Sciences de la Décision et Management des Risques

2020-2021

Titre de la recherche bibliographique

La gestion de l'incertitude des réseaux de neurones dans l'aide à la décision

—

Auteur

JEFFREY VERDIERE

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE
IAE PARIS
SORBONNE BUSINESS SCHOOL

Arts et Métiers
Sciences et Technologies

Notice Bibliographique

Auteur/e : VERDIERE JEFFREY

Année :2020-2021

Directeur/e de recherche : JEAN-MARIE FLAUS

Titre de la recherche bibliographique : La gestion des incertitudes des réseaux de neurones dans l'aide à la décision.

Mots clés (5 maximum) : Réseaux de neurones, Epistémique, MonteCarlo DropOut, Réseaux Bayésiens, Incertitudes

Code Python : La plupart des figures ou résultats sont extraits d'un fichier python disponible au format .pdf sur https://github/jeffreyverdier dans le fichier revue bibliographique incertitude des réseaux de neurones. D'autre part, les modèles mathématiques présentés dans cette revue bibliographique sont disponibles sous format python dans ce même fichier.

Résumé en français (10 lignes) :

Comprendre ce que les réseaux de neurones ne connaissent pas ou ne comprennent pas est une tâche difficile. Aujourd'hui, les réseaux de neurones sont capables de donner des représentations puissantes de grosses bases de données. Les prédictions de ces réseaux de neurones sont souvent considérées comme précises, stables et certaines mais ce n'est pas toujours le cas. Un récent exemple a eu des conséquences désastreuses. En mai 2016, est survenu le premier accident de voiture autonome en Californie causé par un problème de mauvaise perception d'un réseau de neurone. En effet, la confusion d'une intelligence artificielle de la couleur du ciel avec un camion blanc a causé cet accident tragique.

Si ce réseau de neurones aidant la navigation autonome avait été capable de fournir un meilleur résultat d'incertitudes sur ses prédictions, il aurait été capable de prendre des meilleures décisions et d'éviter l'accident.

Cette revue bibliographique a pour objectif de balayer la problématique de l'incertitude des résultats des réseaux de neurones et les solutions envisagées notamment lorsqu'ils sont impliqués dans des processus d'aide à la décision.

Résumé en anglais (10 lignes) :

Understanding what neural networks don't know or understand is a difficult task. Today, neural networks are capable of providing powerful representations of large databases. The predictions of these neural networks are often considered accurate, stable and certain, but this is not always the case. A recent example has had disastrous consequences. In May 2016, there was the first autonomous car accident in California caused by a neural network misperception problem. Indeed, the confusion of an artificial intelligence of the color of the sky with a white truck caused a tragic accident.

If this neural network assisting autonomous navigation had been able to provide a better result of uncertainty in its predictions, it would have been able to make better decisions and avoid the accident.

The objective of this literature review is to explore the problem of uncertainty in the results of neural networks and the solutions considered, particularly when they are involved in decision support processes.

J'atteste que ce travail est personnel, cite en référence toutes les sources utilisées et ne comporte pas de plagiat. Le nom de l'auteur/e du texte vaut signature.

NOM : VERDIERE JEFFREY

Remerciements

Merci à Jean-Marie Flaus, professeur en Risques Industriels à l'université de Grenoble, pour son inconditionnel soutien durant l'écriture de cette revue bibliographique et lors de l'élaboration du projet de recherche.

Merci également à Marc Lassage et Fréderic Gautier pour leurs conseils, orientations et enseignements durant cette année au sein du Master Recherche Sciences de la Décision et Management des Risques.

Table des matières

Notations

Symboles romains

X Base de données d'entrée(input)

Y Base de données de sortie(labels)

W Matrice des poids

x_n Donnée d'entrée du modèle (input)

y_n Donnée de sortie connue du modèle (label)

$\widehat{y_n}$ Sortie du modèle pour une entrée(input) x_n

x^* nouvelle entrée d'un modèle

y^* Sortie d'un modèle associée à l'entrée de validation x^*

Q Dimension des entrées(input)

D Dimension des sorties(output)

Autres symboles

R Les nombres réels

N La distribution Gaussienne

ω les paramètres de la fonction f (provenant de la matrice poids W)

A/B La variable aléatoire A expliquée par la variable aléatoire B

¡A,B¿ Produit scalaire entre l'élément A et l'élément B

Abréviations

NN Réseaux de Neurones

BNN Réseaux de Neurones Bayésiens

Introduction

Delen and Sharda [2008] affirment que les réseaux de neurones sont essentiels dans les systèmes d'aide à la décision.

Cependant, Thompson et al. [2020] expliquent que les réseaux de neurones ne sont pas encore totalement maitrisés par leurs créateurs et avancent qu'il est souvent compliqué de comprendre certaines de leurs prédictions.

Or, Mannarswamy and Roy [2019] constatent que ces réseaux de neurones sont souvent des aides à la décision dans des domaines aussi sensibles que la finance, la défense ou la médecine. Dans ce contexte, il est inenvisageable de ne pas comprendre les incertitudes de leurs prédictions. Des méthodes doivent être mises en place pour mieux les appréhender.

Les structures usuelles de réseaux de neurones et les raisons pour lesquelles ils interviennent dans les prises de décision seront présentées dans un premier temps. Dans un second temps, seront mis en évidence les résultats des réseaux de neurones qui ont posé problème dans des domaines sensibles. Dans un troisième temps, sera formalisé mathématiquement le problème de l'incertitude et de ses possibilités d'évaluation. Dans un dernier temps, seront présentées les méthodes permettant d'évaluer les incertitudes des prédictions des réseaux de neurones ainsi que l'analyse de leurs résultats. Enfin, une revue de ces applications et de leurs limites en "vie réelle" sera effectuée.

Du fonctionnement des réseaux de neurones

Les structures des réseaux

Les structures de base de réseaux de neurones dont toutes les autres découlent seront étudiées. Cette étude se focalisera uniquement sur l'apprentissage supervisé car Gal [2016] affirme qu'il n'existe pas encore d'état de l'art sur les méthodes d'évaluation des incertitudes en apprentissage non supervisé.

Dang et al. [2020] présentent l'apprentissage supervisé comme une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir d'exemples annotés. Sardar et al. [2019] proposent de segmenter l'apprentissage supervisé en deux familles : la classification et la regression. La classification, selon Xin and Wang [2019], vise à classer chaque donnée fournie au réseau suivant des classes que le réseau aura appris à reconnaitre. Par exemple, si un réseau est entraîné à reconnaitre des nombres entre 0 et 10 écrits à la main, pour chaque nouvelle entrée le réseau attribuera en sortie une classe(c'est-à-dire un résultat entre 0 et 10).
La regression, selon Kumar [2005]., consiste à effectuer la même tâche que la regression linéaire. Comme en regression, le réseau doit être capable de prédire une valeur en fonction de données d'entrainements multidimensionelles. La table (1) récapitule les différentes familles d'apprentissage.

Table 1 – Type d'apprentissage d'un réseau de neurones

Apprentissage			
supervisé		non supervisé	
classification	regression	classification	regression

Gal [2016] explique que le multiperceptron et le réseau convolutif sont les deux structures de réseaux de neurones les plus utilisées et dont découlent toutes les autres structures.

Le multiperceptron : Réseau de Neurones (NN)

Rumelhart et al. [2011] ont été les premiers chercheurs en intelligence artificielle à formuler mathématiquement la structure du multiperceptron plus connu sous le nom de réseau de neurones. Le réseau multi-perceptron est une structure "feedforward" où des éléments interconnectés entre eux appelés neurones régissent la structure globale. Le terme feedforward signifie que la structure améliore ses prédictions au fur et à mesure d'une étape que l'on appelle l'entrainement. L'objectif de l'entrainement est de corriger les prédictions du réseau au fur et à mesure en comparant les sorties du réseau(output) à des sorties connues que l'on appelle les labels.

Au niveau de sa structure Rumelhart et al. [2011] expliquent qu'un réseau de neurones contient au minimum trois hiérarchies appelées "couches" : une couche d'entrée(input layer), une couche intermédiaire (hidden layer) et une couche de sortie(output layer). Le nombre et le type de couches intermédiaires peuvent varier en fonction du type de problèmes traités (régression, classification, traitement d'images).

La formalisation mathématique d'une couche d'un réseau de neurones est un élément essentiel dans la compréhension de ce fonctionnement.

On note alors $X = [x_1, x_2, ..., x_N]^t$ le vecteur d'entrée(vecteur d'input) contenant N features(caractéristique) permettant de prédire un output estimé noté $\hat{y}$.
On note $W_j = [w_{j1}, w_{j2}, ..., w_{jN}]$ le vecteur poids entre la couche d'entrée(input layer) et la couche intermédiaire(hidden layer),b_j le biais du réseau.
A partir de ces notations, les sorties des réseaux de neurones peuvent alors être décomposées en deux étapes. Dans une première étape,la sortie de la couche cachée qui est récapitulée dans l'équation(2).

$$y_j = \phi(w_j^t + b_j) = \phi(\sum_{i=1}^{n} w_{ij} + b_j) \tag{1}$$

Avec ϕ_j la fonction d'activation du réseau qui est généralement une fonction non-linéaire comme une fonction sigmoide.
Dans une deuxième étape, les sorties de la couche intermédiaire seront passées dans la couche de sortie notée outputlayer.
Feng and Lu [2019] expliquent que la sortie de la couche intermédiaire passe à travers une fonction d'activation de sortie qui est généralement linéaire pour des problèmes de regression et non-linéaire pour des problèmes de classifications.
L'équation suivante résume alors ce passage de la couche intermédiaire à la couche de sortie :

$$\hat{y} = \phi_0(\sum_{j=1}^{M} w_{0j} * y_j + b_0) \tag{2}$$

A l'exception de la couche d'entrée, tous les neurones contiennent des fonctions d'activations (différentes si pour des méthodes de classification ou des méthodes de régression) qui peuvent être linéaires ou non en fonction du problème traité.

Le multiperceptron présenté précédemment peut être complexifié pour créer des modèles permettant de traiter des images ou des données séquencées.

Le réseau convolutif(CNN)

Rumelhart et al. [2011] ont montré que le multiperceptron est très efficace pour faire des prédictions à partir d'entrées numériques continues mais posent la question des images et des données séquencées.
C'est dans cet objectif que Lecun et al. [1997] ont mis en place le réseau de neurones convolutifs. Cette recherche a permis d'aborder le traitement d'images par réseaux de neurones qui n'était jusqu'alors pas envisagé. Le modèle de Lecun et al. [1997] est alors adapté par Rawat et al. [2017], proposant une composition d'applications récursives de type convolution et de couches pooling suivies d'une couche d'activation à la sortie du réseau.
La couche convolutive(CONV) utilise des filtres qui scannent une image suivant ses dimensions en effectuant des opérations de convolution.
La couche Pooling(POOL) est une opération de sous-échantillonage typiquement appliquée après une couche convolutif. Les types de pooling les plus populaires sont le max et l'average pooling où les valeurs maximales et moyenne sont extraites d'une série de pixels d'une image. Une couche fully-connected s'applique à une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones.
Les couches d'activations sont utilisées à la fin des architectures de réseaux de neurones convolutives pour imprimer des scores.

Pour fixer les paramètres(poids) des structures neuronales présentées précédemment, Liu et al. [2015] proposent une méthodologie pour entrainer les réseaux.

L'entrainement des réseaux de neurones, une nécessité

L'objectif d'un réseau de neurones est d'être capable de faire une représentation d'un ensemble de données. Pour ce faire, une phase d'entrainement fixe les poids du réseau puis ses performances sont vérifiées lors d'une phase de validation(ou phase de test)[Liu et al., 2015].

La méthode de retropropagation du gradient a été mise en place par Rumelhart et al. [2011] puis a été adaptée par Zhang et al. [2019] afin d'optimiser l'entrainement des réseaux de neurones.

L'algorithme de rétropropagation du gradient présenté par Zhang et al. [2019] consiste en deux étapes :
Dans l'étape "forward", les outputs de prédictions sont calculés suivant un input d'entrée. Dans la phase "backward", les dérivées partielles de la fonction coût permettent d'ajuster les paramètres du réseau jusqu'à atteindre un certain critère. Ce process est répété jusqu'à ce que les poids du réseau aient convergé vers des valeurs fixes. Le critère fixé par an Zhang et al. [2019] est la convergence de la fonction de coût vers un certain palier.

an Zhang et al. [2019] utilisent classiquement la fonction de coût des moindres au carré(MSE).

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2 \tag{3}$$

Avec n le nombre de données d'entrainement, $\widehat{y_i}$ l'output du réseau pour l'entrée i et y_i la vraie valeur(label).
Sur la figure(1) est tracé un exemple d'évolution de la fonction de coût(perte) au fur et à mesure de l'entrainement du réseau. Le réseau "converge" bien vers une erreur constante.

La convergence de la fonction de coût vers un palier n'est pas toujours immédiate selon Ying [2019]. En effet,le réseau peut être enclin à de l'overfitting. L'overfitting est une surpécialisation d'un réseau de neurones qui peut être amené à vouloir trop s'adapter aux données d'entrée quitte à ne pas savoir généraliser ou dégager des tendances. Ying [2019] considère qu'une augmentation du nombre de poids ou qu'une augmentation de

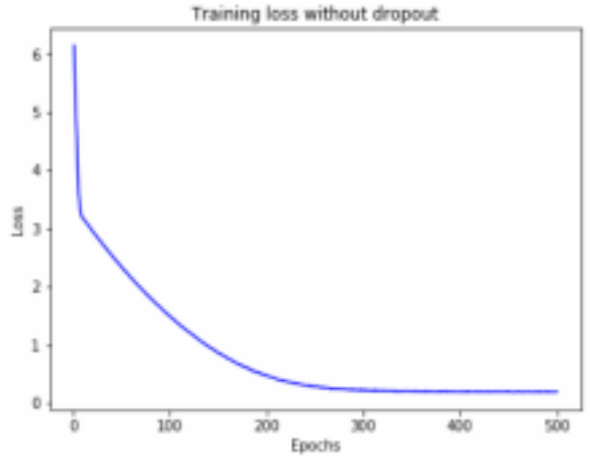

FIGURE 1 – Tracé de la fonction de perte de type MSE

la base de données d'entrainement, pourraient limiter ce phénomène. Néanmoins, Gal [2016] explique qu'il existe rarement la possibilité d'augmenter la taille des bases de données ou d'augmenter la complexité des réseaux. Ainsi, les modèles mathématiques qui permettent d'estimer l'incertitude des réseaux de neurones permettent aussi de compenser ce problème. Ils seront présentés plus loin.

Lecun et al. [1997] ont montré que les réseaux de neurones par leur structure et par leur optimisation, permettent de comprendre des problèmes très complexes et sont également utiles pour trouver des relations dans les bases de données les plus lourdes(images, données séquencées).Ils sont donc des outils précieux dans l'aide à la décision.Zhang et al. [2019] expliquent que les réseaux de neurones ne sont que des fonctions deterministes : à une entrée X est associée une sortie Y. Cependant, cet aspect deterministe d'un réseau de neurones peut poser problème dans certaines applications dans le support à la décision. Les problèmes de sécurité provenant de cet aspect deterministe des réseaux de neurones seront présentés dans la suite de l'étude.

L'aide à la décision par les réseaux de neurones et la problématique de l'incertitude

Mannarswamy and Roy [2019] ont montré que les réseaux de neurones jusqu'alors uniquement utilisés par les laboratoires de recherche des universités américaines comme un outil d'avance technologique commencent à être utilisés dans la vie réelle. Cependant, Huang et al. [2020] expliquent que l'interaction des réseaux de neurones avec le monde réel peut poser des problèmes de sécurité. Certaines situations automatisées par des réseaux de neurones peuvent devenir dangereuses pour la vie humaine si les prédictions des réseaux de neurones ne sont pas maîtrisées. Il peut s'agir en effet d'outils d'aide à la décision automatique ou de recommandation en Médecine, de véhicules autonomes,de robots effectuant des tâches au milieu d'opérateurs dans les usines ou de "robot-trader" dans les salles de marché. Plusieurs scénarii seront évoqués pour illustrer ce propos et mettre en avant la nécessité de quantifier l'incertitude.

D'un cas simpliste à des applications sensibles

Burt et al. [2018] sont les premiers à expliciter l'importance de l'incertitude des réseaux par un cas très simple. Un utilisateur donne à la couche d'entrée d'un réseau une photo de son chien. Il est produit une prédiction de classe(berger, labrador...) car le réseau a été uniquement entrainé à détecter des races de chiens. Si l'utilisateur présente donc une photo de son chat, il recevra une prédiction du réseau de neurones de race de chien puisque le réseau n'a pas été entrainé à détecter des races de chat.Si cet exemple reste ludique, d'autres le sont moins et peuvent poser de graves préjudices à la société qui les met en place.

Un rapport Guynn [2016] évoque une confusion délicate générée par un réseau de neurones de Google. Une application ludique de Google propose d'identifier la nature du contenu d'une photo(objet ou être vivant). Cette application a malheureusement confondu deux hommes afro-américains avec des singes.
Plus grave, quelques mois plus tard, le premier accident d'une voiture autonome de Google a eu lieu. Un réseau de neurones assurant la navigation a confondu un panneau blanc avec la couleur du ciel et a donc provoqué un accident dramatique[Habib et al., 2016].

Pour élargir le propos, sera évoquée dans la suite, la problématique de l'incertitude sur d'autres cas concrets telles que la detection de tumeurs sur des IRM ou les décisions sur les marchés financiers.

Le danger des réseaux de neurones dans l'aide à la décision dans le domaine médical

Pirracchio et al. [2018] expliquent que depuis ces dernières années, en ophtalmologie, en gastrologie et en cancerologie, pour aider au diagnostic, le médecin s'appuye sur des technologies très avancées(IRM,Scanner, Retinographe) dont l'interprétation des résultats sont souvent complexes et chronophages. Pirracchio et al. [2018] expliquent également que les techniques de vision par ordinateur ont permis depuis les dernières années de simplifier l'analyse des résultats émanant de ces technologies et pouvent aider largement le corps médical. Pour automatiser certaines tâches, Tanno et al. [2021] ont montré que les réseaux de neurones peuvent intervenir sur de l'imagerie après avoir été entrainées par des images labellisées par des experts médicaux. L'objectif des réseaux de neurones est de trouver les anomalies des plus faciles aux plus difficiles permettant aux médecins de ne rien laisser passer et de se concentrer sur les taches plus compliquées et donc de gagner du temps. Namee et al. [2002] estiment que l'introduction de systèmes de détection automatique sur imagerie peut rendre le diagnostic plus compliqué qu'il ne l'était originellement. En effet, ces systèmes d'aide à la décision peuvent modifier le jugement initial du médecin et ajouter des biais dans sa prise de décision.
Santoro [2017] explique qu'un réseau de neurones qui donnerait un jugement sur une situation à laquelle il n'aurait jamais été confronté pourrait faire des suggestions déraisonnées et avoir un impact négatif sur le jugement initial du médecin.

Le danger des prises de décisions des réseaux de neurones dans le trading haute fréquence

Dans le trading haute fréquence,Chordia et al. [2013] constatent que des ordinateurs prennent le contrôle sur des systèmes qui pourraient potentiellement dérégler l'économie globale.Pour ces mêmes auteurs, l'objectif du trading haute fréquence consiste à transmettre automatiquement et à très grande vitesse des ordres sur les marchés financiers, sans intervention humaine, à l'aide d'algorithmes. Ces programmes analysent des carnets d'ordre électroniques complexes afin d'anticiper des micro-mouvements de marché pour en tirer profit. En pratique, les transactions sont réalisées dans un laps de temps éclair : de l'ordre de la micro seconde.

Huang et al. [2016] expliquent que pour atteindre ces performances, l'apprentissage supervisé des réseaux de neurones a pris de plus en plus de place en trading haute fréquence.

Il est à considérer, selon Daniel [2013], que le trading haute fréquence a déjà provoqué des micro crash financiers que l'Autorité des Marchés Financiers a des difficultés à anticiper. Des structures algorithmiques peuvent en effet générer des commandes d'actions incontrôlées qui pourraient générer des crises sur l'économie réelle. En effet, il suffit qu'un réseau de neurones soit confronté à une situation qu'il n'a pas bien "comprise" ou à laquelle il n'a jamais été confronté pour qu'il fasse des prédictions déraisonnées. Kirilenko et al. [2017] rapportent une mauvaise interaction entre des réseaux de neurones de différents fonds de trading lors du flash crack en 2011.

Une solution possible proposée par Kirilenko et al. [2017] serait de créer des règles de décision qu'un contrôleur pourrait vérifier avant de lancer le programme. Ce type de vérification permettrait à l'Autorité des Marchés Financiers de vérifier ces algorithmes avant leur déploiement. Néanmoins, Huang et al. [2016] expliquent les réseaux de neurones ne permettent pas ce type de vérification. En effet, les réseaux de neurones ne sont pas des séries d'instructions encodées dans un ordinateur. Se pose alors la question de ce qu'il est possible de faire lorsqu'un réseau de neurones est confronté à des données financières qu'il n'a jamais connues ou dont il n'est pas certain de sa prédiction.

Ces différents exemples mettent bien en évience la nécessité d'évaluer les incertitudes inhérentes à la prédiction des réseaux de neurones.

L'incertitude ou l'art de comprendre ce que les réseaux de neurones ne savent pas ou ne peuvent pas comprendre

Différents types d'incertitudes

Kiureghian et al. [2009] mettent en évidence deux types d'incertitudes liées aux prédictions des réseaux de neurones :

Epistémique

L'incertitude epistémique met en évidence l'ignorance d'un réseau de neurones. En effet, les paramètres d'un réseau de neurones sont générés à partir de données d'entrainement qui ne couvrent généralement pas tout le problème que l'on souhaite résoudre. Cette incertitude pourrait être théoriquement supprimée si l'on fournissait la totalité des données supposées. Cette incertitude est parfois plus connue sous le nom d'incertitude des paramètres(poids) du modèle.

Aléatoire

L'incertitude aléatoire met en évidence les bruits(erreurs de mesures) des données attribuées au modèle. Il peut s'agir des bruits issus de capteurs ou de mauvaises mesures. Ces erreurs ne pourraient donc pas être réduites par augmentation du nombre de données. Elle se décompose en deux sous familles.
L'incertitude homoscédastique est une erreur qui reste constante pour toutes les entrées(inputs) : par exemple, récupérer des photos avec un appareil présentant un défaut. L'ensemble des photos sera donc impacté de la même manière, ce qui donnera lieu à une erreur(bruit) constante sur les données.
L'incertitude hétéroscédastique dépend également des entrées du modèle. Contrairement à l'erreur homoscédastique, certaines entrées peuvent être plus bruitées que d'autres. Par exemple, un capteur qui récupère des données peut très bien mal fonctionner sur une certaine plage de valeurs et bien fonctionner sur d'autres plages. Cela donne lieu à des erreurs non constantes sur les données.

Une illustration des familles d'incertitude : le cas de la regression

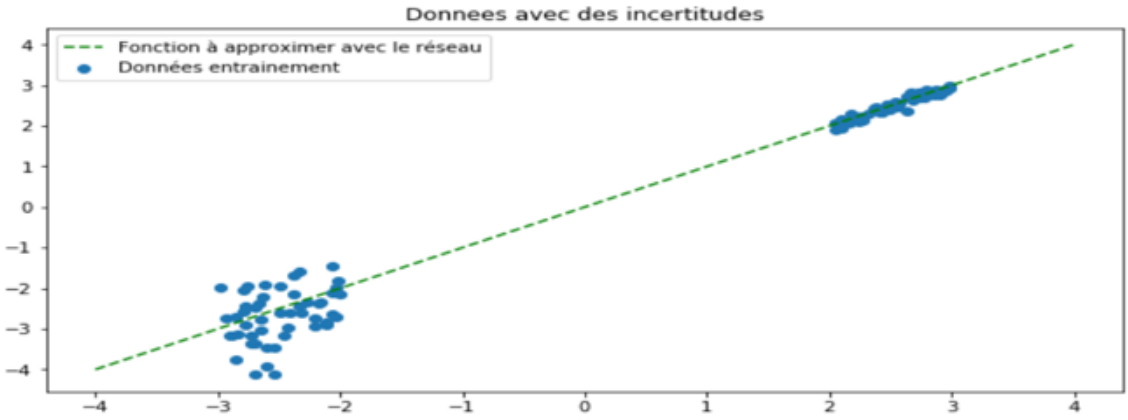

FIGURE 2 – Type d'incertitude

Brando et al. [2019] mettent en évidence les différentes familles d'incertitudes sur un cas simple de regression. L'objectif ici est d'approximer la droite y=x avec un réseau de neurones. Pour l'entrainer et donc régler ses poids, les auteurs des données créent des données d'entrainement suivant différentes distributions comme illustré sur la Figure (2). Cent points seront alors générés dans le nuage de gauche entre $[-3;-2]$ et cent points dans le nuage de droite entre $[2;3]$. Du bruit est rajouté avec une variance constante et faible dans le nuage de points de droite et du bruit est rajouté avec une variance suivant une loi normale pour le nuage de gauche.

L'erreur epistémique est représentée par le manque de données sur les intervalles $]-\propto ;-3[\cup]3;+\propto[$.
L'erreur aléatoire hétéroscédastique est représentée par le bruit non constant du nuage de gauche.
L'erreur aléatoire homoscédastique est représentée par le bruit constant du nuage de droite.

Une illustration des familles d'incertitude : le cas de la classification

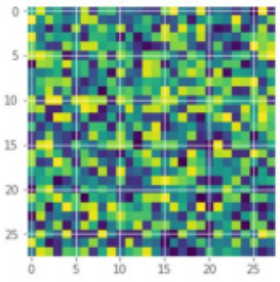

FIGURE 3 – Erreur aléatoire en classification

Sensoy et al. [2018] mettent en évidence les différentes familles d'incertitudes sur un cas simple de classification.Un réseau de neurones convolutif est ici entrainé à reconnaitre des nombres manuscrits entre 0 et 10. Il devra donc prédire pour chaque entrée du réseau une classe de 0 à 10. Suite à son entrainement, il ne pourra cependant pas être en mesure de faire des prédictions correctes dans tous les cas de figures. En effet, le nombre qui est donné est manuscrit, ainsi certaines erreurs d'écriture peuvent entrainer des nombres illisibles comme sur la figure(4). Le nombre de la figure(4) peut être interprété comme 0 ou comme un six. Il s'agit ici de relever une erreur épistémique : le réseau n'est pas en

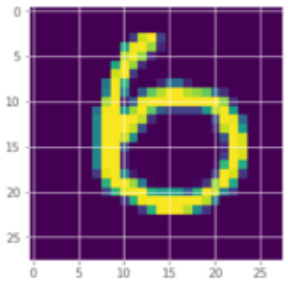

FIGURE 4 – Erreur épistémique en classification

mesure de savoir.

Dans le cas d'une image totalement bruitée comme sur la figure(3), le réseau ne doit émettre aucune prédiction et être capable de discerner une erreur aléatoire hétéroscédastique.

La formalisation du problème de l'incertitude

Zhang et al. [2019] expliquent qu'en apprentissage supervisé, la tâche à accomplir consiste à trouver une fonction $f(\omega)$ faisant le lien entre un ensemble d'entrée $X = [x_1, ..., x_N]$ et de sorties connues $Y = [y_1, ..., y_N]$. Cette fonction f possède un certain nombre n de paramètres $\omega = [\omega_1, ..., \omega_n]$ appelés poids du réseau à régler pendant la période d'entrainement grâce à une méthode d'optimisation du gradient.Les mêmes auteurs expliquent qu'en phase de validation, lorsqu'un nouvel événement(input) $x*$ est présenté au réseau, la fonction f calcule une unique valeur(output) $y^* = f^{\omega}(x^*)$. Thompson et al. [2020] ont montré que cette prédiction est produite sans estimer l'incertitude associée. En effet, le réseau est certain de sa prédiction puisque f est une fonction deterministe : à une entrée est associée une sortie. Pour estimer l'incertitude,Beal et al. [2003] ont considéré $f^{\omega}(x^*)$ comme une distribution de probabilité et non plus comme une fonction deterministe.

La nécessité des réseaux de neurones bayésiens(BNN)

Pour formaliser cette probabilité de distribution des outils bayésiens ont été mis en place.

Les réseaux de neurones bayésiens ont été introduits pour la première fois par MacKay [2009]. Il fallait en effet proposer une interprétation probabiliste des prédictions des réseaux de neurones. A date, Jospin et al. [2007] ont montré que cette vision bayésienne revient à considèrer f^{ω} comme une distribution de probabilité donnant la distribution de y^* en se basant sur l'entrée x^* et les données d'entrainement X, Y.

Jospin et al. [2007] ont modifié les poids ω d'un réseau de telle sorte qu'ils deviennent des variables aléatoires à estimer lors de l'entrainement. L'optimisation consiste à trouver la meilleure distribution permettant d'expliquer Y en se basant sur les entrées X. Ce calcul de la distribution s'effectue en appliquant le théorème de Bayes dont une version conditionnée a été proposée par James [2003] pour 3 événements A,B et C :

$$p(A/B \cap C) = \frac{p(B/A \cap C)p(A/C)}{p(B/C)} \tag{4}$$

Les événements suivants sont considérés :
Likelihood : Probabilité que Y ait été généré avec X et ω
Prior : Probabilité d'apparition des paramètres ω en utilisant les connaissances antérieures

sans connaissance sur X et Y
Evidence : Probabilité d'obtenir Y étant donné X.
En utilisant la version conditionnée du théorème de Bayes proposé par James [2003], on a alors :

$$p(\omega/X,Y) = \frac{p(Y/X,\omega)p(\omega)}{p(Y/X)} = \frac{Likelihood * Prior}{Evidence} = Posterior \tag{5}$$

La probabilité d'obtenir Y étant donné X peut être calculée en intégrant sur l'ensemble des poids :

$$p(Y/X) = \int p(Y/X,\omega)p(\omega)d\omega \tag{6}$$

Enfin à partir des équations (4) et (5), la distribution souhaitée peut être calculée :

$$p(y^*/x^*,X,Y) = \int p(y^*/x^*,\omega)p(\omega/X,Y)d\omega \tag{7}$$

Le problème d'estimation des incertitudes pourrait s'arrêter à ce calcul intégral.
Néanmoins, Gal [2016] montre que le calcul intégral de $p(Y/X)$ demande de savoir calculer analytiquement $p(\omega/X,Y)$ qui est impossible à réaliser avec les outils numériques actuels. Pour calculer cette probabilité conditionnelle, une technique d'optimisation est proposée : l'inférence varationelle.

L'inférence varationelle : outil essentiel pour estimer l'incertitude

Beal et al. [2003] soulignent la difficulté pour estimer analytiquement la probabilité conditionnelle. Les auteurs expliquent que le calcul peut se révéler impossible surtout lorsque le modèle se complexifie(réseaux avec beaucoup de poids et de couches).

Pour surmonter ces limitations analytiques et numériques, Jordan et al. [1999] ont proposé d'approcher la probabilité conditionelle $p(\omega/X,Y)$ par une distribution de probabilité notée $q_\theta(\omega)$ paramétrée par θ. Cette méthode revient à minimiser l'opérateur mathématique de Kullback-Leibler entre la vraie distribution postérieure et la distribution varationnelle. Formellement,cela revient à minimiser la quantité suivante[Filippi et al., 2010] :

$$L(\theta) = KL(q_\theta(\omega), p(\omega/X,Y)) = \int q_\theta(\omega) log(\frac{q_\theta(\omega)}{p(\omega/X,Y)}d\omega \tag{8}$$

Après avoir minimisé l'opérateur de Kullback-Leibler entre la distribution connue et la distribution $q_\theta(\omega)$, on peut alors remplacer dans l'équation (6) :

$$p(y^*/x^*,X,Y) = \int p(y^*/x^*,\omega)p(\omega/X,Y)d\omega \approx \int p(y^*/x^*,\omega)q_\theta(\omega)d\omega \tag{9}$$

Cette minimisation de l'opérateur de Kullback-Leibler proposée par Filippi et al. [2010] permettant d'approximer $p(y^*/x^*,X,Y)$ s'appelle l'inférence varationnelle.

Pour résoudre ce problème d'optimisation et pour trouver une distribution varationelle $q_\theta(\omega)$ qui minimise l'opérateur de Kullback-Leibler, Laves et al. [2020] ont montré que le MonteCarlo Dropout était la méthode la plus adaptée.

Le Monte Carlo Dropout pour approcher l'inférence varationnelle

Denker and LeCun [1990] sont les premiers chercheurs en intelligence artificielle à développer une technique d'approximation de l'inférence pour les réseaux bayésiens en se basant sur les techniques numériques d'approximation de MonteCarlo. Cette méthode numérique peut etre appliquée dans les deux phases qui permettent de règler les poids d'un réseau de neurones : l'entrainement et la validation.

Le MonteCarlo Dropout pendant la phase d'entrainement

Gal and Ghahramani [2015] ont montré que la méthode de Monte Carlo pendant l'entrainement consiste à approximer q_θ par une distribution de probabilité connue pour approximer l'inférence varationnelle. Gal and Ghahramani [2016] précisent que le Monte-Carlo Dropout revient à approximer $q_\theta(\omega)$ par une distribution de Bernoulli $Z_i \hookrightarrow B(p_i)$ avec p_i la probabilité de dropout sur le poids ω_i. Chaque poids est alors calculé de la manière suivante $\omega_i = m_i * Z_i$. L'apprentissage pour trouver le paramètre m_i se fait par la méthode classique de descente de gradient présenté en section 2. Le paramètre de dropout(p_i) peut etre défini par l'utilisateur de l'algorithme d'optimisation. Par sa structure, Gal and Ghahramani [2016] ont montré que la loi de Bernoulli a pour effet de désactiver certains poids du réseau bayésien avec une probabilité p_i.Les effets d'un "tirage de dropout" sur une structure classique de type multiperceptron sont illustrés sur la figure 5.

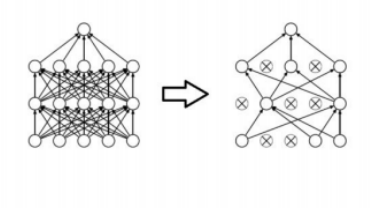

FIGURE 5 – Impact du MonteCarlo Dropout sur un réseau de neurones

L'impact du MonteCarlo dropout sur une structure de réseaux de neurones bayésiens est illustré sur la figure 5. Celui-ci a pour effet de désactiver certains poids du réseau bayésien avec une probabilité p_i et donc de générer une nouvelle structure neuronale sensiblement différente de l'originale.

Srivastava et al. [2014] ont montré que la modification de cette phase d'entrainement permet de limiter le phénomène d'overfitting présenté en section 2. En effet, cela évite que le réseau se superpécialise sur un type de données. Le fait de "cacher" certains poids oblige le réseau à s'adapter aux nouvelles situations.

Pendant l'entrainement, le MonteCarlo Dropout ne permet pas de mettre en évidence l'incertitude des prédictions des réseaux de neurones. Gal and Ghahramani [2016] ont montré que l'utilisation du MonteCarlo Dropout pendant la phase de validation permet de la mettre en évidence.

Le dropout pendant la phase de validation et une première mise en évidence de l'incertitude

Ces mêmes auteurs (Gal and Ghahramani [2016]) expliquent qu'une fois l'entrainement réalisé avec cette méthode de dropout, il est possible de la réutiliser pendant la phase de validation.

Pour une même entrée x^* passée T fois dans notre réseau bayésien composé de couches de dropout, T prédictions sensiblement différentes sont obtenues. En effet,le dropout a pour effet d'annuler certains poids du réseau bayésien suivant une loi de Bernoulli de probabilité p_i et donc de générer un réseau bayésien différent à chaque tirage. Cette même entrée x^* passe donc par T réseaux bayésiens sensiblements différents,obtenus à partir du réseau bayésien initial. Cet effet de tirage plus couramment connu sous le nom de masque de dropout est illustré par la figure (6).

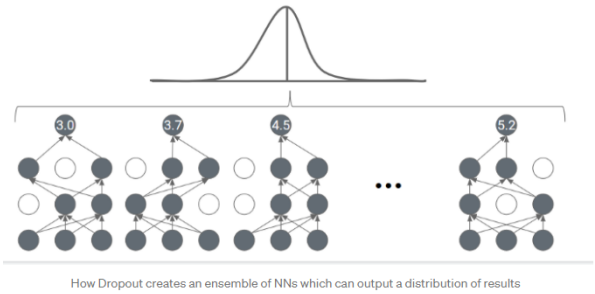

FIGURE 6 – Effet de T tirages de dropout sur un même réseau baysien

Kendal and Gal [2017] expliquent que ce phénomène de masque de dropout, illustré sur la figure (6), est assimilable à une distribution de type gaussienne. Comme chacun des "tirages" des réseaux bayésiens est indépendant et que la distribution sur les poids est une loi de Bernoulli, les résultats peuvent donc être "rassemblés" sous forme d'une gaussienne définie par sa moyenne et son écart-type. Ainsi, la distribution de $f^{\omega}(x^*)$ pour les T tirages permet de récupérer les moments d'ordre 1 et d'ordre 2 correspondant respectivement à une moyenne et à une variance.
Kendal and Gal [2017] précisent que la moyenne peut être interprétée comme la prédiction du réseau de neurones bayésiens et la variance comme l'incertitude epistémique du réseau bayésien. Ces moyennes et variances sont calculées à partir des prédictions des réseaux bayésiens de la manière suivante :

$$E(y^*) = \frac{1}{T}\sum_{t=1}^{T} f^{\omega}(x^*) \tag{10}$$

$$V(y^*) = E(y^{*2}) - E(y^*)^2 \tag{11}$$

L'exemple présenté par Brando et al. [2019] consiste à entrainer un réseau de neurones pour prédire la droite $y = x$ avec des données présentant des erreurs epistémiques et aléatoires. Ils comparent les performances d'un multiperceptron et d'un réseau de neurones bayésiens incluant des couches de dropout. Pour comparer leurs performances, les deux réseaux(multiperceptron et NN bayésiens) sont testés sur des données de validation représentées par les traits en pointillés de la figure(7). Le multiperceptron prédit une droite parfaite avec un coefficient de corrélation élevée avec les données de validation. Le réseau de neurones bayésiens incluant une couche de dropout, montre des oscillations plus importantes dans les zones où le réseau n'a pas été entrainé et met donc en évidence l'incertitude epistémique.

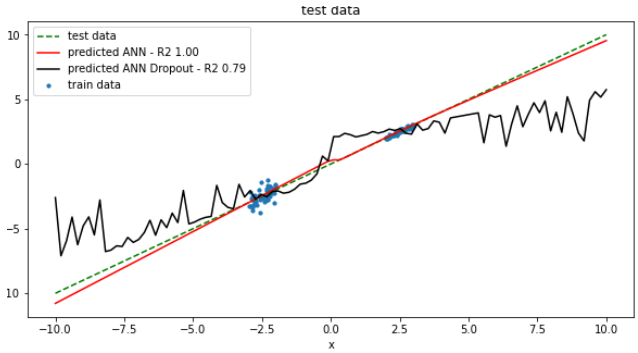

FIGURE 7 – Comparaison des prédictions d'un multiperceptron et d'un réseau de neurones bayésiens

La nécessité d'un lien avec les réseaux bayésiens(BNN) pour évaluer les incertitudes des prédictions des réseaux de neurones(NN)

Les limites des réseaux de neurones bayésiens(BNN) et la nécessité d'une combinaison "bayésiens-réseaux de neurones"(BNN et NN)

Comme l'explique Gal [2016], les réseaux de neurones(NN) ne permettent pas la prise en compte de l'incertitude dans des problèmes de regression ou de classification car ils ne sont que des fonctions deterministes. Les prédictions(outputs) issues de ces réseaux de neurones ne sont que vecteurs scores non normalisés ou des scalaires qui ne capturent pas l'incertitude. Ainsi, comme l'ont montré He et al. [2016] du centre de recherche de Microsoft, même si les réseaux de neurones présentent de bons résultats, l'évaluation des incertitudes posent toujours problème.

C'est pourquoi Wenzel et al. [2020] affirment que les structures de réseaux classiques présentés dans la section 2 ont été abandonnés au profit des réseaux de neurones bayésiens. Cependant, ces mêmes auteurs présentent les nombreuses limites des réseaux bayésiens.En effet, s'il est facile de mettre en place les réseaux de neurones bayésiens, il est difficile de déterminer l'inférence associée. L'inférence $p(Y/X)$ nécessite de calculer la probabilité conditionnelle(posterior) $p(\omega/X,Y) = \frac{p(Y/X,W)*p(W)}{p(Y/X)}$ qui ne peut être calculée analytiquement.

Pour Kendal and Gal [2017], une combinaison de réseaux de neurones bayésiens et de structures classiques(multiperceptron ou convolutif) reste la meilleure alternative.

La mesure de l'incertitude épistémique dans des structures neuronales classiques grâce à l'apprentissage bayésien

La nécessaire modification de l'entrainement pour determiner l'incertitude epistémique

Beal et al. [2003] ont montré que le dropout est applicable dans les réseaux de neurones bayésiens (BNN) pour prévenir le phénomène d'overfitting durant les phases d'entrainements et pour calculer l'incertitude epistémique en phase de validation en cachant certains poids du réseau avec une probabilité p.
Gal and Ghahramani [2015] ont prouvé que cette méthode baysienne est applicable à des réseaux de neurones classiques(type multiperceptron). La structure initiale d'un réseau de neurones(NN) est conservée tout en lui appliquant de couches de dropout. Pouvoir l'appliquer à tout réseau de neurones sans effectuer des changements majeurs est le grand avantage avancé par Lengerich et al. [2020]. Ce dernier rappelle qu'il faut tout de même modifier la fonction de coût afin d'apprendre au réseau à minimiser cette erreur epistémique pendant la phase d'entrainement. La fonction de coût modifiée est présentée dans l'équation suivante :

$$L(\theta, p = -\frac{1}{N}\sum_{i=1}^{N} logp(y_i/f_i^{\omega}(x_i)) + \frac{1-p}{2N}||\theta||^2 \tag{12}$$

Avec N le nombre de données d'entrainements, p la probabilité de dropout choisie par l'utilisateur, θ le paramètre de distribution.

L'incertitude epistémique calculée pendant la phase de validation

Mullins et al. [2016] expliquent que pour obtenir les T tirages des poids $\omega \approx q_\theta(\omega)$,

la couche de dropout est laissée active pendant la phase de validation et les T tirages de $f^{\omega}(x^*)$ seront différents.

Toujours Mullins et al. [2016], pour des problèmes de regression, la distribution des sorties du réseau de neurones est définie pour une même entrée par une gaussienne de moyenne et variance donnée par la variabilité des résultats de $f^{\omega}(x^*)$. La probabilité associée à cette distribution est : $p(y/f^W(x)) = N(f^W(x), \sigma^2)$ avec σ le bruit d'observation. L'espérance et l'incertitude associées à cette distribution sont determinées par :

$$E(y) \approx \frac{1}{T}\sum_{t=1}^{T} f^{\omega}(x) \tag{13}$$

$$V(y^*) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} f^{\omega}(x_t)^T f^{\omega}(x_t) - E(y)^T E(y) \tag{14}$$

Pour des problèmes de classification, les auteurs proposent que les sorties du réseau de neurones passent à travers une fonction softmax résultant dans un vecteur probabilité : $p(y/f^W(x) = Softmax(f^W(x))$. Cette inférence peut être approximée par :

$$p(y = c/x, X, Y) \approx \frac{1}{T}\sum_{t=1}^{T} Softmax(f^{\omega}(x)) \tag{15}$$

Cette incertitude liée à la probabilité p est calculée grâce à l'opérateur entropie : $H(p) = -\sum_{c=1}^{C} p_c log(p_c)$

Mise en évidence de l'erreur epistémique sur un cas simple

Pour illustrer cette méthode de l'évaluation des incertitudes épistémiques Brando et al. [2019] reprennent l'exemple de la regression de la droit y=x est reprise. Après modification de la structure d'un réseau de neurones, l'erreur epistémique est tracée sur la figure(8).

Comme attendu, l'incertitude entre $]-\infty; -3 \cup]3; +\infty[$ est très élévée puisque aucune donnée d'entrainement n'a été fournie dans ces zones.

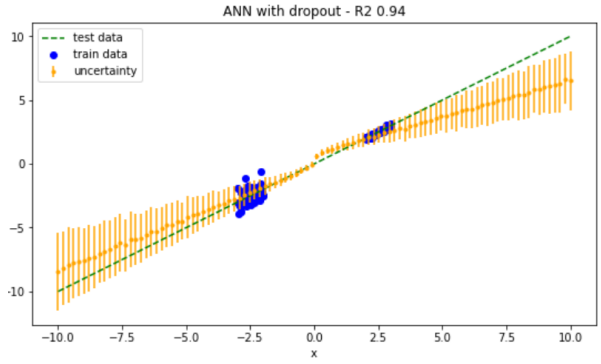

FIGURE 8 – Impact du MonteCarlo Dropout sur un réseau de neurones

La méthode présentée Mullins et al. [2016] permettant d'estimer l'incertitude aléatoire dans les réseaux de neurones ne permet pas de determiner l'erreur aléatoire hétéroscédastique.

Une modification des réseaux de neurones pour déterminer l'erreur hétéroscédastique

L'incertitude aléatoire est donc l'erreur non constante récupérée fournies aux réseaux de neurones. Kersting et al. [2007] expliquent que construire une méthode permettant d'évaluer l'incertitude hétéroscédastique est très important dans le cas où certaines données n'ont aucune cohérence avec le problème traité. Pour cela, il affirme qu'il faut modifier la phase d'entrainement du réseau.

La nécessité de la modification de l'entrainement pour estimer l'erreur hétéroscédastique

Dans la fonction de coût de l'équation (10), l'observation du bruit pourrait être fixée ou ignorée par les poids du réseau puisque l'erreur inhérente aux donnés σ est supposée constante dans le modèle calculant l'erreur epistémique. Pour prendre en compte cette variabilité dans l'erreur sur les données, Kersting et al. [2007] ont modifié la forme de la fonction de coût :

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\sigma(x_i)^2}||y_i - f(x_i)||^2 + \frac{1}{2}log(x_i)^2 \tag{16}$$

Mise en évidence de l'erreur aléatoire hétéroscédastique sur un cas simple

Brando et al. [2019] modifient la fonction de coût de leur réseau de la même manière que Kersting et al. [2007].

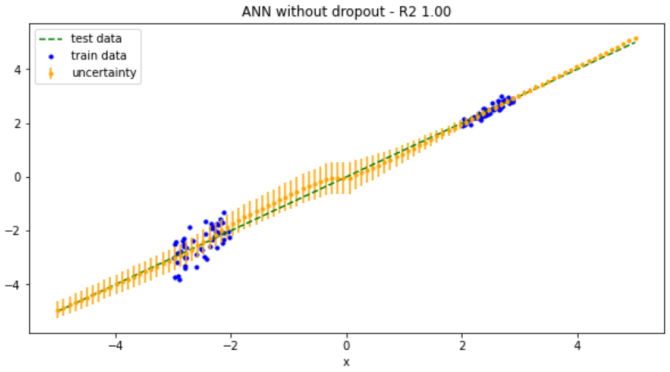

FIGURE 9 – Evaluation des incertitudes aléatoires cas d'une regression

Les résultats de la figure(9) étaient attendus. L'erreur aléatoire est plus élevée pour le nuage de points de gauche que pour le nuage de points de droite. Rajouter des données dans le nuage de droite, ne changerait rien. L'incertitude aléatoire heteroscédastique resterait toujours aussi élevée.

Le modèle va alors dans sa fonction de coût apprendre à minimiser l'erreur aléatoire hétéroscédastique. Néanmoins, cette approche proposée par Wang et al. [2019] ne permet pas de prendre en compte l'erreur épistémique puisqu'elle est propriété du modèle et non des données. Il est donc nécessaire de mettre en place un modèle permettant d'estimer à la fois l'erreur aléatoire et epistémique.

L'indispensable combinaison de l'erreur aléatoire et de l'erreur epistémique

Pour déterminer l'incertitude aléatoire et l'incertitude epistémique dans un même modèle, il est nécessaire selon Kendal and Gal [2017] de déterminer à la fois la distribution sur les sorties comme dans les réseaux bayésiens permettant de prédire un écart-type

associé à l'incertitude epistémique ainsi que l'incertitude aléatoire σ. Ces mêmes auteurs affirment qu'il faut d'abord modifier la structure du réseau de neurones.

Une essentielle modification de la structure neuronale pour estimer l'erreur hétéroscédastique et epistémique

Pour Hou et al. [2018] l'entrainement du réseau doit être modifié pour lui apprendre à prédire une variance sur les données qui lui sont fournies. La tête du réseau est divisée en deux, une pour la prédiction et l'autre pour la variance $f^{\omega}(x) = [\widehat{y}, \widehat{\sigma}]$ où f est un réseau de neurones composé de couches de dropout. Les valeurs observées en sortie du réseau comme précédemment vont suivre une loi normale. Cependant, selon les mêmes auteurs, le réseau ne va pas apprendre à prédire une variance par des données d'entrainement labellisées mais grâce à une modification de la fonction de coût.

Une modification nécessaire de la fonction de coût

Pritzel et al. [2020] proposent que la fonction de coût contienne deux composantes. La première est le résidu de la regression obtenu avec une technique d'inférence varationnelle de type MonteCarlo drop out permettant d'obtenir l'incertitude sur les paramètres. Le second est un terme de régularisation sur les erreurs contenues dans les données. Pour ce second terme, il n'est pas nécessaire d'avoir des 'labels' sur l'incertitude pour que le réseau apprenne à minimiser l'incertitude. Il suffit de superviser la tâche de regression. La variance σ^2 sera prédite par le réseau à partir de la fonction de perte. Ce second terme de regularisation évite au réseau de prédire une incertitude infinie pour tous les points.

En pratique, la fonction de coût est calculée avec le logarithme de la variance car cela évite des divergences de la variance σ^2. Cette nouvelle fonction de coût est exprimée de la manière suivante :

$$L(\theta) = \frac{1}{N}\sum_{i}\frac{1}{2}exp(-s_i)||y_i - f^{\omega}(x_i)||^2 + \frac{1}{2}s_i \tag{17}$$

Avec D le nombre de données d'entrainements,y_i la sortie du réseau, $\widehat{y_i}$ la valeur qu'il faut prédire(label), $\widehat{\sigma_i}$ la variance prédite par le réseau.

Pour résumer, l'incertitude inhérente à l'entrée x dans ce modèle combinant l'incertitude aléatoire et l'incertitude epistémique peut etre résumée dans l'équation suivante :

$$Var(x) \approx \frac{1}{T}\sum_{t=1}^{T}\widehat{x_t}^2 - (\frac{1}{T}\widehat{x_t})^2 + \frac{1}{T}\sum_{t=1}^{T}\widehat{\sigma_t}^2 \tag{18}$$

avec $y_t, \widehat{\sigma_t}^2$ une série de T outputs du réseau pour $\widehat{y_t}, \sigma_t^2 = f^{\widehat{\omega_t}(x)}$ pour des masques de dropout aléatoires $W_t \approx q_\theta(\omega)$

Le modèle présenté dans cette partie permet d'estimer les deux familles d'incertitude : epistémique et aléatoire. La sortie du réseau de neurones prédira donc une valeur numérique de l'incertitude. Néanmoins, l'interprétation de ce résultat n'a rien d'évident et il est nécessaire de mettre en place des méthodes pour analyser ces résultats.

Les méthodes d'analyse des résultats des incertitudes

Jiang [2007] explique que les modèles présentés dans les parties précédentes permettent d'estimer une incertitude modélisée par un écart-type.Toujours Jiang [2007],cette incertitude correspond à une valeur numérique et n'est pas interprétable directement par un opérateur. Il est possible à partir de valeurs numériques de l'évaluation de l'incertitude de comprendre si un réseau de neurones est certain ou non de ses prédictions.

L'analyse des résultats de l'incertitude

Un critère sur l'écart-type

Pour des problèmes de regression comme de classification, Ghoshal and Tucker [2020] imposent un critère sur l'écart-type pour savoir si une prédiction est certaine ou pas. En effet, le réseau pour chaque entrée de validation va retourner une prédiction avec un résultat d'incertitude. Ce résultat d'incertitude se modèlise par un écart-type $\sigma = \sigma_e + \sigma_a$ avec σ_e l'incertitude epistémique et σ_a l'erreur aléatoire. Ainsi, savoir quand cette valeur d'incertitude devient critique est totalement arbitraire.Ces mêmes auteurs affirment que l'utilisateur va devoir empiriquement choisir un seuil sur l'écart-type à partir duquel il considera que la prédiction renvoyée par le réseau n'est pas certaine. Ce seuil sur l'écart-type sera noté σ_s. Selon les mêmes auteurs, une prédiction sera jugée certaine lorsque l'écart-type associée est tel que $\sigma < \sigma_s$.

Ce critère sur l'écart-type permet de faire la dichotomie entre les prédictions certaines et incertaines. Steinbrener et al. [2020] expliquent que ce critère insuffisant et qu'il faut également s'interesser à la précision du réseau. Pour cela, ils mettent en place un outil appelé matrice de confusion.

La matrice de confusion

La matrice de confusion mise en place par Steinbrener et al. [2020]est un outil très utile pour avoir un outil visuel de l'estimation de l'incertitude. Elle permet de mettre en évidence les quatre cas de figures suivants :
Vrai et Certain(VC) : le résultat prédit par le réseau est bon et le réseau est certain $\sigma < \sigma_s$
Vrai et Incertain(VI) : le résultat prédit par le réseau est bon et le réseau est incertain. Autrement dit, le réseau est incertain de sa bonne prédiction.$\sigma < \sigma_s$
Mauvaise et Incertain(MI) : le résultat prédit est mauvais et le réseau est incertain. Autrement dit, le réseau n'est pas certain de sa mauvaise prédiction.$\sigma > \sigma_s$
Mauvaise et Certain(MC) :le résultat prédit correspond est et le résultat est certain. Autrement dit, le réseau est certain de sa mauvaise prédiction. $\sigma > \sigma_s$

En reprenant l'exemple de la classification de nombres écrits manuscritement que doit reconnaitre un réseau de neurones classiques,Steinbrener et al. [2020] mettent en évidence la matrice de confusion suivante.

	Quite Certain	Uncertain
correct	9609	297
wrong	14	80

FIGURE 10 – Exemple d'une matrice de confusion

Une majorité des résultats sont certains et corrects. Cependant, 14 entrées ont fait l'objet d'une mauvaise prédiction certaine. Ce sont les prédictions les plus dangereuses qu'il faut absolument comprendre. Dans ce cas, les 14 entrées de validation générant une mauvaise prédiction certaine sont récapitulées dans la figure 14.
Ces 14 entrées de validation pourront donner lieu à une modification du modèle initial ou à une relabellisation de certaines données d'entrainement.

Les modèles et analyse de l'incertitude ont été mis en évidence de manière théorique. La simulation de ces modèles sur des cas en "vie réelle".

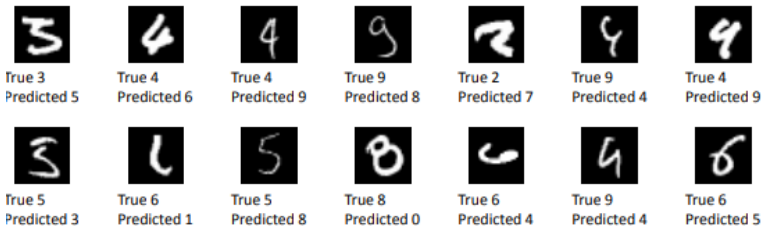

FIGURE 11 – Exemple d'une matrice de confusion

Les applications des méthodes d'évaluation des incertitudes des réseaux de neurones en "vie réelle" et leurs limites

De l'aide au diagnostic à la décision plus sécuritaire

Ortmaier et al. [2019] affirment que depuis quelques années, certains systèmes d'aide au diagnostic utilisant des réseaux de neurones atteignent des performances similaires à celles de certains experts dans certaines tâches comme la detection de pathologies en ophtalmologie ou en imagerie médicale.

Pour être utile dans l'aide au diagnostic, Ayhan et al. [2020] considèrent qu'il faut obtenir des mesures précises de l'incertitude des prédictions que ces réseaux de neurones fournissent. Ces systèmes utilisent souvent des réseaux de neurones convolutifs pour traiter de grandes bases de données contenant beaucoup d'images. Les modèles d'évaluation des incertitudes epistémiques et aléatoires ont permis de mettre en évidence les cas les plus compliqués et les plus incertains. Cela permet de faire appel aux experts dans ces cas précis. Une plus grande confiance dans ces systèmes d'aide au diagnostic composés de réseaux de neurones a été accordée depuis la mise en place de ces modèles d'évaluation des incertitudes.

De l'aide à la décision pour des bases de données plus faibles

Les systèmes d'aide au diagnostic ou à la prise de décision sont capables d'automatiser certaines tâches grâce à de larges bases de données d'entrainement. Malheureusement, la labellisation de larges bases de données pour entrainer les réseaux de neurones n'est pas toujours possible. Wu et al. [2020] expliquent que les experts n'ont pas toujours le temps d'effectuer cette démarche fastidieuse de labellisation. Ces modèles d'évaluation de l'incertitude présentent une alternative à ce problème. En effet, même si le réseau est entrainé avec des "petites bases de données", les méthodes d'évaluation des incertitudes permettront de montrer les cas les plus incertains. Cela permettra au médécin de "relabelliser" ces cas incertains et d'améliorer les performances du réseau de neurones.

Les limites et les verrous des connaissances des modèles d'évaluation des incertitudes

Les modèles d'évaluation des incertitudes ont permis :
-De rendre les systèmes d'aides au diagnostic plus fiables car ils permettent de mettre en évidence certaines limites des prédictions des réseaux
-De travailler avec des bases de données plus faibles quitte à "relabelliser" les données.

Néanmoins, ces modèles présentent certaines limites :
-Ils ne permettent pas d'évaluer les incertitudes dans le cas d'un apprentissage non supervisé
-Il n'existe aujourd'hui que le Monte Carlo drop out comme méthode efficace pour estimer

l'incertitude epistémique
-Ces modèles d'évaluation des incertitudes des réseaux de neurones semblent n'avoir été appliqués que dans un nombre restreint de domaines. Pourquoi ne pas envisager de telles applications en finance, en management des risques,dans l'industrie manufacturière ou en météorologie.
-Ces modèles ont été appliqués dans l'aide au diagnostic au médecine. La démarche ultérieure pourrait être de l'imagerie médicale réalisée en temps record.

Table des figures

Liste des tableaux

Références

Delen and Sharda. Artificial neural networks in decision support systems. *None*, 2008.

Thompson, Greenewald, Lee, and Manso. The computational limits of deep learning. *arXiv*, 2020.

Mannarswamy and Roy. Evolving ai from research to real life-some challenges and suggestions. *International Joint Conference on Artificial Intelligence*, 2019.

Gal. Uncertainty in deep learning. *Thesis*, 2016.

Dang, Lin, and Wang. Supervised learning in spiking neural networks : A review of algorithms and evaluations. *Neural Networks*, 2020.

Sardar, Abbot, Kundu, Aronow, Granada, and Giri. Impact of artificial intelligence on interventional cardiology : From decision-making aid to advanced international procedure assistance. *American College of Cardiology foundation*, 2019.

Xin and Wang. Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, 2019.

Kumar. Comparison of neural networks and regression analysis : A new insight. *Expert systems with applications*, 2005.

Rumelhart, Hinton, and Williams. Learning representations by back-propagating errors. *Enclyclopedia of Machine Learning*, 2011.

Feng and Lu. Performance analysis of various activation functions in artificial neural networks. *Journal of Physics*, 2019.

Lecun, Bengio, and Yoshua. Convulational networks for images, speech, and time-series. *Neural Computation*, 1997.

Rawat, Wang, and Jaskaran. Deep convulational neural networks for image classification : A comprehensive review. *Neural Computation*, 2017.

Liu, Fang, Zhao, Wang, and Zhang. Implementation of training convolutional neural networks. *None*, 2015.

Zhang, Martens, and Grosse. Fast convergence of natural gradient descent for overparametrized neural networks. *arXiv*, 2019.

Shen an Zhang, Zhou, and Xu. Investigation on performance of neural networks using quadrative relative error cost function. *IEEE Access*, 2019.

Ying. An overview of overfitting and its solutions. *Journal of Physics : Conference Series*, 2019.

Huang, Kroening, Ruan, Sharp, Sun, Thamo, Wu, and Yi. A survey of safety and trustworthiness of deep neural networks : Verification, testing, adversarial attack and defence, and interpretability. *None*, 2020.

Burt, Torosdagli, Khosravan, Raviprakash, Mortazi, Tissavirasingham, Hussein, and Bagci. Deep learning beyond cats and dogs : recent advances in diagnosing breast cancer with deep neural networks. *British journal of Radiology*, 2018.

Jessica Guynn. Google search sparks outrage. *USA Today*, 2016.

Habib, Ridella, and Quandt. Odi resume : Automatic vehicule control systems. *U.S Department of Transportation*, 2016.

Pirracchio, Cohen, Malenica, Chambaz, Cannesson, Lee, Resche-Rigon, and the ACTER-REA Research Group. Big data and targeted machine learning in action to assist medical decision in the icu. *Société française de réanimation*, 2018.

Tanno, Worrall, Kaden, Ghosh, Grussu, Bizzi, Sotiropoulos, Criminisi, and Alexander. Uncertainty modelling in deep learning for safer neuroimage enhancement :demonstration in diffusion mri. *NeuroImage*, 2021.

Namee, Cunningham, Byrne, and Corrigan. The problem of bias in training data in regression problem. *Artificial Intelligence in Medicine journal*, 2002.

Santoro. Artificial intelligence in medicine : limits and obstacles. *NIH : National Journal of Medicine*, 2017.

Chordia, Goyal, Lehmann, and Saar. High-frequency trading. *Journal of Financial Markets*, 2013.

Huang, Han, and Hussain. High-frequency trading strategy based on deep neural networks. *Lecture Notes in Computer Science(including subseries Lectures Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 2016.

Daniel. New technologies and market abuses : Outdated legal frameworks, short-falling reforms and new proposals. *None*, 2013.

Kirilenko, Kyle, Samadi, and Tuzun. The flash crash : High-frequency trading in an electronic market. *Journal of Finance*, 2017.

Kiureghian, Ditlevsen, Ove, and Der. Aleatory or epistemic ? does it matter ? *Structural Safety*, 2009.

Brando, Serrano, Ciprian, Maestre, and Vitria. Uncertainty modelling in deep networks : Forecasting short and noisy series. *None*, 2019.

Sensoy, Kaplan, Kandemir, Lance, Melih, and Murat. Evidential deep learning learning to quantify classification uncertainty. *arXiv*, 2018.

Beal, Ghahramani, and Matthew. The variational bayesian em algorithm for incomplete data : with application to scoring graphical model structure. *Oxford University press*, 2003.

MacKay. Bayesian methods for neural networks. *Neural Networks*, 2009.

Jospin, Buntine, Boussaid, Laga, and Bennamoun. Hands-on bayesian neural networks-a tutorial for deep learning users. *arXiv*, 2007.

James. Bayes theorem. *The Standford Encyclopedia of Philosophy*, 2003.

Jordan, Gharamani, Michael, Jaakkola, and Saul. An introduction to varational methods for graphical methods. *Machine Learning*, 1999.

Filippi, Cappé, and Garivier. Optimism in reinforcement learning and kullback-leibler divergence. *48th Annual Allerton Conference on Communication, Control, and Computing, Allerton*, 2010.

Laves, Ihler, Kortmann, Ortamaier, Tobias, and Heinrich. Calibration of model uncertainty for dropout variational inference. *arXiv*, 2020.

Denker and LeCun. Transforming neural-net output levels to probability distributions. *ATT Bell Laboratories journal*, 1990.

Gal and Ghahramani. Dropout as bayesian approximation : Representating model uncertainty in deep learning. *33rd International Conference on Machine Learning*, 2015.

Gal and Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *Cambdrige Engineering Department Journal*, 2016.

Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov. Dropout :a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

Kendal and Gal. What uncertainties do we need in bayesian deep learning for computer vision. *Advances in Neural Information Processing Systems*, 2017.

He, Zhang, Ren, and Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

Wenzel, Roth, Nowozin, Veeling, Swiatkowski, Tran, Mandat, Snoek, Salimans, Jenatton, and Nowozin. How good is the bayes posterior in deep learning neural networks really ? *arXiv*, 2020.

Lengerich, Xing, and Caruana. On dropout, overfitting, and interaction effects in deep neural networks. *arXiv*, 2020.

Mullins, Ling, Mahadevan, Sun, and Strachan. Separation of aleatory and epistemic uncertainty in probabilistic model validation. *Reliability Engineering and System Safety*, 2016.

Kersting, Plageman, Pfaff, and Burgard. Most likely heteroscedastic gaussian process regression. *ACM International Conference Proceeding Series*, 2007.

Wang, Li, Aertsen, Deprest, Ourselin, and Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolution neural networks. *Neurocomputing*, 2019.

Hou, Li, and Liang. Mixed aleatory/epistemic uncertainty analysis and optimization for minimum eedi hull form design. *Ocean Engineering*, 2018.

Pritzel, Lakshminarayanan, and Blundel. Simple and scalable predictive uncertainty estimation using deep ensembles. *DeepMind*, 2020.

Jiang. Uncertainty in the output of artificial neural networks. *None*, 2007.

Ghoshal and Tucker. Estimating uncertainty and interpretabilty in deep learning for coronavirus(covid-19) detection. *None*, 2020.

Steinbrener, Posch, and Pilz. Measuring the uncertainty of predictions in deep neural networks with variational inference. *Sensors*, 2020.

Ortmaier, Laves, Ihler, and Kahrs. Quantifying the uncertainty of deep learning-based computer-aided diagnosis for patient safety. *Current Directions in Biomedical Engineering*, 2019.

Ayhan, Kuhlewein, Aliyeva, Inhoffen, Ziemssen, and Berens. Expert-validated estimation of diagnostic uncertainty for deep neural networks in diabetic retinopathy detection. *Medical Image Analysis*, 2020.

Wu, Chen, Zhong, Wang, and Shi. Covid-al : The diagnosis of covid-19 with deep active learning. *Medical Image Analysis*, 2020.