0	y^1	$\hat{f_{11}}$
00	1	2	1
01	0	1	1
10	2	1	0
11	0	0	1*

Il s'agit de la première couche, composée de 2 luts. Les bits d'entrée de chaque lut sont déterminés aléatoirement. Deux fonctions sont apprises $\hat{f_{10}}$ et $\hat{f_{11}}$. Elles nous donnent les sorties de deux luts, et par conséquent les entrées du lut de la seconde couche :

Table 3. Exemple de la couche finale d'un réseau de lut et de la fonction apprise associée

$x_0x_1x_2$	$w_{10}w_{11}$	y
000	11	0
000	11	1
000	11	1
001	11	1
100	00	0
110	10	0
110	10	1

$w_{10}w_{11}$	y^0	y^1	$\hat{f}_{20}$
00	1	0	0
01	0	0	1*
10	1	1	0*
11	1	3	1

La fonction $\hat{f}_{20}$ nous donne la sortie finale du réseau.

3. Expériences

Le dataset utilisé pour la plupart des expériences est le MNIST dont les images ont été binarisées (chaque pixel prend une valeur binaire, soit blanc soit noir), et dont les labels ont également été binarisés (0 si chiffre < 5, 1 si chiffre $>= 5$).

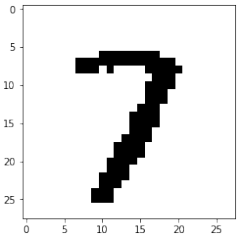

Figure 3. Image binarisée avec le label 1 (7 à l'origine)

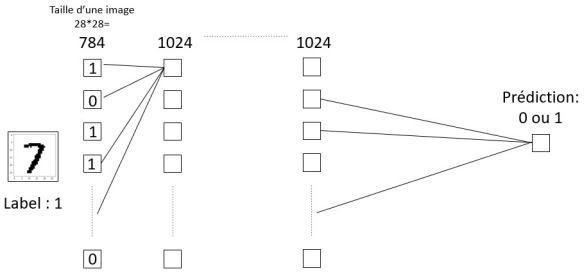

Figure 4. Architecture du réseau

3.1. Expérience 1

La première expérience consiste à utiliser un réseau de luts comme décrit ci-dessus, avec 6 couches et 8 entrées par lut ($k = 8$).

Table 4. Expérience 1 : précision lors de l'entraînement

COUCHE	NOMBRE DE LUTS	PRÉCISION
0	784	0.51076
1	1024	0.78768
2	1024	0.84706
3	1024	0.86198
4	1024	0.87266
5	1024	0.88226
6	1	0.88486

Comme il est mentionné dans l'article, plus nous augmentons la profondeur du réseau plus la mémorisation et la précision augmentent.

Cela s'explique en partie grâce à la propriété "monotone" de l'algorithme. En effet, la sortie d'un lut ne peux pas être moins précise que ses entrées.

3.2. Expérience 2

Dans cette expérience, nous faisons varier le nombre d'entrées pour chaque Lut.

Table 5. Expérience 2 : précision pour différents nombre de bits

K	PRÉCISION À L'ENTRAÎNEMENT	PRÉCISION AU TEST
2	0.621	0.6217
4	0.806	0.8123
6	0.843	0.8463
8	0.889	0.8748
10	0.945	0.8928
12	0.989	0.8886
14	0.999	0.8062
16	0.999	0.6382

Quand nous augmentons le nombre d'entrée par lut (bit/lut) le réseau mémorise mieux, nous obtenons par exemple avec $k = 14$ une précision de 1.0 sur la base d'apprentissage, cependant quand le nombre de bits par lut est trop élevé, le réseau généralise moins bien sur les données de test, la précision sur le test diminue à partir de 10 entrées par lut.

3.3. Expérience 3

Ici, nous réalisons la même expérience que celle précédente sauf que cette fois nous utilisons des labels aléatoires.

Table 6. Expérience 3 : précision pour différents nombre de bits

K	PRÉCISION À L'ENTRAÎNEMENT	PRÉCISION AU TEST
2	0.533	0.504
4	0.566	0.513
6	0.657	0.511
8	0.766	0.493
10	0.938	0.471
12	0.987	0.508
14	0.998	0.512
16	0.998	0.508

Comme précédemment la mémorisation augmente quand le nombre de bits par lut augmente. La précision sur le test varie peu. Nous pouvons constater qu'il faut plus de bit par lut pour mémoriser avec des données aléatoires qu'avec des données réelles, en se référant à la précision sur l'entraînement (expérience 2). Les données réelles sont donc plus facilement apprenables car corrélées.

3.4. Expérience 4

L'expérience 4 consiste à comparer différents modèles d'apprentissage dont les k plus proches voisins ou encore les arbres de décisions aléatoires, régression logistique, bayes naif.

Table 7. Expérience 4 : comparaison entre différents modèles

NOM DU MODÈLE	ENTRAÎNEMENT	TEST
5-NEAREST NEIGHBORS	0.984	0.98
1-NEAREST NEIGHBORS	1.0	0.975
RANDOM FOREST (10 TREES)	0.998	0.961
MEMORIZATION	0.881	0.87
LOGISTIC REGRESSION	0.865	0.872
NAIVE BAYES	0.673	0.68
RANDOM GUESS	0.4978	0.4939

Nous pouvons voir que le modèle mémorisation pour k=12 performe aussi bien que la régression logistique sur les données de test et mieux que naive bayes. Cependant des modèles simples comme les k plus proches voisins et arbres de décision arrivent à faire mieux que le modèle atteignant presque une précision de 1 sur le test set. Le modèle de mémorisation peut donc être envisagé comme un nouveau modèle d'apprentissage suite à ses performances très correctes sur le dataset bien connu MNIST. Le modèle sera comparé dans la suite sur le dataset CIFAR-10.

3.5. Expérience 5

Nous ramenons maintenant le problème à une classification binaire. Il y a donc 2 parmi 10, soit 45 classifications binaires à réaliser.

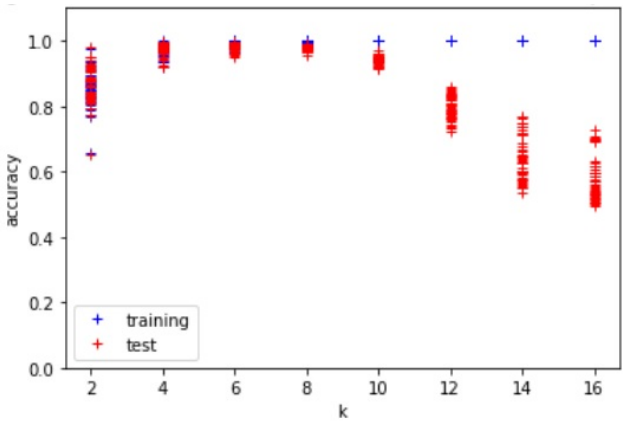

Figure 5. $45 * 8$ paires de précision d'entrainement et test en fonction de k pour Pairwise-MNIST

Globalement les résultats sont supérieurs à la chance (> 0.5). Nous obtenons comme dans le papier de meilleurs résultats sur le test à $k = 8$ et nous votons qu'à partir de $k = 8$, nous avons du mal à généraliser.

3.6. Expérience 6

Maintenant, nous testons l'impact du nombre de couche lorsque que nous avons une taille de lut faible ($k = 2$). Nous réalisons comme dans l'expérience 5, 45 classifications binaires mais en ne variant que la profondeur.

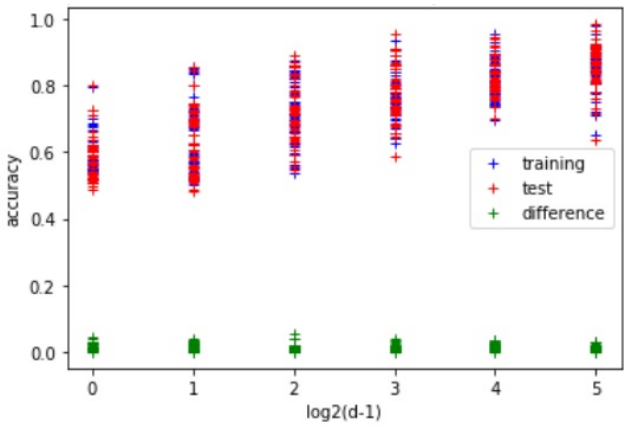

Figure 6. Effet de la profondeur sur le réseau de tables avec $k = 2$ pour Pairwise-MNIST

Nous pouvons voir que la mémorisation / précision augmente sur le train et le test set. De même, la différence entre le train et le test set sur l'erreur de généralisation est réduite.

3.7. Expérience 7

Dans cette partie, nous reproduisons les expériences de mémorisation sur le dataset d'images à 10 classes : CIFAR 10.

Nous utilisons un réseau à 5 couches, contenant chacune 1024 luts et avec $k = 10$. Au préalable, nous binarisons CIFAR10. La fonction à apprendre sera donc $f : \mathbb{B}^{3*32*32} \rightarrow \mathbb{B}$ à partir des 50000 images de la base.

L'algorithme de mémorisation réalise un score supérieur à celui de la chance ($0.556 > 0.5$).

3.8. Expérience 8

Nous reproduisons l'expérience avec la même architecture d'algorithme de mémorisation choisit dans l'expérience 7 sur CIFAR-10, sauf que nous transformons le problème en 45 classifications binaires comme dans l'expérience 5.

Les résultats sont meilleurs que la chance en moyenne sur l'ensemble des classifications binaires. Cependant, en reprenant la même architecture que dans l'expérience

Table 8. Expérience 7 : comparaison entre différents modèles

NOM DU MODÈLE	ENTRAÎNEMENT	TEST
CNN	0.69	0.5935
RANDOM FOREST (300 TREES)	0.996	0.871
5-NEAREST NEIGHBORS	0.96	0.92
1-NEAREST NEIGHBORS	1.0	0.93
MEMORIZATION	1.0	0.556
LOGISTIC REGRESSION	0.958	0.809
NAIVE BAYES	0.666	0.687
RANDOM GUESS	0.492	0.498

Table 9. Expérience 8 : précision pour différents nombre de bits

J	PRÉCISION À L'ENTRAÎNEMENT	PRÉCISION AU TEST
1	0.566	0.513
2	0.657	0.511
3	0.766	0.493
4	0.766	0.493
5	0.766	0.493
6	0.766	0.493

7, le modèle mémorisation n'arrive pas à apprendre les classes 2 à 2. Nous sommes sensés obtenir une précision à l'entraînement supérieure à 0.95 et une précision sur le test de plus de 0.65.

3.9. Expérience 9

Dans cette expérience, nous faisons apprendre à l'algorithme de mémorisation une frontière de décision (en forme de cercle) à partir de points dont les positions sont encodés aléatoirement sous forme de nombre à 10 bits.

La fonction à apprendre est $f : \mathbb{B}^{20} \to \mathbb{B}$ (10 pour la position x et 10 pour la position y). Le dataset est équitablement répartie : même nombre de 0 et de 1 pour le train et le test set.

Pour cette expérience, nous utilisons un réseau à 32 couches de 2048 luts. Nous faisons varier k de 2 à 10 afin d'étudier l'apprentissage du modèle (voir image de droite (figure 7)).

Comme précédemment mais de manière plus visuelle, nous pouvons constater la mémorisation sur le train set qui se fait plus rapidement que la généralisation sur le test set. Le concept de forme cercle est bien appris. La mémorisation et la généralisation est optimale pour $k = 6$ et bous voyons que pour k=10, la précision sur le test set chute. Nous avons déjà abordé ce problème dans l'expérience 2 lorsque

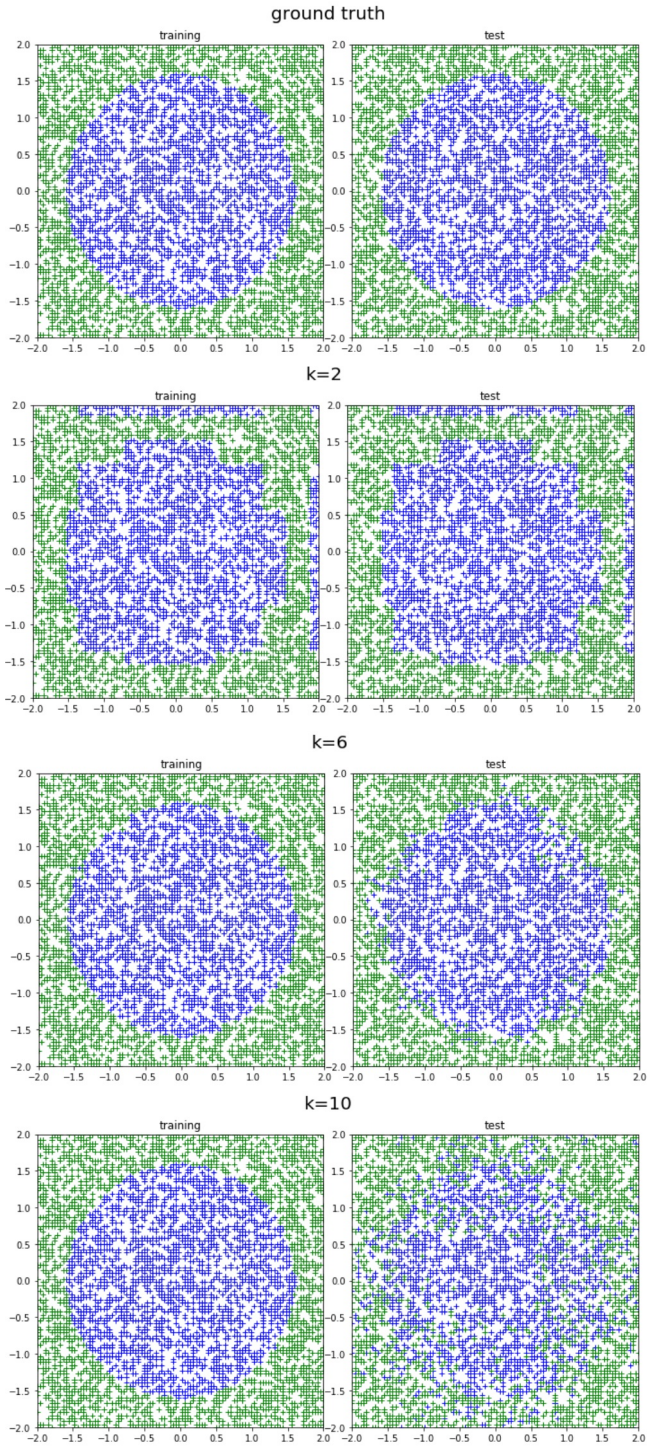

Figure 7. Frontières de décision apprises

l'objectif était d'étudier l'influence de k sur le modèle.

Le concept de généralisation dépend donc de plusieurs hyper-paramètres:

- la taille de combinaison de lut (k)

- le nombre de look-up tables ou profondeur (d)

- la taille de chaque look-up table (principalement fixé à 1024 pour chacune des expériences réalisées)

Ces hyper-paramètres quand ils choisis avec attention peuvent mener à une bonne généralisation.

4. Amélioration du modèle

Le temps de calcul du modèle - à la fois pour l'entraînement et l'utilisation - peut être réduit. En effet, l'unique lut de la dernière couche ne pourra prendre en entrée que 8 luts de l'avant dernière couche. Ainsi, le calcul des 1016 autres luts est inutile. De même, au maximum, seuls 64 luts de la couche précédente ne pourront être utilisés. Avec un raisonnement similaire, nous pouvons également réduire les luts utiles des couches précédentes.

Dans le pire des cas, nous pouvons donc limiter le calcul à 2633 $(1024+1024+512+64+8+1)$ luts au lieu de 5121 $(5*1024+1)$ luts sans modifier en rien le fonctionnement de l'algorithme. Le temps de calcul est donc approximativement divisé par deux.

Cette méthode est évoquée dans l'article mais l'auteur a décidé de ne pas en tenir compte car les calculs qu'ils effectuent sont déjà très rapides. Ce n'est pas ce que nous avons pu constater en implémentant le modèle. Il y a donc deux explications probables :

- les ordinateurs que nous avons à notre disposition sont peu puissants en comparaison de ceux utilisés par l'auteur de l'article
- notre implémentation est plus coûteuse en temps de calcul que celle effectuée par l'auteur de l'article.

Nous avons mis en place cette méthode, les résultats d'une expérience sont présentés dans la table 10.

Table 10. Précision lors de l'entraînement

COUCHE	LUTS	LUTS CALCULÉS	PRÉCISION
0	784	784	0.50962
1	1024	1024	0.78028
2	1024	985	0.83979
3	1024	418	0.86093
4	1024	64	0.87126
5	1024	8	0.88729
6	1	1	0.88952

5. Analyse et limites du modèle

L'objectif de cette partie est, au delà de ses apparentes bonnes performances, de discuter les limites du modèle de mémorisation. En outre, nous tenterons de comprendre l'utilité qu'il est possible d'espérer de ce modèle.

5.1. Classification multi-classes

Il est à noter que si le modèle peut effectivement être utilisé pour faire de la classification multi-classes - comme nous avons pu le constater avec l'expérience 5 - cela nécessite a priori de transformer le problème en problème binaire, en utilisant des méthodes *un contre un* ou bien *un contre tous*. En effet, généraliser la méthode à la classification directe de problèmes multi-classes est en théorie possible. Pour ce faire, il faudrait utiliser, à la place des valeurs binaires pour chaque lut, des valeurs en base n pour faire de la classification de n classes distinctes. Ce n'est cependant pas utilisable en pratique : si nous reprenons l'exemple du MNIST et que nous voulons classifier correctement chaque image, nous passerons de 2^8 possibilités pour chaque lut pour la classification binaire à 10^8 pour la classification en 10 classes distinctes (en gardant k = 8). Cela nécessitera non seulement un ordinateur très puissant, mais aussi une base de données de taille bien supérieure si nous voulons éviter que chaque valeur prise pour un lut ne concerne qu'un nombre très faible d'images du jeu d'entraînement, dans quel cas, la généralisation ne serait pas possible.

5.2. Inconvénients du modèle

Nous pouvons noter qu'aucune généralisation aux problèmes de régression n'est proposée contrairement à la plupart des algorithmes de machine learning. Par exemple, même les forêts aléatoires, qui sont des algorithmes pensés pour des tâches de classification, peuvent être utilisés pour des problèmes de régression.

Par ailleurs, les jeux de donnés utilisés dans ces expérimentations sont des données structurées et relativement simples, par exemple le MNIST est un problème de classification pour lequel de nombreux algorithmes obtiennent des résultats satisfaisant. Il n'est pas démontré que la mémorisation fonctionne aussi bien pour des problèmes non structurés ou plus complexes.

Bien que le modèle fonctionne donc assez bien, il n'est pas démontré qu'il existe un type de problème pour lequel ses performances surpassent - en rapidité de calcul ou en en précision - d'autres modèles, notamment les réseaux de neurones.

Un dernier point que nous avons relevé est que le temps de calcul qui semble bien plus important que ce que suggère (Chatterjee, 2018). Outre la plus importante puissance de calcul à disposition de l'auteur, il est possible que notre implémentions soit mois performante. La procédure de mémorisation du modèle est proportionnel à la taille des données d'entraînement. En effet, nous calcu-

lons tous les luts d'une nouvelle couche à partir des données d'entraînement. La profondeur du réseau et la taille de chaque couche augmente considérablement le temps de calcul. Cependant, l'auteur affirme que puisque chaque lut est indépendant dans une couche, le modèle serait parallélisable et permettrait ainsi d'améliorer grandement le temps d'apprentissage. Ainsi, ce modèle assimilé finalement à un modèle de compte pourrait être effectué entre plusieurs machines sur des sous-sets de données et les résultats seraient ainsi agrégés.

5.3. Avantages du modèle

Malgré les inconvénients présentés précédemment, le modèle étudié n'est cependant pas dénué d'avantages.

Pour commencer, à l'instar des arbres de décisions, il est très simple d'expliquer un résultat produit par l'algorithme, ce qui est l'un des actuels soucis majeurs de certains algorithmes de machine learning, les réseaux de neurones profonds notamment. Cette transparence est très importante dans certaines applications de l'intelligence artificielle telles que le domaine judiciaire par exemple.

En outre, le modèle de mémorisation présente de nombreuses similarités avec les réseaux de neurones profonds : l'architecture est en forme de couches successives et les luts ont plusieurs similarités avec les neurones artificiels. Les neurones/luts prennent en entrée une valeur provenant de plusieurs neurones/luts de la couche précédente et émettent en sortie une valeur déterministe. Nous en déduisons donc que l'étude du modèle de mémorisation pourrait nous permettre de mieux comprendre les réseaux de neurones profonds.

6. Conclusion

Les résultats que nous obtenons semblent très proches de ceux obtenus par (Chatterjee, 2018) et le modèle que nous avons implémenté permet bien de généraliser à de nouveaux exemples une tâche, uniquement à partir de la mémorisation d'exemples. Nous avons cependant soulevé une différence avec les résultats de l'auteur : le temps de calcul qui semble bien plus important que ce que suggère l'article. Un avantage notable par rapport aux réseaux de neurones est que nous pouvons comprendre à quoi correspondent la valeur de chaque lut, alors que les neurones des couches intermédiaires d'un réseau de deep learning correspondent à des représentations abstraites qui échappent le plus souvent à notre compréhension.

References

Chatterjee, Satrajit. Learning and memorization. In Dy, Jennifer and Krause, Andreas (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 755–763, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/chatterjee18a.html.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. 2017. URL https://arxiv.org/abs/1611.03530.