Résistance aux attaques et corrections adversaires

Résumé

On introduit la notion de *résistance*, qui quantifie la difficulté à tromper un réseau de neurones classificateur avec un exemple adversaire créé à partir d'une image donnée.

Ce concept offre trois apports intéressants :

- une méthode de diminution de l'erreur commise par un réseau donné,
- une méthode de détection des exemples adversaires,
- Une quantification de la certitude d'une prédiction.

1. Les attaques adversaires

1.1 Les exemples adversaires

Les réseaux de neurones sont notoirement vulnérables aux attaques par *exemples adversaires* [1] : il s'agit d'entrées imperceptiblement perturbées pour induire en erreur un réseau classificateur.

Plus concrètement, en considérant $Pred$ la fonction qui à une image associe la catégorie prédite par réseau ; et en considérant une image img de $[0,1]^n$ (c'est à dire à n pixels), on cherche une perturbation r de norme minimale telle que :

$$\begin{cases} img + r \in [0,1]^n \\ Pred(img + r) \neq Pred(img) \end{cases}$$

Dans toute la suite, on utilisera la norme euclidienne. D'autres normes sont évidemment possibles, mais sans amélioration sensible des résultats.

1.2 Les attaques adversaires

On cherche un algorithme qui détermine un exemple adversaire à partir d'une image donnée. On dit qu'un tel algorithme réalise une *attaque adversaire*.

Une méthode d'attaque possible est la suivante. Introduisons $Conf_c$ la fonction qui à une image associe la probabilité (selon le réseau) que l'image appartienne à la catégorie c ; et soit une image img de catégorie c. On cherche alors à minimiser par descente de gradient la fonction $Loss_1$ suivante :

$$Loss_1 = \begin{cases} \|r\| & \text{si } Conf_c(img + r) \leq 0.2 \\ Conf_c(img + r) + \|r\| & \text{sinon.} \end{cases}$$

Cette première fonction est expérimentalement peu satisfaisante, car l'attaque échoue souvent. La perturbation reste "bloquée" en 0, et n'évolue pas. Pour pallier cela, on oblige la perturbation à grossir en ajoutant un troisième cas de figure, quand $Conf_c(img + r) > 0.9$, c'est à dire quand la perturbation n'est pas du tout satisfaisante :

$$Loss_2 = \begin{cases} \|r\| & \text{si } Conf_c(img + r) \leq 0.2 \\ Conf_c(img + r) + \|r\| & \text{si } Conf_c(img + r) \leq 0.9 \\ Conf_c(img + r) - \|r\| & \text{sinon.} \end{cases}$$

Cette deuxième fonction produit presque toujours un exemple adversaire pour un nombre d'étapes de descente de gradient suffisamment élevé (généralement 200 étapes suffisent), et c'est celle-ci qui sera utilisée par la suite.

La Figure 1 montre le résultat d'une attaque adversaire : à gauche l'image originale, au milieu la perturbation et à droite l'image adversaire.

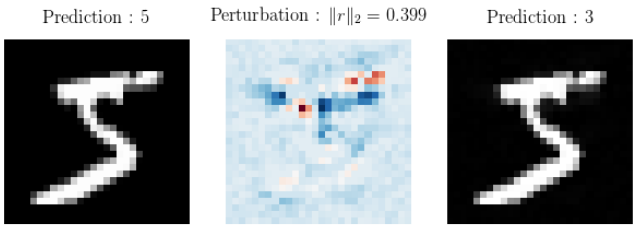

Figure 1: Résultat d'une attaque adversaire

On appellera $Pert_N$ la fonction qui à une image associe la perturbation obtenue après N étapes de descente de gradient (avec un taux d'apprentissage $\eta = 10^{-3}$).

1.3 Quelques résultats

Considérons un réseau de type `AlexNet` (CNN avec Dropout) [2, 3], appliqué au problème de la classification des chiffres manuscrits de `MNIST`. Ce réseau est entraîné avec 50000 images, et sa performance évaluée sur les 10000 autres images de test. Les 10000 images de validation ne sont pas utilisées pour le moment.

On réalise l'attaque adversaire sur les images de test, en effectuant 500 étapes de descente du gradient de $Loss_2$, avec un taux d'apprentissage $\eta = 10^{-3}$, et on s'intéresse aux valeurs prises par $\|r\|$ et $Conf_c$ au cours de l'attaque.

La Figure 2 a été obtenue en attaquant la première image de test de `MNIST`.

Qualitativement, la norme de la perturbation augmente jusqu'à ce que $Conf_c$ passe en dessous de 0.9, à partir de quoi la norme diminue en gardant une valeur de $Conf_c$ stabilisée autour de 0.2.

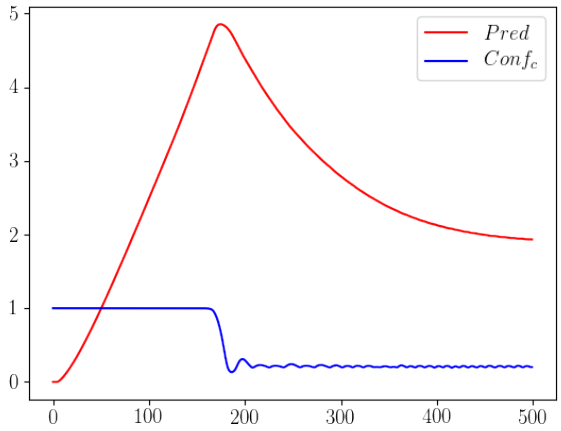

Figure 2: Attaque adversaire “difficile”

Cette image peut être qualifiée de “difficile à attaquer” : il a été nécessaire d'augmenter très fortement la norme de la perturbation pour réussir à casser la prédiction du réseau, ce qui ne se produit qu'après un grand nombre d'étapes, et la norme finale de la perturbation est élevée.

En attaquant une autre image, on a obtenu la Figure 3. Cette image peut alors au contraire être qualifiée de “facile à attaquer” : bien moins d'étapes ont été nécessaires pour casser la prédiction du réseau, la norme finale est très basse, et il n'y a pas eu de pic.

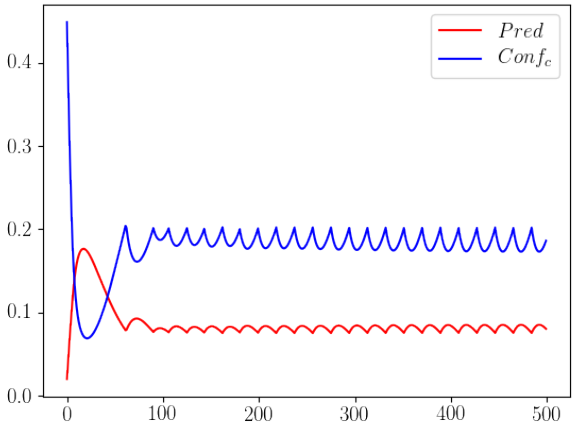

Figure 3: Attaque adversaire “facile”

On voit nettement ici l'influence de la valeur du seuil à 0.2 dans la fonction *Loss*. Dès que $Conf_c$ est en dessous de 0.2, l'algorithme a pour seul objectif de réduire la norme de la perturbation, et fatalement $Conf_c$ repasse au dessus de 0.2. Il s'agit alors de réduire à la fois $\|r\|$ et $Conf_c$, jusqu'à ce que $Conf_c$ repasse en dessous de 0.2...

D'autres exemples d'attaques d'images "faciles" ou "difficiles" à attaquer sont présentés dans l'Annexe A.

Résumons les différences principales entre ces deux types d'images :

	Images "faciles"	Images "difficiles"
Pic	Absent ou faible	Haut
Étapes nécessaires	moins de 50	plus de 200
Norme de r finale	Faible	Élevée

Pour quantifier plus précisément cette difficulté à attaquer une image, introduisons le concept de *résistance*.

2. La résistance Res

2.1 La résistance à une attaque

Pour chaque image, on essaie de quantifier la résistance, du réseau à une attaque adversaire. Plusieurs définitions sont possibles, par exemple la norme de la perturbation minimale mettant en échec le réseau :

$$Res_\infty(img) = min\{\|r\| \ ; \ Pred(img + r) \neq Pred(img)\}$$

Cette expression de la résistance n'est que d'un faible intérêt en pratique, car incalculable. C'est pourquoi on utilisera plutôt les trois définitions suivantes :

- La norme finale obtenue après un certain nombre d'étapes dans l'attaque adversaire :

$$Res_N(img) = \|Pert_N(img)\|$$

- La hauteur du pic de la norme de la perturbation :

$$Res_{max}(img) = max\{\|Pert_N(img)\| \ ; \ N \in \mathbb{N}\}$$

- Le nombre d'étapes qu'il a fallu pour abaisser $Conf_c$ à 0.2 :

$$Res_{min}(img) = min\{N \in \mathbb{N} \ ; \ Conf_c(Pert_N(img)) < 0.2\}$$

Le calcul pratique de ces trois valeurs est explicité dans l'Annexe B.

2.2 Une corrélation avec la fiabilité de la prédiction

Les images attaquées dans l'Annexe A n'ont pas été choisies au hasard : celles du haut sont les 6 premières de la base de donnée de test, (classifiées correctement par le réseau) , et celles du bas correspondent aux 6 premières erreurs de classification commises par le réseau.

En utilisant toujours le réseau classificateur précédent, sur les 10000 images de test, toutes sauf 89 sont classifiées correctement.

Étudions la répartition des valeurs de la résistance, d'abord sur 250 images correctement classifiées (notées **V**), puis sur les 89 incorrectement classifiées (notées **F**) :

	Res_N	Res_{max}	Res_{min}
Premier décile (**V**)	0.90	1.89	83
Dernier décile (**F**)	0.75	1.09	58

Dans les trois cas, le premier décile des images correctement classifiées est inférieur au dernier décile des images sur lesquelles le réseau se trompe ! Une corrélation se dessine donc nettement entre la résistance et la justesse de la prédiction du réseau.

3. Les corrections adversaires

On observe un autre phénomène : si une attaque adversaire cherche à tromper le réseau, une attaque adversaire sur une image incorrectement classifiée va, le plus souvent, produire une image qui sera correctement classifiée ! On parlera alors de *correction adversaire*.

Toujours avec le même réseau, sur les 89 erreurs commises, 80 des corrections adversaires donnent la bonne catégorie, soit dans 90% des cas.

4. Une méthode pour réduire l'erreur du réseau ?

4.1 Un premier résultat

Exploitons les deux phénomènes précédents pour tenter de réduire l'erreur commise par le réseau : On on détermine la résistance de chaque image du réseau. Si la résistance est supérieure à un certain critère, on considérera que la prédiction du réseau est correcte ; sinon on choisit comme prédiction le résultat de la contre-attaque adversaire.

Sur un lot de 275 images (250 justes, 25 erreurs), avec respectivement $Res_{N=500}$, Res_{min} et Res_{max}, on obtient le nombre d'erreurs commises en fonction du critère choisi, Figure 4.

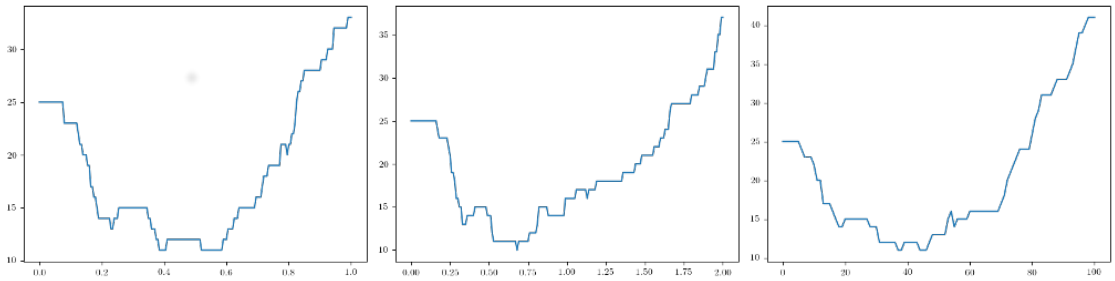

Figure 4: Nombre d'erreurs en fonction du critère choisi

Avec des critères à 0, on retrouve naturellement 25 erreurs, puisque l'on n'a rien modifié aux prédictions du réseau.

En revanche, avec des critère respectivement à 0.4, 0.685 et 45, le réseau ne commet plus que respectivement 11, 10 et 11 erreurs.

4.2 Une méthode qui se généralise difficilement

Pour évaluer la généralisation de ces méthodes, on applique les critères optimaux trouvés sur les images de test de `MNIST` aux images de validation `MNIST`, sur lesquelles on n'a toujours pas travaillé, que ce soit pour l'entraînement du réseau ou la détermination du critère.

En utilisant les trois critères précédents à l'ensemble de 10000 images de validation, on ne réussit qu'à faire passer le nombre d'erreurs de X à X dans le meilleur des cas. Ceci s'explique simplement : le nombre d'erreurs initial est proportionnellement trop faible (moins de 1%), et donc les faux positifs, même si peu fréquents, vont annuler tout le gain obtenu.

Le choix arbitraire d'une fonction de résistance et d'un critère fixé n'est donc pas une méthode efficace dans ce cas.

4.3 Affinage de la méthode précédente

Le choix arbitraire d'un critère est donc une méthode peu efficace. Essayons alors d'affiner la distinction entre les images correctement ou incorrectement prédites : on s'intéresse maintenant à toutes les valeurs prises par $\|r\|$ et $Conf_c$ au cours d'une attaque, et on cherche à entraîner un réseau de neurones à faire cette distinction, à partir des images de test.

(à continuer)

5. Une méthode pour détecter les exemples adversaires

(à continuer)

Bibliographie

[1] C. Szegedy, I. Goodfellow & al. CoRR, **Intriguing Properties of Neural Networks.** (Déc. 2013)

[2] A. Krizhevsky, I. Sutskever & G. Hinton. NIPS'12 Proceedings, **ImageNet Classification with Deep Convolutional Neural Networks .** Volume 1 (2012), Pages 1097-1105

[3] N. Srivastava, G. Hinton, A. Krizhevsky & al. JMLP, **Dropout: A Simple Way to Prevent Neural Networks from Overfitting.** Volume 15 (2014), Pages 1929-1958

[4] Trouver un papier qui propose des vraies méthodes d'attaques

A. Un peu plus de résultats

En Figure 5, les valeurs prises par $\|r\|$ et $Conf_c$ au cours de l'attaque de 6 images "faciles" à attaquer.

Attaques adversaires "difficiles"

En Figure 6, même chose avec 6 images "difficiles" à attaquer.

Attaques adversaires "difficiles"

B. Calcul pratique de $Res_{N=500}$, Res_{min} et Res_{max}