Vulgarisation: Détection d'anomalies basée sur l'Energie

Martin BENITO-RODRIGUEZ

1 Introduction

Les IA sont de plus en plus présentes dans nos vies et font parfois des erreurs qui peuvent se révéler coûteuses, mais il existe des solution pour prévenir ces erreurs. L'article [2] propose un modèle basé sur l'énergie pour une meilleur détection des anomalies dans les réseaux de neurones. Nous allons voir ensemble ce qu'est un réseau de neurones, la fonction softmax, les anomalies, les fonctions de coût et l'utilisation de l'énergie.

2 Qu'est ce que le deep learning ?

Le deep learning est une technologie appartenant au domaine de l'apprentissage automatique qui à la particularité d'utiliser un réseau de neurones pour faire de nombreux calculs.

Un réseau de neurones se décompose en plusieurs couches successives, une couche étant un ensemble de neurones.

La première couche correspond à l'input de notre réseau, la dernière couche correspond à l'output, entre les deux, il y a plusieurs couches que l'on appelle des "couches cachées". Dans un probleme de classification, chaque outputs correspond à une classe, le réseau de neurones doit associer à chaque classe une probabilité.

La valeur d'un neurone est déterminée par la valeur des neurones de la couches prédécentes. Dans le réseau de neurones le plus classique, le réseau de neurones à propagation avant (Feedforward neural network en anglais), chaque neurone de la couches i est connecté avec tous les neurones de la couche $i-1$ via des connecteurs qu'on appelle synapses. Ainsi les couches successives de l'input à l'output verront leurs neurones s'activer.

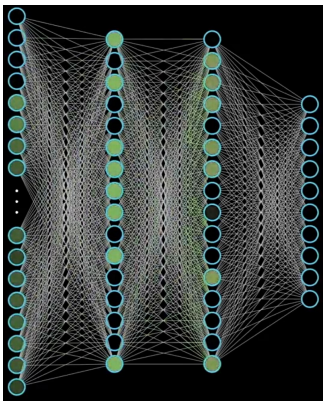

Figure 1: Un réseau de neurones en plein travail.
But what is a Neural Network? Deep learning, chapter 1 — 3Blue1Brown

Pour toute synapse, on assigne un *poids* qui joue le role de coefficient sur la valeur du neurone de la couche précédente. Pour connaitre la valeur d'un neurone, il faut calculer la somme de chaque neurone de la couche précédente pondéré avec le poids de sa synapse. On peut aussi ajouter un *biais* a chaque couche qui se manifeste comme une addition qui permet de corriger la valeur.

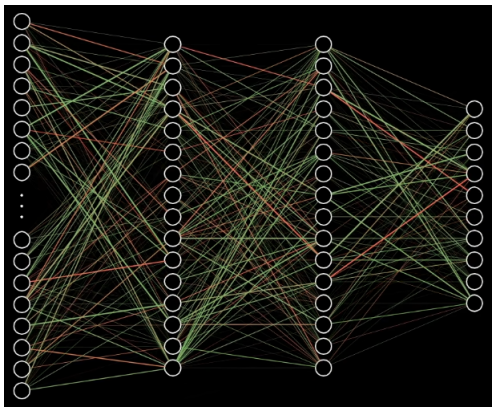

Figure 2: Un réseau de neurones pondéré, plus la synapse est visible, plus le poids est important.
But what is a Neural Network? Deep learning, chapter 1 — 3Blue1Brown

Puisque chaque neurone contient un nombre, et que chaque couche contient des neurones, on peut représenter une couche comme un vecteur de nombres. Aussi, puisque chaque neurone dépend de la couche précédentes, on peut représenter chaque neurone comme une fonction qui prend en paramètre un vecteur et fait des calculs avec le vecteur, les poids et le biais pour sortir une valeur.

Au final, on peut représenter le réseau de neurones comme une fonction qui prend en input un vecteur et des paramètres de poids et biais, et donne en output un vecteur (la taille du vecteur correspond au nombre de classes).

L'entrainement consiste à chercher quel sont les poids et les biais les plus optimaux pour faire des prédictions exacts. Au début de l'entrainement, les poids et les biais ont des valeurs aléatoires, le réseau de neurones fera des premières prédictions probablement erronées. Pour lui faire comprendre cela, nous avons besoin d'une fonction de coût qui mesure la différence entre la prédiction et la réalité.

A chaque prédiction, le réseau de neurones va alors modifier ses poids et ses biais afin de minimiser la fonction de coût et donc faire des prédictions de plus en plus précises. [4]

Pour que la dernière couche soit facilement appréhendable par un humain, la valeur de chacun de ses neurones peut représenter une probabilité d'appartenance à une classe. Un des outils pour établir ces probabilités est la fonction softmax.

3 Softmax

La fonction softmax (appelée distribution de Boltzman en physique ou distribution de Gibbs en mathématiques) est utilisée notamment en dernière couche d'un réseau de neurones. Elle prend en paramètre un vecteur $z = (z_1, ..., z_K) \in \mathbb{R}^k$ et le normalise en distribution de probabilité avec K probabilités proportionnelles à l'exponentiel du nombre entré. Les K probabilités représentent les probabilités des K classes d'etre associé à l'input.

$$\sigma(z)_i = \frac{e^{z_i/T}}{\sum_{j=1}^{K} e^{z_j/T}} \tag{1}$$

La normalisation permet à la somme du vecteur de sortie d'etre égale à 1. On peut ainsi voir quel proportion a chaque valeur de e^{z_i} dans la somme.

Le paramètre température $T \in \mathbb{R}^+$ permet d'applatir la courbe, plus T est élevé, plus la distribution est plate. D'un point de vue physique, en comparant la fonction softmax avec la distribution de Boltzmann, on remarque que la température T de softmax ne correspond pas à une température mais à une energie. On peut voir l'applatissement de la courbe en fonction de T en prenant les valeurs extremes.

$$\lim_{T \to 0} \frac{e^{z_i/T}}{\sum_{j=1}^{K} e^{z_j/T}} = argmax(z)$$

$$\lim_{T \to +\infty} \frac{e^{z_i/T}}{\sum_{j=1}^{K} e^{z_j/T}} = \frac{1}{\sum_{j=1}^{K} 1} = \frac{1}{K}$$

Quand T tend vers 0, une seule classe a une probabilité de 1 et toute les autres ont une probabilité nulle, alors que quand T tend vers $+\infty$, la distribution est uniforme.

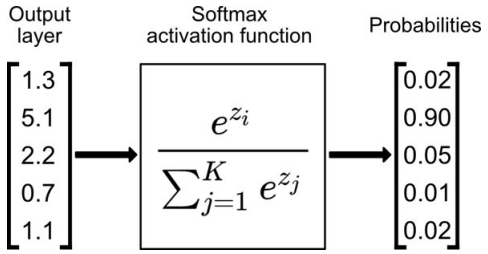

Figure 3: Exemple de softmax avec T=1.
https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

On utilise une exponentielle pour exagérer les différences. La fonction retournera une valeur proche de 0 quand z_i est largement inférieur aux autres valeurs, et elle retournera une valeur proche de 1 si elle est largement supérieur aux autres valeurs (voir Figure 3). Cela permet de construire une moyenne pondérée qui agit comme une fonction lisse.

La fonction softmax est utilisée pour normaliser le résultat d'un réseau de neurones, son input sera un vecteur correspondant à la dernière couche du réseau de neurone. Puisqu'on peut représenter le réseau de neurones comme une fonction on peut écrire la fonction softmax comme ceci:

$$p(y|x) = \frac{e^{f_y(x)/T}}{\sum_{j=1}^{K} e^{f_j(x)/T}} \tag{2}$$

Avec le réseau de neurones $f(x) : \mathbb{R}^D \to \mathbb{R}^K$, et $f_y(x)$ la y^e valeur du vecteur output de $f(x)$.

$p(y|x)$ est donc la probabilité d'obtenir la classe y avec l'input x du réseau de neurones f.

4 Détection d'anomalies

Les programmes utilisant le machine learning sont entrainés avec des échantillons de données, on s'attend à ce que quand notre machine sera solicitée, les données qu'elle devra traiter seront semblables aux données d'entrainement. Mais ce n'est pas toujours le cas, parfois les machines font faces à des cas tellement rares et exceptionnels qu'on ne les a pas entrainé à réagir face à cette situation. Si je donne à mon detecteur d'animal une image d'avion, peut-etre dira-t'il que c'est un oiseau avec une forte probabilité. Il nous faut trouver un moyen de détecter si l'input correspond bien à ce que la machine peut traiter ou si il s'agit d'un cas inclassable.

Pour tester la capacité de notre modèle a savoir si un input est une anomalie, il faut lui donner des données pour lesquels il a été entrainé et des données pour lesquels il n'a pas été entrainé appelées anomalies. Il essaiera ensuite de trouver quelles données appartiennent à quelle distributions. On peut observer la capacité de détection avec une cartographie de notre modèle des 2 distributions, plus l'aire commune aux 2 distributions sera petite, moins il y aura de confusions et plus notre modèle sera apte à détecter les anomalies.

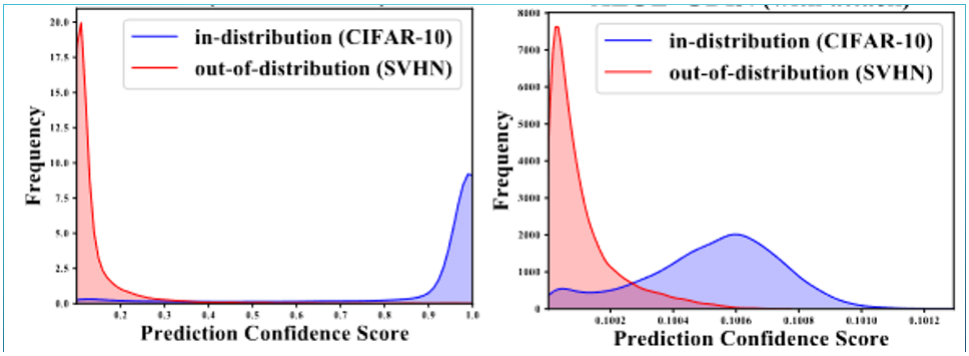

Figure 4: Distributions de scores de confiance. On voit que le modèle de la courbe de gauche sépare bien mieux les 2 distributions que le modèle de la courbe de droite, il est donc plus efficace.
[3]

5 Fonctions de coût

Une fonction de coût est une fonction qui cherche à estimer la fiabilité d'un modèle par rapport à la réalité. Lors de l'entrainement, la fonction de coût signale au réseau de neurones sa fiabilité, ce dernier modifie ses poids et ses biais pour minimiser son coût.

Pour évaluer notre modèle, on cherche à minimiser la différence entre notre distribution estimée $\hat{Y}$ et la distribution réelle Y.

Une fonction de coût commune est le carré de l'erreur moyenne:

$$MeanSquaredError(MSE) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3}$$

Le MSE est compris entre 0 et 1, plus il est proche de 0, plus le modèle est précis.

Une autre fonction de coût est le maximum de vraisemblance [1]:

$$L = \prod_i \hat{y}_i^{\,y_i} \tag{4}$$

Le maximum de vraisemblance est compris entre 0 et 1, plus il est proche de 1, plus le modèle est précis.

Voici un exemple avec les données: $\hat{y} = (0.1, 0.2, 0.7)$ et $y = (0, 0, 1)$.

$MSE = \frac{1}{3}(0.01 + 0.04 + 0.09) = 0.046$
$L = 0.1^0 \times 0.2^0 \times 0.7^1 = 0.7$
On peut dériver une autre fonction de coût à partir du maximum de vraisemblance: l'entropie croisée qui est le négatif du logarithme du maximum de vraisemblance:

$$EC = -\log L = -\log \prod_i \hat{y_i}^{y_i} = -\sum_i y_i \log \hat{y_i} \tag{5}$$

6 Energie

Les modèles basés sur l'énergie sont des modèles de machine learning. Il repose sur l'encodage de dépendances entre variables et en fonction de ces varibles connues, on cherche les variables inconnues qui minimise une mesure que l'on appelle énergie. [5] explique ce qu'est l'apprentissage basé sur l'énergie et les différentes fonctions de coût que l'on peut utiliser.

L'enjeu de [2] est de chercher à exploiter l'énergie pour identifier les anomalies. Dans un premier temps, on peut comparer la détection d'anomalies de notre modèle en testant la fonction softmax et une fonction basé sur l'énergie dans le role de dernière couche du réseau de neurones. On peut aussi entrainer des modèles à la détection d'anomalies.

L'énergie peut etre exprimée avec le negatif du logarithme du dénominateur de la fonction softmax (Eq.2):
[1]

$$E(x; f) = -T. \ln \sum_i^K e^{f_i(x)/T} \tag{6}$$

On peut faire l'analyse de cette fonction.

Soit $v(x) : \mathbb{R} \to \mathbb{R}^K$, un vecteur de taille k avec la meme valeur x sur toute ses composantes.

$$\lim_{f(x) \to v(-\infty)} E(x; f) = +\infty$$
$$f(x) = v(0) => E(x; f) = -T \ln k$$
$$\lim_{f(x) \to v(+\infty)} E(x; f) = -\infty$$

On voit que plus les valeurs d'output du réseau de neurones sont importantes, plus l'énergie est faible, ce qui veut dire que plus l'énergie est faible, moins l'anomalie est probable. Mais les détecteurs d'anomalies assigne l'anomalie à une

[1] L'article ne précisais pas quel logarithme, mais il me semble logique que ce soit le logarithme népérien.

faible valeur de leur fonction, c'est pour cela que pour comparer les détecteurs entre eux, on utilisera le négatif de l'énergie: $-E(x; f)$.

En réalité, $-E(x; f)/T$ est aligné linéairement avec le logarithme du maximum de vraisemblance[2], ce qui permet une détection d'anomalie efficace.

L'energie est une mesure sans unité. Pour faire des comparaisons d'énergies, la manière la plus consistante est de faire une distribution normalisée. La manière la plus commune et simple de faire cela est l'utilisation d'une distribution de Gibbs:

$$p(x) = \frac{e^{-E(x;f)/T}}{\int_x e^{-E(x;f)/T}} \tag{7}$$

Nous pouvons aussi entrainer un modèle avec l'énergie. Pour cela, il faut donner au réseau de neurones comme objectif de créer un écart d'énergie entre les données attendues et les données non-attendues. On peut faire cela en assignant une faible énergie aux donées de distributions attendues et une haute énergie aux données d'anomalies.

7 Résultats

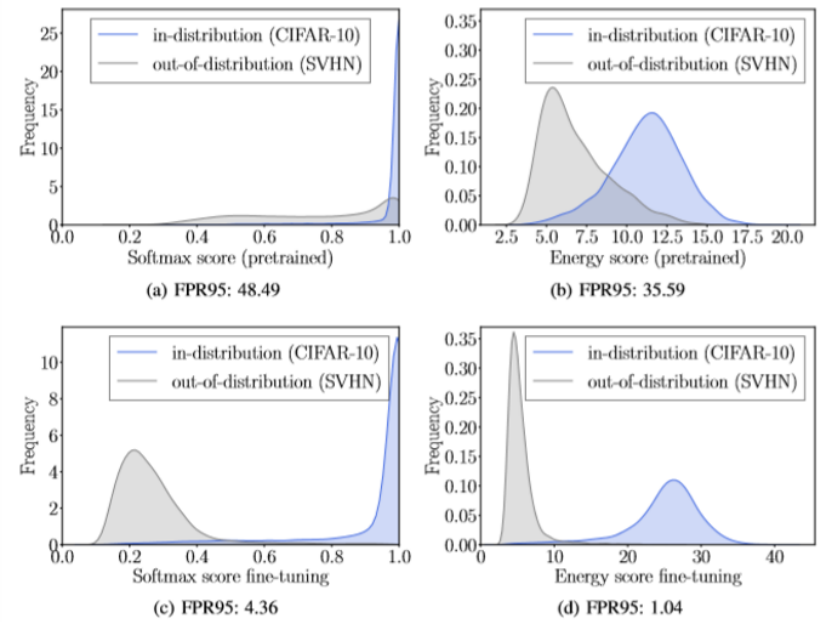

Figure 5: Distribution de softmax (à gauche) vs energie avec T=1 (à droite). En haut les résultats pour réseaux de neurones pré-entrainés, en bas les résultats pour réseaux de neurones fine-tunés avec le classificateur.
[2]

On voit graphiquement que les distributions obtenues avec l'énergie sont bien mieux disjointes que les distributions de softmax. Ce qui est confirmé par les scores de FPR95 (taux de faux positifs quand il y a 95% de vrais positifs).

8 Conclusion

L'article montre une solution prometteuse qui semble assez efficace pour détecter les anomalies dans les réseaux de neurones, que ça soit avec un modèle pré-entrainé ou un modèle déjà entrainé. La solution proposée pourrait permettre d'avoir des machines plus fiables à l'heure ou elles ont des roles de plus en plus important dans la société.

9 Remerciement

Je remercie Violaine ANTOINE qui m'a parrainé durant cette UE. Son temps et ses conseils m'ont été précieux.

References

[1] L. Le Cam. *Maximum Likelihood: An Introduction*. 1990.

[2] Weitang Liu et al. *Energy-based Out-of-distribution Detection*. 2020.

[3] Xi Wu Yingyu Liang Somesh Jha Jiefeng Chen, Yixuan Li. *Robust Out-of-distribution Detection for Neural Networks*. 2020.

[4] Jürgen Schmidhuber. *Deep learning in neural networks: An overview*. 2014.

[5] Raia Hadsell Marc'Aurelio Ranzato Yann LeCun, Sumit Chopra and Fu Jie Huang. *A Tutorial on Energy-Based Learning*. 2006.