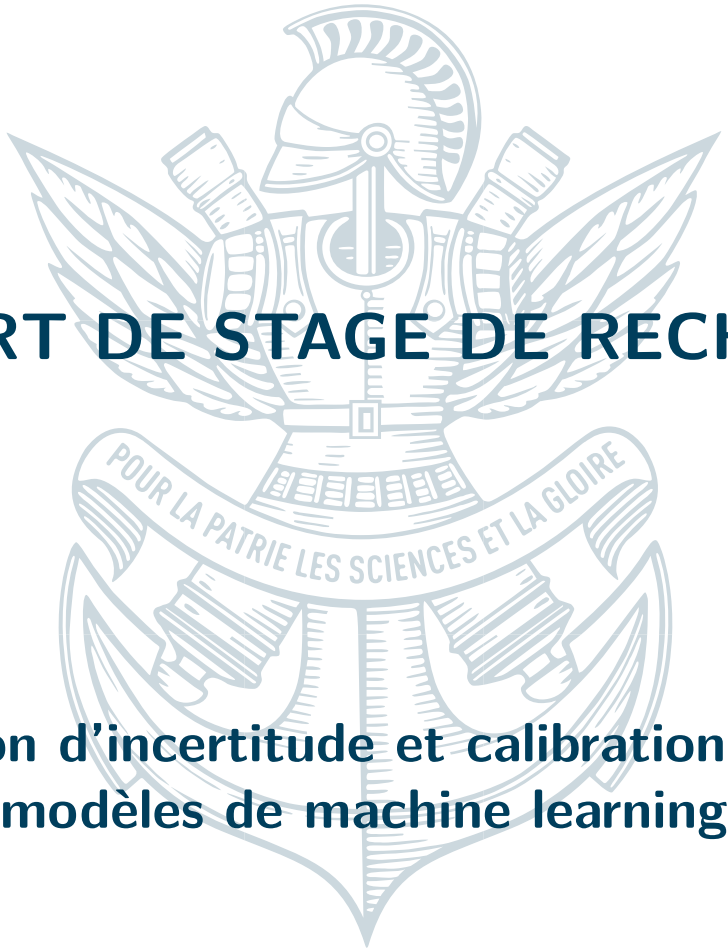


RAPPORT DE STAGE DE RECHERCHE

Prédiction d'incertitude et calibration dans les modèles de machine learning

19 juillet 2023

Nathan Herzhaft

TABLE DES MATIÈRES

DÉCLARATION D'INTÉGRITÉ RELATIVE AU PLAGIAT

Je déclare être l'auteur du présent travail, et certifie que tout ce qui a été emprunté à autrui y est proprement référencé et attribué à sa source.

Nathan Herzhaft
19/07/2023

REMERCIEMENTS

Avant de commencer ce rapport je tiens à remercier toute l'équipe de l'UQAM pour son acceuil chaleureux et bienveillant, qui m'a permis de me sentir à ma place, et être dans d'excellentes conditions pour réaliser ce stage.

En particulier, je remercie Arthur Charpentier, Philipp Ratz et François Hu pour leur encadrement sur toute la durée du stage, pour leurs conseils et pour m'avoir transmis leur envie et leur motivation.

Ils m'ont introduit au monde de la recherche en mathématiques, en me guidant à travers mes échecs comme mes réussites.

Ce fut un grand plaisir de travailler avec eux.

INTRODUCTION

En dernière année à l'École polytechnique, j'ai réalisé ce stage de recherche à l'université du Québec à Montréal, dans le département de mathématiques. Mon objectif pour ce stage était d'**approfondir les compétences acquises au cours de ma formation** en mathématiques appliquées et de commencer à me **spécialiser dans le domaine du machine learning**, sous un angle **théorique**.

Plus précisément, au sein de l'équipe d'Arthur Charpentier, j'ai pu appliquer mes connaissances basiques en machine et deep learning, en étudiant le domaine à travers **la théorie probabiliste**.

Le machine learning ne cesse de gagner en popularité ces dernières années, et les algorithmes prédictifs se répandent et sont utilisés dans tous les domaines scientifiques aujourd'hui (physique, économie, biologie, ...).

Un enjeu récent est **la quantification d'incertitude** dans un modèle prédictif. Dans des modèles de régression ou de classification, il est souvent nécessaire de prédire **des intervalles de confiance calibrés** afin d'améliorer la prise de décision. De tels algorithmes trouvent leur application en médecine, en météorologie ou encore en actuariat. De fait, ce sont des domaines ou **la variabilité de la prédiction est cruciale**, autant que la prédiction elle-même.

Les entreprises d'assurance en contact avec Arthur Charpentier, mon encadrant durant ce stage, sont particulièrement intéressées par ces recherches.

Toutefois, les recherches dans ce domaine s'intéressent rarement aux algorithmes de régression, comme nous l'expliquerons par la suite. Mon objectif était donc d'**étudier les méthodes de calibration pour les modèles probabilistes en régression**, en croisant les domaines du machine learning et des probabilités.

Dans ce rapport je commencerais par définir **le cadre global de la prédiction probabiliste**, l'état de l'art et les méthodes courantes pour l'implémentation de ces modèles. Dans un deuxième temps, je présenterai mes recherches sur la méthode du **gradient naturel** et la topologie des espaces probabilistes. Je présenterai ensuite les méthodes de **calibration** pour les modèles de machine learning. Enfin, j'amènerai le dernier sujet de mon stage, qui concernait les **réseaux de neurones** et l'étude d'une méthode en particulier, nommée **MC Dropout**.

1
PRÉDICTION PROBABILISTE : CADRE THÉORIQUE ET MÉTHODES HABITUELLES

1.1 Théorie Bayésienne

Dans le cadre courant, les algorithmes de machine learning ont pour rôle de faire des prédictions à partir de données d'entrée. Ces prédictions sont en général des valeurs scalaires (on parle alors de **régression**) ou des classes (on parle alors de **classification**).

Pour faire de telles prédictions, il est nécessaire d'**apprendre** la loi qui lie les prédictions aux données d'entrées, en utilisant des **données d'entraînement**.

Formellement, on note $\mathcal{X} \subset \mathbb{R}^n$ l'espace des données d'entrées, $\mathcal{Y}$ l'espace de sortie, X et Y des variables aléatoires à valeurs dans $\mathcal{X}$ et $\mathcal{Y}$ respectivement, représentant nos données.

Les données d'entraînement $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$ sont indépendantes et identiquement distribuées, de même loi que (X, Y).

L'objectif de la prédiction probabiliste est de prédire la distribution de $Y|X$. Cela s'oppose au machine learning classique, ou fréquentiste, où l'objectif est de prédire $\mathbb{E}[Y|X]$ pour minimiser l'erreur en moyenne.

La principale différence avec l'approche fréquentiste est que la prédiction est une distribution, et non une valeur scalaire.

Il faut reconsidérer les notions classiques venant de l'approche fréquentiste. Par exemple, que deviendrait une fonction de coût sachant que notre sortie est une ditribution et que l'on doit la comparer à une valeur théorique Y_i ? Nous ne manipulons plus les données directement, mais leur distribution, en supposant que les données sont des échantillons générés à partir de celle-ci.

Pour cela, le cadre formel est celui du **modèle de Bayes**. D'un point de vue probabiliste le problème revient à supposer que X et Y ont une distribution conjointe, et de chercher la distribution $Y|X$ en utilisant des échantillons, à savoir les données $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$.

Souvent, les modèles sont **paramétriques**, c'est à dire qu'il existe des paramètres θ régissant l'expression de la distribution prédite de $Y|X$. Ce sont ces paramètres que l'on **optimise** pendant l'entraînement.

L'objectif de l'entraînement est de trouver la loi liant θ aux données d'entrée, en utilisant les données d'entraînement $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$

L'exemple le plus simple est celui de la régression linéaire avec bruit gaussien constant. En notant $\theta = (\theta_0, \theta_1)$, on modélise la distribution $Y|X = x$ par une loi normale de moyenne $\theta_0 + \theta_1 x$ et de variance $\epsilon = cste$. L'objectif de l'entraînement est d'optimiser les paramètres θ pour améliorer la prédiction.

Contrairement à l'approche fréquentiste, il ne s'agit plus de trouver $\theta \in \Theta$ minimsant l'espérance du coût. Ici, il faut trouver une **distribution sur** Θ.

Formellement, en notant $\mathcal{D} = \{(X_i, Y_i)\}_{i \in [\![1,n]\!]}$, la formule de Bayes peut s'écrire :

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{1}$$

Il faut noter plusieurs points importants :

1. Le calcul de $p(\theta|\mathcal{D})$ nécessite d'avoir accès à $p(\theta)$. Le terme $p(\theta)$ correspond à une connaissance sur nos paramètres que l'on a **avant les données d'entraînement**. Cette distribution est nommée le **prior** du modèle. Il peut être modélisé comme on le souhaite suivant notre modélisation **a priori**. Le choix du prior est une question essentielle du formalisme bayésien, et nous aurons le temps d'y revenir plus en détail par la suite.

2. Le calcul nécessite également la connaissance de $p(\mathcal{D})$. Cela correspondrait à une connaissance sur les échantillons de données que nous allons recevoir. On peut le calculer en intégrant sur tout le prior : $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)\, d\theta$.

3. Enfin, le facteur essentiel est $p(\mathcal{D}|\theta)$. Dans le cadre bayésien on nomme ce terme **évidence**, ou **likelihood**. C'est la probabilité qu'on aurait d'observer un tel jeu de données $\mathcal{D}$ si notre loi théorique était paramétrée par θ. Ce facteur mesure la **vraisemblance** d'un paramètrage θ : plus il est élevé, plus θ est un bon candidat pour paramétrer notre distribution de sortie.

4. Le terme $p(\theta|\mathcal{D})$ est le **posterior**. C'est la connaissance que l'on a sur nos paramètres **après observation des données d'entraînement**.

Il est important de remarquer que l'approche fréquentiste consiste à maximiser la vraisemblance $p(\mathcal{D}|\theta)$ pour trouver un θ optimal selon une certaine fonction de coût. Dans l'approche bayésienne, on confronte la vraisemblance à une connaissance *a priori* sur θ pour quantifier nos connaissance *a posteriori* après l'entraînement. De fait, il existe de nombreuses équivalences entre les fonctions de coût fréquentistes et les priors bayésiens. Ces objets modélisent les mêmes concepts dans des cadres différents.

Enfin, après avoir trouvé la distribution sur nos paramétrisations, on peut calculer la distribution des prédictions sachant les données d'entrées. Cela nécessite d'intégrer sur toutes les paramétrisations possibles, c'est la **marginalisation**.

$$p(y|X, \mathcal{D}) = \int p(y|X, \theta)p(\theta|\mathcal{D})\, d\theta \tag{2}$$

Pour résumer, l'école bayésienne s'oppose à l'approche fréquentiste et **utilise les probabilités comme moyen de quantifier un degré de connaissance**. L'entraînement bayésien

consiste à passer de connaissances *a priori* à des connaissances *a posteriori*, et ce processus se nomme l'**inférence du modèle**.

Le machine learning bayésien n'est pas le seul moyen de faire de la prédiction de distribution, mais c'est cette théorie qui fournit le cadre de base pour tous ces modèles.

1.2 Limites du formalisme bayésien et autres modèles

Le machine learning bayésien fournit un cadre exact pour la prédiction d'incertitude. En pratique toutefois, il est très difficile d'appliquer le cadre défini plus haut.

De fait, le procédé de **marginalisation** vu précedemment est très coûteux. De nombreux calculs d'intégrales sont nécessaires à chaque étape de l'entraînement pour actualiser notre degré de connaissance sur les paramètres du modèle.

On essaye parfois de faciliter les procédés de marginalisation par des hypothèses simplificatrices. On peut par exemple approcher le posterior par des méthodes de Monte Carlo pour éviter de déterminer analytiquement les distributions.

Ces procédés sont des méthodes de *variational inference*. Ces méthodes regroupent les techniques habituelles utilisées en pratique pour approcher le calcul du posterior et pouvoir prédire des distributions en un temps raisonnable.

Pour prédire des distributions sans passer par le cadre bayésien, on peut contourner le problème et entraîner notre modèle à prédire directement les paramètres d'une loi donnée.

Par exemple, plutôt que d'entraîner le modèle à prédire $\hat{y} \in \mathcal{Y}$, on l'entraîne à prédire $(\hat{\mu}, \hat{\sigma})$ les paramètres d'un loi normale. Il faut alors adapter la fonction de coût, passer d'une forme $f(\hat{y}, y)$ à $f(\hat{\mu}, \hat{\sigma}, y)$.

Cette méthode a l'avantage d'être entièrement **déterministe** pendant l'entraînement. Schématiquement, cela revient à remplacer nos variables cibles par des paramètres et à changer la fonction de coût pour optimiser ces paramètres en accord avec notre objectif.

Cette technique entraîne de nombreuses subtilités pour adapter correctement l'entraînement du modèle. La descente de gradient, notamment, doit être modifiée pour correspondre à l'espace des prédictions (μ, σ). Nous reviendrons très longuement sur ce point dans la prochaine partie.

1.3 La notion de calibration

Il existe différents moyens pour juger un modèle prédictif déterministe après son entraînement. Les plus classiques sont des critères tels que la **précision** pour les modèles de classification ou l'**erreur carrée moyenne** pour les modèles de régression.

Pour un modèle qui prédit une distribution, il nous faut des notions plus complexes. En effet, il faut connaître la vraie distribution $Y|(X = x)$ pour chaque x, et un moyen de la comparer aux ditributions prédites. C'est ce qu'on appelle la **calibration**.

Commençons par un exemple classique. Soit m un modèle de classification à deux classes. A chaque $x \in \mathcal{X}$, m assigne un probabilité $m(x)$ d'appartenir à la classe 1, et une probabilité $1 - m(x)$ d'appartenir à la classe 0. La distribution est celle d'une variable de Bernoulli, il est facile de vérifier si le modèle est **calibré** : pour chaque p, il faut que $Y|(m(X) = p)$ suive une distribution de Bernoulli de paramètre p.

On étudie donc la fonction suivante :

$$C : p \to C(p) = \mathbb{E}[Y|m(X) = p]$$

Plus cette fonction est proche de la fonction identité $Id : p \to p$, meilleur est notre modèle en terme de calibration. C'est pourquoi on représente souvent des **courbes de calibration**, comme montré sur la Figure 1.

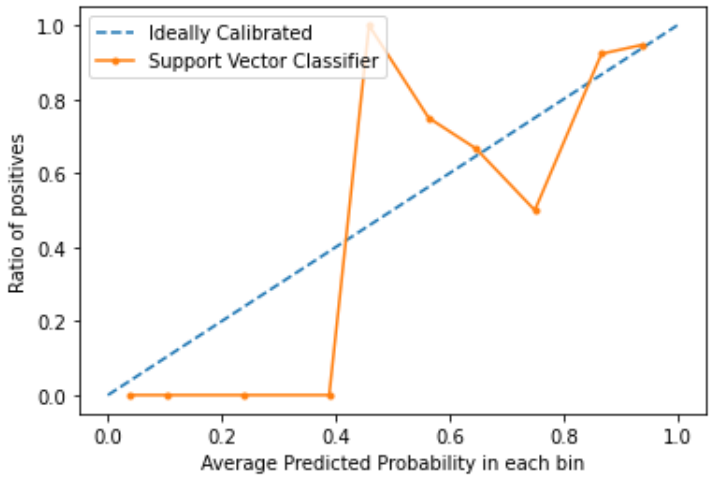


Figure 1 – Exemple de courbe de calibration obtenue pour un modèle de *Support Vector Classifier*

Sur cet exemple, le modèle n'est pas bien calibré.

Sur l'axe des abscisses, on parle de *bins*. Ce terme désigne un groupe de données d'entrée dont la prédiction par le modèle est proche d'une valeur donnée. Le regroupement par bin est nécessaire pour les calculs d'espérance, un faible nombre de valeurs ne suffit pas à estimer $\mathbb{E}[Y|m(X) = p]$, donc il faut regrouper dans le voisinage de chaque p.

La notion de calibration est largement étudiée dans le cadre de la classification. Comme vu précedemment, la modélisation par une variable de Bernoulli est aisée et permet l'étude des outils tels que les courbes de calibration. C'est pourquoi la calibration en classification est un sujet très documenté et bien connu.

Toutefois, il n'en va pas de même pour la régression. En général, on ne peut pas résumer la distribution à un seul paramètre, et certaines distributions ne sont même pas paramétriques.

La prédiction en probabilité est un grand sujet de recherche, et nous reviendrons longuement sur ces notions de calibration par la suite.

Pour résumer, la **calibration** est une notion permettant de juger la qualité d'un modèle prédictif en distribution. Cependant, son étude est plus complexe que les *loss functions* du cadre déterministe et, à ce jour, il n'existe pas de moyen conventionnel établi pour vérifier si un modèle de régression en distribution est bien calibré.

2
TOPOLOGIE DE L'ESPACE DES DISTRIBUTIONS ET GRADIENT NATUREL

2.1 Prédiction paramétrique

2.1.1 • Définition

La première partie de mes recherches portait sur les méthodes d'entraînement de modèles probabilistes pour la régression, et plus précisément sur la **prédiction paramétrique**. La prédiction par les paramètres désigne le fait de prédire la distribution $Y|(X = x)$ en l'identifiant à des paramètres $\omega(x)$.

Dans le cas d'une loi normale par exemple, on suppose qu'il existe des fonctions μ et σ tels que $Y|(X = x)$ suive une loi $\mathcal{N}(\mu(x), \sigma(x))$, et l'objectif du modèle est d'apprendre la fonction $x \to (\mu(x), \sigma(x))$. Notre but est toujours de prédire une distribution, mais on choisit de la représenter par ses paramètres.

Il est important de comprendre que l'entraînement est ici **déterministe** et non **bayésien** : on cherche à trouver une fonction de coût $\mathcal{S}(\omega, y)$ qui tend à améliorer les paramètres ω selon un critère donné.

Le choix de la fonction de coût détermine ce que l'on considère être une bonne prédiction de distribution. La subtilité est que nous n'observons que les réalisations Y sans connaître leur distribution, et il est donc compliqué de juger si une distribution est plus adaptée qu'une autre à réaliser un Y_i donné.

En général, on choisit de dire qu'une distribution est meilleure qu'une autre selon un critère de **vraisemblance**. Dans l'exemple d'un paramétrage gaussien, les paramètres ω_1 sont meilleurs que ω_2 au vu de l'évènement $(Y = y, X = x)$ lorsque :

$$\mathcal{N}(y \mid \omega_1(x)) \geq \mathcal{N}(y \mid \omega_2(x))$$

Où $\mathcal{N}(y \mid \omega)$ renvoie à la fonction de densité de la loi normale dont les paramètres sont ω, évaluée en y.

Ainsi, une fonction de coût $\mathcal{S}(\omega, y)$ adaptée serait ici la *negative log-likelihood*, utilisée également en classification, dont l'expression avec une loi normale est :

$$\mathcal{S}(\omega, y) = -log(\ \mathcal{N}(y \mid \omega)\ )$$

Avec $\omega = (\mu, \sigma)$ dans notre cas.

Toutefois, d'autres critères que la vraisemblance peuvent être considérés en changeant la fonction de coût.

2.1.2 • Topologie de l'espace paramétrique

En représentant une distribution par ses paramètres, on change la topologie de notre espace. En effet, au lieu d'avoir une fonction d'erreur qui compare notre distribution à une réalisation : $\mathcal{S}(\mathbb{P}_\omega, y)$, on identifie la distribution à ses paramètres : $\mathcal{S}(\omega, y)$.

On peut identifier une distribution $\mathbb{P}_\omega$ à ses paramètres ω, mais cela modifie la topologie de l'espace considéré. La direction de plus forte descente dans l'espace paramétrique $\mathcal{P}$ ne correspond pas nécessairement à la direction de plus forte descente dans l'espace des distributions $\mathcal{D}$.

Ainsi, on change les dynamiques d'entraînement par descente de gradient car les déplacements selon $\nabla_\mathcal{P}\mathcal{S}(\omega, y)$ ne correspondent pas à des déplacements selon $\nabla_\mathcal{D}\mathcal{S}(\omega, y)$. Si un déplacement $\omega \to \omega + d\omega$ fait du sens dans un espace euclidien, on ne peut identifier ce déplacement à $\mathbb{P}_\omega \to \mathbb{P}_{\omega+d\omega}$.

En terme topoligque, la bijection $\mathbb{P}_\omega \to \omega$ n'est pas un isomorphisme. Il est donc nécessaire de considérer des notions de distance entre distributions pour ensuite définir un gradient dans l'espace des distributions.

2.2 Gradient Naturel

2.2.1 • Définition et enjeux

On utilise la représentation paramétrique pour simplifier les calculs et l'entraînement. Mais lorsque l'on effectue une descente de gradient on cherche à parcourir directement l'espace des distributions : même si l'on choisit de représenter par des paramètres, ce sont toujours les distributions qui nous intéressent.

Ce sont ces observations qui nous ont conduit à étudier le *Natural Gradient*, une opération de changement d'espace du gradient pour traduire le déplacement $\nabla_\mathcal{D}\mathcal{S}(\omega, y)$ dans l'espace paramétrique [1].

L'enjeu principal est que l'espace des distributions $\mathcal{D}$ n'est pas euclidien. Sinon, il serait aisé de déterminer une matrice de changement de base et de modifier le gradient. $\mathcal{D}$ possède en fait une structure de **variété riemannienne**, c'est à dire que l'espace est **localement euclidien**. Plus précisément, l'espace tangent en chaque point peut être muni d'un produit scalaire, dépendant du point considéré.

Ainsi, pour passer de $\nabla_\mathcal{P}\mathcal{S}(\omega, y)$ au gradient naturel $\nabla_\mathcal{D}\mathcal{S}(\omega, y)$, il faut considérer la topolo-

gie induite par $\mathcal{S}$ localement dans $\mathcal{D}$, pour trouver la matrice de changement de base adaptée. L'objectif est de trouver $M(\mathcal{S}, \omega)$ telle que :

$$\nabla_{\mathcal{D}}\mathcal{S}(\omega, y) = M(\mathcal{S}, \omega)\nabla_{\mathcal{P}}\mathcal{S}(\omega, y) \tag{3}$$

La difficulté est que cette transformation dépend entièrement du choix de $\mathcal{S}$. Philip Dawid étudie ces propriétés de *geometry of proper scoring rule* [2]. La détermination de M par propriété topologique dépasse le cadre de ces recherches. Le résultat important est que dans le cas usuel de la *log-likelihood*, on a l'identité suivante :

$$\nabla_{\mathcal{D}}\mathcal{S}(\omega, y) = \mathcal{I}_{\mathcal{S}}(\omega)^{-1}\nabla_{\mathcal{P}}\mathcal{S}(\omega, y) \tag{4}$$

Où $\mathcal{I}_{\mathcal{S}}(\omega)$ est **la matrice d'information de Fisher**, objet bien connu de la théorie bayésienne :

$$\mathcal{I}_{\mathcal{S}}(\omega) = \mathbb{E}_{y\sim\mathbb{P}_\omega}[\quad \nabla_P\mathcal{S}(\omega, y)\nabla_P\mathcal{S}(\omega, y)^T \quad] \tag{5}$$

Ainsi, à partir du calcul classique de $\nabla_{\mathcal{P}}\mathcal{S}(\omega, y)$, on peut calculer $\nabla_{\mathcal{D}}\mathcal{S}(\omega, y)$ puis effectuer le déplacement de ω selon la bonne direction.

2.2.2 • Performances

Après l'exploration de ces concepts, nous avons mis en oeuvre quelques expériences pour observer l'efficacité et les propriétés du gradient naturel.

Nous voulions observer deux propriétés en particulier. Tout d'abord vérifier que le gradient naturel converge plus rapidement vers le point minimum. Ensuite, sur un aspect plus théorique, nous avons essayé d'observer les trajectoires empruntés par les deux algorithmes dans l'espace paramétrique $\mathcal{P}$ pour visualiser la différence dans les dynamiques d'apprentissage.

Premièrement, on étudie la vitesse d'entraînement. Nous pouvons générer un dataset $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$ régi selon une loi $x \to \omega(x) = (\mu(x), \sigma(x))$ arbitraire, comme par exemple $\mu(x) = x, \sigma(x) = x^2$. On utilise des algorithmes de gradient boosting, un avec le gradient naturel et l'autre avec le gradient ordinaire.On regarde ensuite la courbe de la *loss function* selon les itérations pour chacun des algorithmes, et on obtient les résultats suivants :

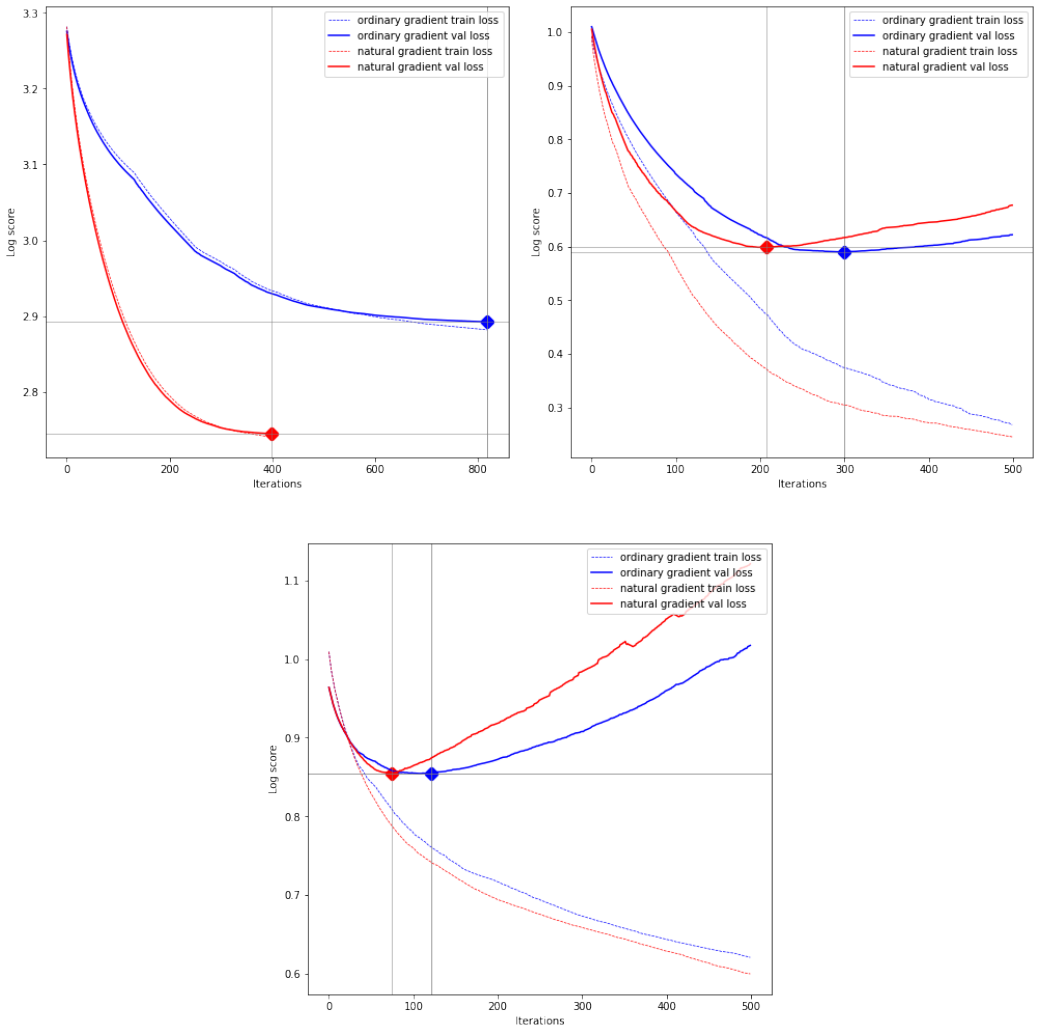


Figure 2 – Comparaison des vitesses d'entraînements des algorithmes utilisant le gradient ordinaire et le gradient naturel, pour différentes fonctions cibles $x \to \omega(x)$. En rouge, le gradient naturel et en bleu le gradient ordinaire. Les points minimum sont marqués d'un losange.

Les résultats sont très parlants : l'algorithme utilisant le gradient naturel converge plus vite vers son point minimum. De plus, ce point minimum est parfois meilleur (log score plus faible) que celui de l'algorithme ordinaire. Cela s'explique par le fait que le gradient ordinaire est plus suceptible de s'arrêter à un minimum local de la fonction de coût, car il n'emprunte pas la trajectoire optimale de minimisation.

La deuxième expérience visait à visualiser la trajectoire suivie par chacun des algorithmes pour observer les différences dans les dynamiques.

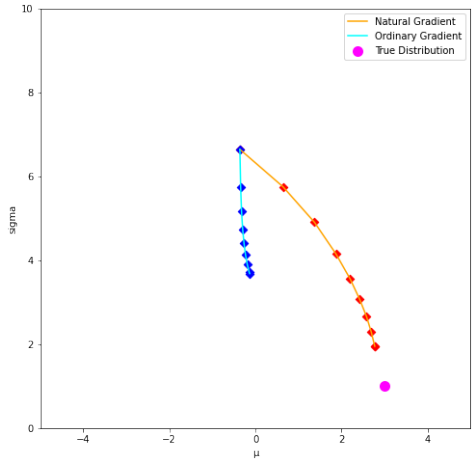
Pour cela, on reprend nos dataset $\{(X_i, Y_i)\}_{i \in [\![1,n]\!]}$ générés selon une fonction cible arbitraire. Nous pouvons également essayer de choisir un autre paramétrage qu'une loi normale, comme

par exemple $\omega = (\alpha, \beta)$ pour une loi gamma. On choisit ensuite un $x_0 \in \mathcal{X}$ et des paramètres $\omega_0 = \omega(x_0)$ associés.

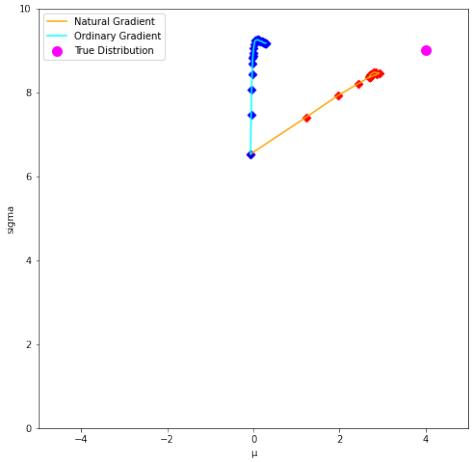
A chaque étape de l'entraînement, on affiche les prédictions $\hat{\omega_0}$ faites par les algorithmes pour l'entrée x_0. Ces prédictions devraient converger vers ω_0 au cours de l'entraîenement.

On peut représenter l'espace $\mathcal{P}$ ainsi que le point cible ω_0, et observer la trajectoire définie par les prédictions à chaque étape. Selon, la théorie, l'algorithme utilisant le natural gradient devrait prendre une trajectoire relativement directe et rapide, tandis que le gradient ordinaire devrait emprunter un chemin plus hasardeux.

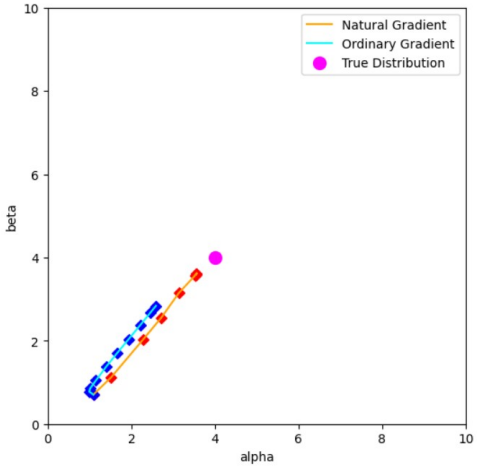
En faisant l'expérience pour différentes lois $x \to \omega(x)$ et différents points cibles ω_0, on obtient ces résultats :



(a) Modélisation par une loi normale



(b) Modélisation par une loi normale



(c) Modélisation par une loi gamma

Figure 3 – Observations des trajectoires de prédiction des deux algorithmes. Le code couleur est le même que précedemment. Chaque losange représente une étape de l'entraînement. Les expériences sont faites pour des paramétrisations et des points cibles variables.

A nouveau, on observe clairement l'efficacité du gradient naturel, qui converge plus vite et plus près du point objectif dans tous les cas. Cette expérience nous a permis d'avoir une représentation visuelle de cette dynamique.

2.2.3 • Lien vers la calibration

L'intérêt du *Natural Gradient* est d'améliorer la dynamique de l'entraînement en trouvant plus efficacement les minima de la fonction de coût dans l'espace des distributions. Une expérience inspirée de [1] nous a permis d'illustrer ce principe en visualisant les distributions prédites en comparaison au dataset d'entraînement.

A nouveau, on détermine à l'avance des fonctions $x \to \mu(x)$ et $x \to \sigma(x)$ qui nous permettent de générer X uniformément puis Y selon $\mathcal{N}(\mu(X), \sigma(X))$. En répétant ce procédé n fois, on obtient nos données d'entraînement $\{(X_i, Y_i)\}_{i \in [\![1,n]\!]}$.

On entraîne ensuite nos deux algorithmes en parallèle, mais cette fois on s'intéresse à une représentation des prédictions dans $\mathcal{X} \times \mathcal{Y}$. Il s'agit d'afficher nos données d'entraînement par des points, et d'afficher par dessus la courbe des fonctions prédites $x \to \hat{\mu}(x)$ et $x \to \hat{\mu}(x) \pm \hat{\sigma}(x)$. Cette représentation nous permet d'observer clairement si la distribution prédite par l'algorithme semble être la bonne. En d'autres termes, on peut estimer visuellement la **calibration** du modèle.

Les résultats sont visibles ci-dessous :

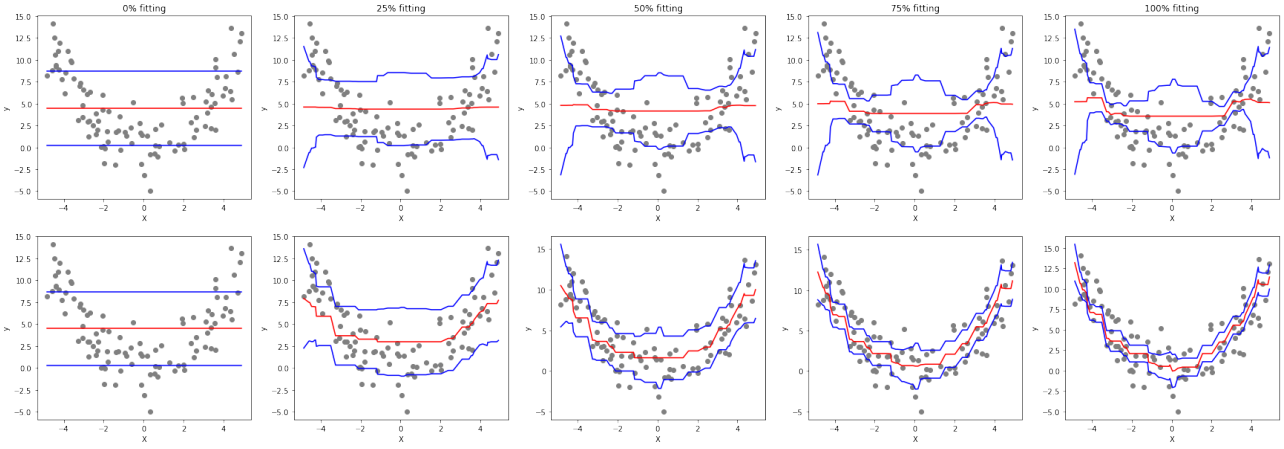


Figure 4 – Phases d'apprentissage (de gauche à droite) de deux modèles pour $\mu(x) = x^2$ et $\sigma(x) = \sigma_0$. Les modèles utilisés sont des modèles de gradient boosting. La ligne du dessus représente un modèle utilisant un gradient ordinaire, et celle du dessous un modèle utilisant le natural gradient. On a représenté le dataset par des points gris, les prédictions pour μ en rouge, et les prédictions pour $\mu \pm \sigma$ en bleu.

On remarque que l'algorithme utilisant le gradient classique ne parvient pas à ajuster la moyenne en fonction de X, qui reste donc quasi-constante peu importe la donnée d'entrée. On observe le même comportement peu importe la loi choisie pour $(\mu(x), \sigma(x))$. Pour compenser, l'algorithme cherche à augmenter la variance prédite pour les échantillons écartés de la moyenne, car c'est le seul moyen d'augmenter la vraisemblance et donc de diminuer la fonction de coût.

Il semblerait que l'algorithme utilisant le gradient classique fonctionne pour l'ajustement de la variance mais pas celui de la moyenne. Toutefois, prédire la variance n'a du sens que si la moyenne est déjà correcte. Notre algorithme, bien qu'affichant un bon *log score*, prédit des distributions très éloignées des véritables distributions pour chaque X : c'est un modèle de régression **mal calibré**.

C'est ainsi que vient la question de la calibration en régression. L'enjeu est de trouver une méthodologie analogue à la courbe de calibration en classification pour attester de la calibration d'un modèle. En effet, l'expérience précédente nous a permis d'observer la mauvaise calibration du modèle utilisant le gradient ordinaire mais nous avons besoin d'outils plus concrets pour juger nos modèles prédictifs.

3
CALIBRATION DES MODÈLES PROBABILISTES EN RÉGRESSION

3.1 Construction d'un formalisme

Pour définir une méthologie semblable au cas de la classification, nous avons besoin d'une fonction de vérification, analogue à $C : p \to C(p) = \mathbb{E}[Y|m(X) = p]$.

L'objectif de cette fonction est de **vérifier** si la distribution prédite est bien celle observée en pratique. Pour une variable de Bernoulli, vérifier la distribution revient à calculer la moyenne de Y pour voir si elle correspond au paramètre prédit.

Nos recherches nous ont mené vers l'article *Distribution Calibration for Regression* [6], introduisant un formalisme pour la régression. Étant donné qu'on ne peut plus vérifier notre distribution à l'aide d'un simple paramètre, cet article introduit de la calibration par régression de la fonction quantile. Toutefois, l'entraînement de la fonction quantile est extrêmement coûteux, et nous avons souhaité poursuivre nos recherches sur l'approche paramétrique et ses implications, en lien avec les études précédentes.

Dans le cas d'une loi normale, les paramètres prédits s'interprètent comme la moyenne et la variance de la distribution. Nous avonc donc pensé en premier temps à des fonctions de vérification de cette forme :

$$C_1(\mu) = \mathbb{E}[\, Y \mid \hat{\mu}(X) = \mu \,] \tag{6}$$

$$C_2(\sigma) = \sqrt{\mathbb{E}[\, (Y - \bar{Y})^2 \mid \hat{\sigma}(X) = \sigma \,]} \tag{7}$$

Où $(\hat{\mu}, \hat{\sigma})$ sont les prédictions du modèle.

On s'aperçoit vite d'un problème : si C_1 est éloigné de la fonction identité (i.e si le modèle est mal calibré pour μ), quel est le sens de $\bar{Y}$ dans (7) ?

En fait, c'était déjà ce problème que nous observions avec l'algorithme de gradient boosting classique, comme vu précedemment.

Il vient donc qu'on ne peut pas séparer nos paramètres dans la fonction de vérification. Nous avons besoin d'une fonction $C : \mathbb{R}^2 \to \mathbb{R}^2$ qui vérifie chaque couple de paramètre à la fois :

$$C(\mu, \sigma) = \left(\mathbb{E}[Y|m(X) = (\mu, \sigma)], \quad \sqrt{\mathbb{E}[Y^2|m(X) = (\mu, \sigma)] - E[Y|m(X) = (\mu, \sigma)]^2} \right) \tag{8}$$

Où on a noté m la prédiction du modèle $m = (\hat{\mu}, \hat{\sigma})$.

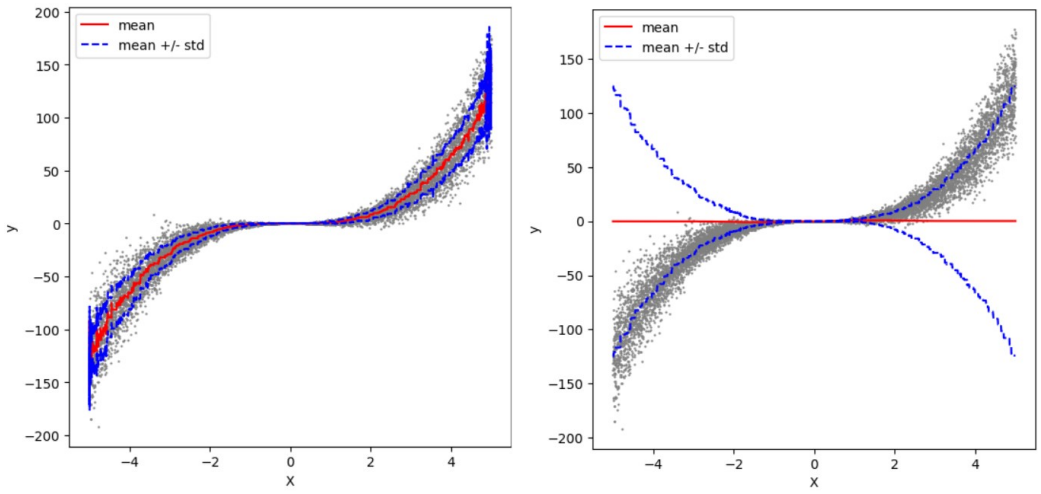
Cette fonction utilise des estimateurs par méthode des moments, comme pour le cas de la classification.

Le modèle est bien calibré si la fonction C est proche de la fonction identité de $\mathbb{R}^2$ vers $\mathbb{R}^2$.

3.2 Exemple et applications

On cherche maintenant à mettre en oeuvre ce formalisme nouvellement établi. Pour cela, on génère comme dans les expériences précédentes un dataset $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$, généré par des fonctions $\mu : x \to \mu(x)$ et $\sigma : x \to \sigma(x)$ prédéfinies. On choisira ici $\mu(x) = x^3$ et $\sigma(x) = x^2$. On rappelle que dans un cadre concret, nous n'aurions pas accès à l'expression de nos fonctions.

Pour tester notre méthodologie, nous avons besoin d'un modèle bien calibré et l'autre non, nous choisissons donc les modèles de gradient boosting par gradient naturel et par gradient ordinaire, utilisés précedemment. Après entraînement on obtient des prédictions que l'on peut représenter comme ceci :



(a) Prédictions en utilisant le gradient naturel

(b) Prédictions en utilisant le gradient classique

Figure 5 – Prédictions de deux modèles sur un dataset $\{(X_i, Y_i)\}_{i\in[\![1,n]\!]}$, avec $\mu(x) = x^3$ et $\sigma(x) = x^2$ On a représenté le dataset par des points gris, les prédictions pour μ en rouge, et les prédictions pour $\mu \pm \sigma$ en bleu. Les résultats sont semblables à ceux obtenus précedemment dans l'étude du *natural gradient.*

On s'intéresse, après entraînement, à la fonction C définie précedemment, nous permettant

de vérifier la calibration de notre modèle. Pour chaque couple (μ, σ), on sélectionne les (X_i, Y_i) tels que $m(X_i) \approx (\mu, \sigma)$, et on calcule empiriquement les deux estimateurs $\mathbb{E}[Y|m(X) = (\mu, \sigma)]$ et $\sqrt{\mathbb{E}[Y^2|m(X) = (\mu, \sigma)] - E[Y|m(X) = (\mu, \sigma)]^2}$.

Nous sommes obligés de considérer des **voisinages** aux alentours des points (μ, σ), sans quoi nous aurions un seul couple (X_i, Y_i) dans chaque cas, ce qui n'est pas suffisant pour les estimateurs empiriques. Rappelons qu'on considère aussi des voisinages autour de p pour la calibration en classification. Dans chaque cas cela nécessite des conditions de régularité sur notre modèle, que nous n'étudions pas ici et que l'on suppose vérifiées.

Toujours en se basant sur les méthodes définies pour la classification, on cherche à représenter C pour voir si elle est proche de la fonction identité. Toutefois, le cadre est plus complexe maintenant que C est à valeurs de $\mathbb{R}^2$ dans $\mathbb{R}^2$, il nous faudrait une visualisation en 4 dimensions.

Nous pouvons essayer de changer notre visualisation de C. **Nous pouvons représenter les antécédents et les images de C sur un même plan, dans des couleurs différentes**. Voici une telle représentation :

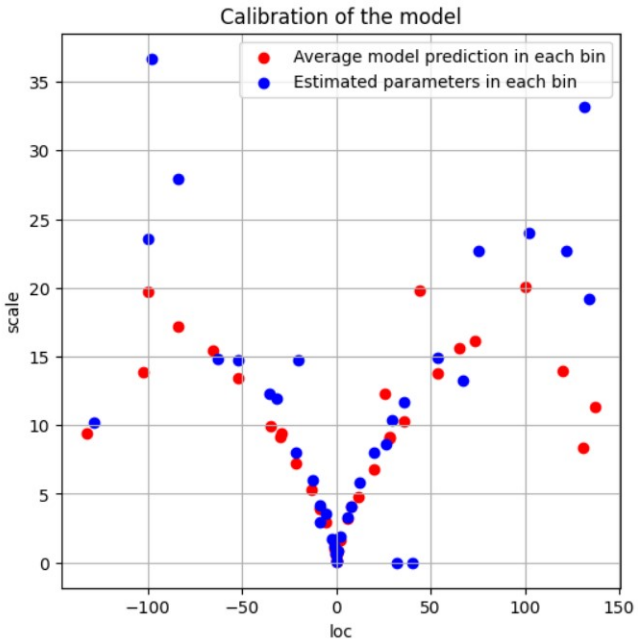


Figure 6 – Représentation de C sur un seul plan $\mathbb{R}^2$. Chaque point rouge est un antécédent et chaque point bleu une image par C.

Nous avons ici représenté la fonction C pour le modèle utilisant le natural gradient. Il faut comprendre que chaque paire de point correspond à un (μ, σ), le point rouge est la moyenne des prédictions faites pour les X_i dans ce "bin" (c'est-à-dire les X_i dont l'image par le modèle est dans le voisinage de (μ, σ)), et le point bleu est l'estimation empirique de l'image par C faite à partir de ces mêmes X_i.

Comme dans le cas de la classification, cette représentation ne fait pas apparaître la dé-

pendance en X, et ne considère que l'espace paramétrique. La construction de ce graphique utilise les données X_i, en plaçant des points $(\mu(X), \sigma(X))$ pour chaque sous-ensemble (ou "bin"). C'est la même idée que pour les « espaces de phases » en physique, où l'on représente les points $(X(t), V(t))$ pour chaque t.

On remarque facilement que les points bleus et rouges suivent une tendance similaire, ce qui laisse à penser que notre fonction C est proche de la fonction identité (surtout pour les μ assez faibles). En effet, une telle représentation de la fonction identité dans ce plan nous donnerait deux courbes superposées (une rouge et une bleue). Plus les points se superposent, plus le modèle est calibré.

Quelques points s'écartent de la tendance générale et s'expliquent par le caractère empirique de nos estimations et de nos prédictions, qui sont donc très dépendantes du découpage en voisinages. Certains voisinages ne comportent pas assez de données pour fournir des estimateurs robustes.

On peut aussi voir que nos prédictions sont moins fiables dans les régions où μ est grand en valeur absolue : pour ces points, notre modèle prédit des σ trop faibles. C'est donc que notre modèle n'est pas parfaitement calibré.

Le même graphique pour le modèle utilisant le gradient classique est donné ici :

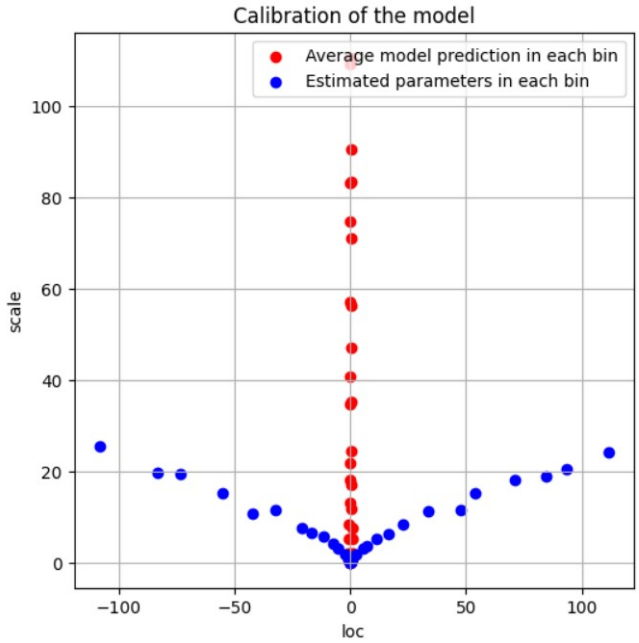


Figure 7 – Représentation de C sur un seul plan $\mathbb{R}^2$, dans le cas du modèle utilisant le gradient classique.

On observe bien que peu importe X, le modèle classique prédit toujours la même valeur pour μ. Les points bleus et rouges suivent des courbes très distinctes, c'est donc que C est très

éloignée de la fonction identité.

Sur nos graphiques, il est compliqué d'observer la correspondance entre points rouges et points bleus. Pour avoir une meilleure représentation de C, nous pouvons visualiser sous forme d'animation le déplacement des points des antécédents vers les images, et ainsi avoir une bonne intuition de l'action de C sur l'espace des (μ, σ). Ces animations ne sont, bien sûr, pas représentables sur papier.

Cette méthode s'applique à d'autres lois et d'autres paramétrages, l'idée est toujours de se rapprocher de la fonction identité d'un espace donné. Un autre exemple pour la loi Gamma (deux paramètres) est donné en annexe.

3.3 Généralisation de la méthode à d'autres lois

En suivant le même schéma, on peut définir un cadre général à notre fonction C.

On considère des paramètres $\omega_1, \omega_2, ..., \omega_n$ définissant une loi $\mathcal{L}(\omega_1, \omega_2, ..., \omega_n)$. On possède des données $\{X_i, Y_i\}_{i \in [\![1,n]\!]}$ avec $X \in \mathcal{X}$ et $Y \in \mathcal{Y}$. On suppose que $Y \sim \mathcal{L}(\omega_1(X), \omega_2(X), ..., \omega_n(X))$.

On cherche à prédire nos paramètres $\omega_1(X), \omega_2(X), ..., \omega_n(X)$ pour chaque X, par un modèle $m(X) = (\hat{\omega_1}(X), \hat{\omega_2}(X), ..., \hat{\omega_n}(X))$.

On suppose que l'on dispose d'estimateurs pour nos $\omega_1, \omega_2, ..., \omega_n$ basés par exemple sur la méthode des moments. On nomme ces estimateurs $W_1, W_2, ..., W_n$ qui sont donc des fonctions de Y (ou plutôt de $Y|X$). On suppose également que nous avons des moyens empiriques d'approximer ces estimateurs.

La fonction de calibration est alors $C : \mathbb{R}^n \to \mathbb{R}^n$ définie par :

$$C(\omega_1, \omega_2, ..., \omega_n) = (W_1(Y|m(X) = (\omega_1, \omega_2, ..., \omega_n)), ..., W_n(Y|m(X) = (\omega_1, \omega_2, ..., \omega_n)))$$

Plus cette fonction est proche de la fonction identité de $\mathbb{R}^n$, mieux notre modèle est calibré : c'est-à-dire qu'il prédit la vraie distribution pour chaque X.

3.4 Limites de la méthode de calibration en régression

Finalement, nous avons bien créé une méthode de représentation visuelle de la calibration d'un modèle en régression. Cette méthode suit la même logique que la méthode en classification en adaptant les problématiques à notre cadre. Pour ces deux méthodes, il reste à quantifier l'écart entre C et la fonction identité pour obtenir un critère rigoureux sur la calibration.

Notre étude nous a permis d'identifier certaines limites, que nous listons ci dessous.

1. Cette méthode ne permet pas de valider ou d'invalider l'hypothèse de départ sur la paramétrisation de la loi. On est donc très dépendants de nos connaissances à priori sur les distributions. Ce n'est pas le cas en classification car il n'y a qu'une seule paramétrisation possible.

2. Nous sommes obligés de séparer en sous-sensemble correspondant à des voisinages (ou "bins") pour avoir suffisamment de données pour faire nos calculs. Nous sommes donc limités par des opérations de moyenne sur les points considérés, qui réduisent la précision que l'on avait sur chaque point. De plus, les bins qui ne contiennent pas assez de données donnent des estimations faussées.

C'est aussi le cas pour la classification, mais plus le problème a de dimensions (c'est à dire de paramètres), plus la séparation en bins est problématique.

3. De même que pour la classification, les conclusions dépendent fortement de notre estimateur empirique. Si l'estimateur basé sur le moment d'ordre 1 pour la moyenne est très robuste, il n'en va pas de même pour toutes les paramétrisations, et les estiamteurs d'ordres supérieurs demandent un nombre de données conséquent.

4
RÉSEAUX DE NEURONES ET MC DROPOUT

4.1 Contexte

Le *MC Dropout* est une méthode pour créer de la variabilité sur les prédictions d'un réseau de neurones et ainsi modéliser l'incertitude sur les données. Il est popularisé par *Gal* et *Ghahramani* [4] en 2020 et est la méthode la plus utilisée aujourd'hui pour représenter l'incertitude d'un réseau de neurones en régression.

MC Dropout se base sur **l'extinction de certains neurones, selon une probabilité fixée**, au moment de la prédiction. Les prédictions sont donc générées par des modèles variés, ce qui s'apparente à des émissions d'un posterior sur les modèles dans le formalisme bayésien. Par cela, il peut s'apparenter à une méthode de *variational inference* (c.f 1.2)

Cependant, le posterior associé à la méthode *MC Dropout* s'éloigne des posterior habituels gaussiens utilisés pour les réseaux de neurones bayésiens. Le lien entre le Dropout et le formalisme bayésien est complexe, et de nombreux articles critiquent la fiabilité du modèle et ses résultats.

Bien que la méthode soit très simple à implémenter, les preuves mathématiques de son efficacité sont encore floues. En effet, le *MC Dropout* génère de l'incertitude et une variabilité sur les données de sortie, mais rien n'indique que cette incertitude est bien **calibrée** sur les données d'entraînement. La question est donc de savoir d'où vient fondamentalement cette incertitude, en faisant le rapprochement avec le formalisme bayésien.

Dans cette partie, nous étudierons le *MC Dropout* en dressant un bilan des recherches en cours, illustrées par des expériences.

4.2 Fonctionnement et utilité

Le terme "Dropout" renvoie à une méthode habituellement utilisée pour l'entraînement d'un modèle, plutôt que pour ses prédictions.

L'idée est la suivante : En éteignant certains neurones aléatoirement à chaque étape de l'entraînement, on les empêche de trop se spécifier sur la donnée. C'est donc une méthode pour éviter le sur-apprentissage par régularisation. En choisissant une probabilité d'extinction appropriée, on cherche à trouver le juste milieu entre la performance du modèle et le sur-apprentissage. On utilise en général une probabilité d'extinction $p = 0.2$.

Le *MC Dropout* utilise l'extinction aléatoire dans l'entraînement mais aussi dans la prédiction. L'idée est que les dropouts en phase d'entraînement créent artificiellement plusieurs modèles entraînés sur un même schéma, chacun avec une configuration de neurones actifs/éteints qui lui est propre. Ces modèles sont donc capables **collectivement** d'exprimer plusieurs tendances sur la donnée et rendent compte de sa variabilié.

Ainsi, en utilisant le dropout au moment de la prédiction on simule des émissions aléatoires selon la distribution d'une multitude de modèles. En supposant que cette distribution sur les modèles, i.e. ce posterior, est bien adaptée à nos données, on se rapproche donc du formalisme bayésien pour la prédiction (mais pas pour l'inférence).

MC Dropout utilise un grand nombre de prédictions différentes et s'en sert pour modéliser l'incertitude. Cette technique s'apparente aux méthodes de Monte Carlo pour échantilloner sur un posterior insoluble analytiquement, et c'est de là que la méthode tire son nom.

Chaque prédiction par un modèle (déterminé par ses neurones actifs et éteints) est vu comme un échantillonage sur le posterior par méthode de Monte Carlo. Ces échantillons modélisent ensemble une **distribution** sur l'espace des prédictions $\mathcal{Y}$.

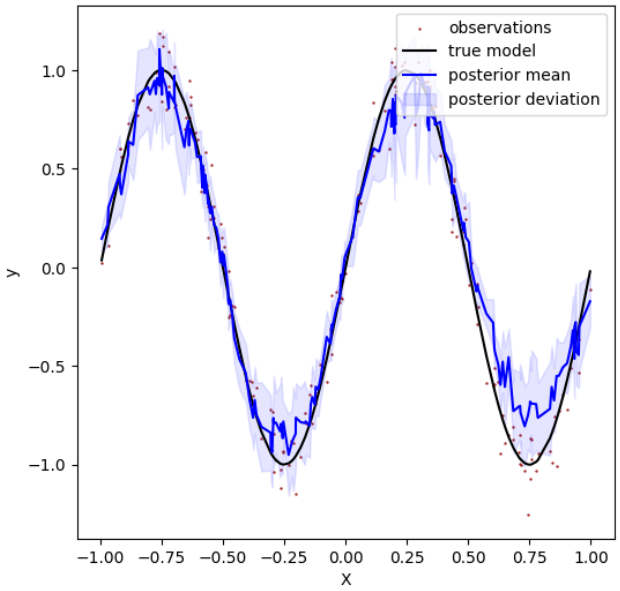


Figure 8 – Moyenne et déviation des prédictions par méthode de *MC Dropout*, en utilisant 10 échantillons.

Toutefois, être capable de prédire une incertitude ne veut pas dire qu'elle est bien calibrée. En effet, il est aisé de remarquer sur la figure précédente que la déviation prédite par le modèle n'est pas interprétable comme la véritable dévation des données. De fait, ces données ont été générées avec un bruit gaussien de paramètre $\sigma = 0.2$ constant. Or, on peut voir que la variance prédite par le modèle varie significativement avec la prédiction. Cette propriété sera étudiée plus en détail par la suite.

4.3 Étude probabiliste par la théorie bayésienne

4.3.1 • Modélisation du posterior par variables de Bernoulli

On peut interpréter les différentes prédictions en sortie du *MC Dropout* comme étant des émissions issues d'un posterior bayésien sur le réseau. En considérant un réseau avec N neurones, et une probabilité d'extinction p, on peut modéliser le posterior sur les neurones. Il est compliqué de parler de prior associé car l'entraînement du modèle est déterministe, et la variabilité sur les coefficients n'arrive qu'à la fin. Le modèle est bayésien uniquement pour la prédiction, pas pour l'entraînement.

Si on identifie chaque neurone à son poids ω après entraînement, le posterior sur ce coefficient est modélisé par ωB où B suit une loi de Bernoulli de paramète $1-p$. C'est donc une combinaison de masses de Dirac, l'une en ω et l'autre en 0, dont les coefficients sont $1-p$ et p respectivement.

Le posterior sur la totalité du réseau de N neurones est donc une combinaison de 2^N masses de Dirac, chacune avec un coefficient $(1-p)^{N_{actifs}} p^{N_{éteints}}$.

Toute la question du choix d'un prior (et d'un poterior associé) est au coeur du formalisme bayésien pour modéliser l'incertitude. C'est pourquoi il est a priori difficile de juger la fiaibilité de ce posterior en comparaison à d'autres plus classiques.

4.3.2 • Interprétation

Le cadre habituel pour un réseau de neurones bayésien est de modéliser l'incertitude par un bruit gaussien et de choisir un prior gaussien conjugué sur les coefficients. Mais ce n'est pas toujours le cas. Le posterior dans ce cadre habituel est une distribution gaussienne multivariée.

Afin d'obtenir le meilleur fit d'un posterior insoluble analytiquement, on peut utiliser différentes techniques de *Variational Inference*. Le but est d'obtenir une distribution qui fit au mieux le posterior théorique de notre modèle, mais qui soit facile à calculer. Le *MC Dropout* peut être interpreté comme une technique de *Variational Inference* car il crée un posterior sur les coefficients du réseau.

Pour visualiser des posteriors et les assimiler, on peut les représenter dans un cas simpliste, inspiré de *Is MC Dropout Bayesian ?, Le Folgoc (2021)* [3], dont est tirée la figure suivante.

Considérons un réseau à deux neurones, sans biais. Il n'y a que deux paramèteres, que l'on nommera θ_1 et θ_2.

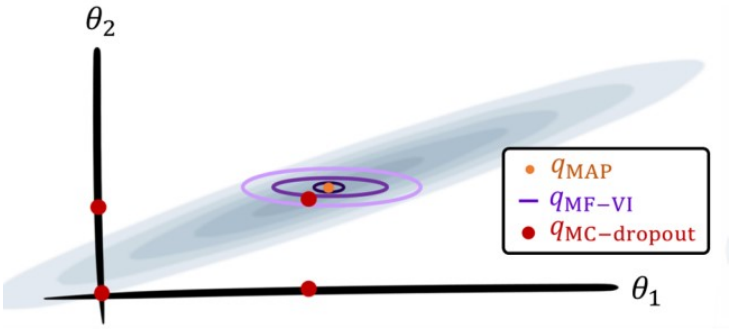


Figure 9 – Illustration des différentes méthodes. MAP est le *Maximum A-Posteriori*, qui est une masse de Dirac placée au maximum de la fonction de densité. MF-VI est le $Mean\ Field\ Variational\ Inference$, ici paramétré par des fonctions gaussiennes.

Dans le fond de la figure précédente est représentée en bleue la distribution gaussienne multivariée. Les posteriors calculés par différentes méthodes de $Variational\ Inference$ sont représentés par dessus.

La méthode *MC Dropout* a visiblement plusieurs défauts. Le Dropout crée de la variabilité calculée à partir du point que l'on atteint à la fin de la phase d'entraînement. Ce point correspond en théorie au *Maximum A-Posteriori*. Toutefois, la variabilité ne rend pas compte de la densité de probabilité aux alentours du point maximum, et place des masses de Dirac à des points qui n'ont pas de réelle signification au vu de l'incertitude du modèle.

De plus, comme pour la méthode MAP, il est impossible de modéliser des distributions multi-modales (cf annexe).

4.4 Expression de la distribution en sortie

Plusieurs articles se sont penchés sur les propriétés mathématiques du *MC Dropout*, et notamment sur l'expression de la variabilité en sortie. *Verdoja* et *Kyrki* prennent l'exemple d'un réseau simpliste avec une seule couche de neurones, sans fonction d'activation, et trouvent une expression de la distribution [7].

On garde les notations habituelles et on notera f la fonction représentant le réseau, et N le nombre de neurones sur la couche. En supposant f entraînée et optimale, les coefficients sont tels que $\forall x \in \mathcal{X}, f(x) = \mathbb{E}[Y|X = x]$

Après application du Dropout, $Verdoja$ et $Kyrki$ démontrent les résultats :

$$E[f(x)] = \frac{N(1-p)}{N(1-p)+p}\mathbb{E}[Y|X=x] \tag{9}$$

$$Var[f(x)] = \frac{Np(1-p)}{(N(1-p)+p)^2}\mathbb{E}[Y|X=x]^2 \tag{10}$$

On remarque plusieurs points :

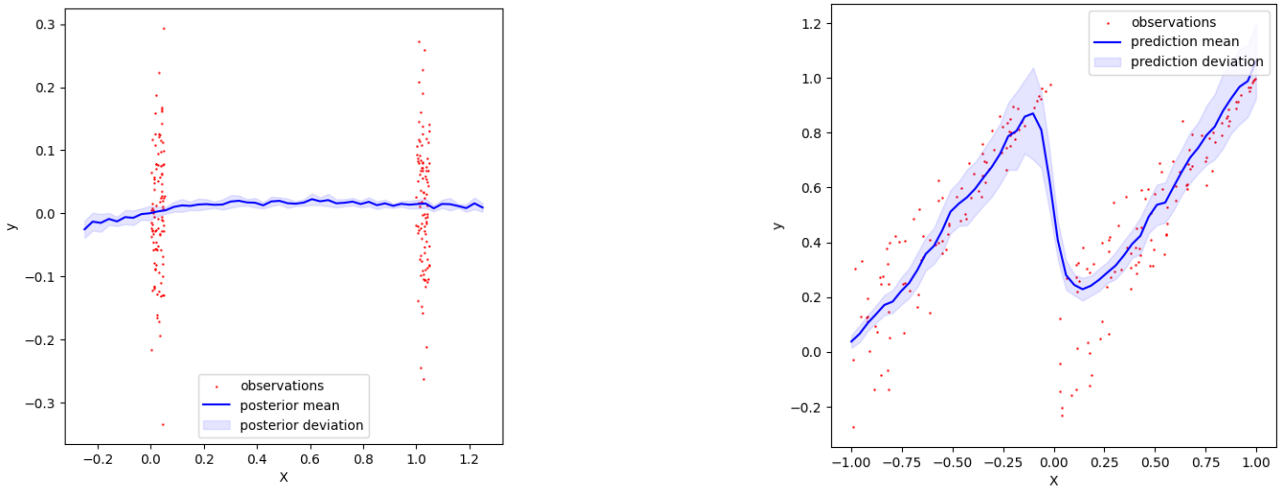
1. $E[f(x)] \neq \mathbb{E}[Y|X=x]$ dès lors que $p \neq 0$. Le *MC Dropout* est un estimateur biaisé. Cette propriété est négligeable avec un grand nombre de neurones.

2. Les prédictions ne dépendent pas de la distribution de la donnée d'entraînement. On retrouve bien que la variabilité ne rend pas compte de nos données. Le *MC Dropout* ne peut pas prédire correctement l'incertitude.

3. La variance prédite augmente avec la norme de $\mathbb{E}[Y|X=x]$, quelque soit la distribution $Y|X$, ce qui n'a aucune justification.

Dans la suite, nous nommerons ces propriétés (1), (2) et (3) respectivement.

Ces propriétés sont démontrées pour cet exemple simpliste uniquement. *Mae*, *Kumagai* et *Kanamori* trouvent des formules plus générales en prenant en compte les fonctions d'activation et la superposition de couches de neurones, en étudiant la propagation de l'incertitude [5]. Toutefois, on peut se contenter d'observer que les propriétés ci-dessus sont valables pour des réseaux plus complexes en les illustrant dans des expériences.



(a) L'incertitude ne dépend pas de la distribution des données.

(b) L'incertitude augmente avec la norme de la prédiction.

Figure 10 – Illustration des propriétés démontrées précédemment. Pour les deux figures, le modèle utilisé a deux couches de 512 neurones avec une probabilité d'extinction $p = 0.2$, activation ReLU et entraîné sur 100 itérations. Le posterior est calculé avec 10 échantillons.

On observe bien sur la figure (a) que l'incertitude prédite par le modèle n'est pas liée aux données. On s'attendrait à avoir une incertitude plus élevée là où il n'y a pas de données

d'entraînement, et une incertitude fittée sur le bruit (variabilité en y) là où le modèle a eu des données d'entraînement. La propriété (2) est toujours vérifiée.

Sur la figure (b), on met en valeur le fait que la déviation augmente avec la norme de la prédiction, alors même que le bruit sur la donnée est décroissant avec y. La propriété (3) est toujours vérifiée.

Nous avons réalisé d'autres expériences pour vérifier que l'architecture du réseau n'avait pas d'impact sur les propriétés. Les résultats sont disponibles en annexe.

4.5 Comparaison avec un modèle témoin : le réseau de neurones bayésien

Pour comparer avec la théorie du cadre bayésien, il faut utiliser un réseau de neurones bayésien. C'est à dire que, contrairement au *MC Dropout*, l'entraînement est probabiliste et le prior de chaque neurone évolue au cours de l'entraînement.

Pour des expériences simples comme celles vues précedemment, le temps de calcul reste abordable pour N le nombre de neurones assez faible. En entraînant un réseau bayésien dans les mêmes conditions (architecture, données, itérations), on obtient :

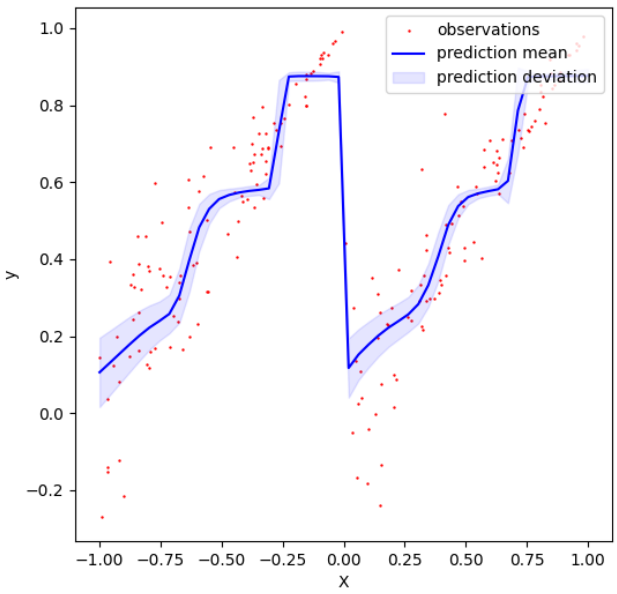


Figure 11 – Prédiction faite sur le deuxième jeu de données vu précedemment (cf Figure 10) par un réseau de neurones bayésien.

Le réseau a besoin de plus d'itérations pour s'entraîner, ce qui explique que les prédictions ne sont pas parfaites, le réseau bayésien est moins précis et efficace avec les mêmes conditions d'entraînement. Toutefois, les résultats suffisent à observer que ce réseau est capable d'apprendre la distribution des données, et de prédire une incertitude adaptée aux tendances observées à l'entraînement.

Cette expérience témoin montre que le formalisme bayésien est capable de prédire des distributions en accord avec les tendances du jeu de données. Cela confirme que les problématiques vues précedemment ne proviennent que du *MC Dropout*, et non du cadre ou de la donnée.

4.6 Conclusion sur le MC Dropout

La méthode *MC Dropout* est très pratique d'utilisation car son entraînement est déterministe, il n'est pas soumis aux mêmes complications que les réseaux de neurones bayésiens (temps de calcul et approximation pour les posteriors insolubles).

Toutefois, l'incertitude fournie par le modèle est infondée. Le modèle affiche des déviations prédites sur la donnée qui ne rendent pas compte de leur distribution. La variabilité du *MC Dropout* relève d'une propriété intrinsèque et n'est aucunement liée à la donnée. L'utilisation du *MC Dropout* pour de la prédiction probabiliste en régression est donc impossible, ou du moins à prouver dans des cas particuliers.

CONCLUSION

Ces recherches nous ont permis de mieux comprendre les subtilités liées à la prédiction d'incertitude, et nous avons développé différents outils et expériences nous permettant d'explorer ce domaine.

Nous avons consacré les premières semaines à l'étude théorique du machine learning bayésien, afin de comprendre les enjeux liés au domaine probabiliste en machine learning.

Puis, nous nous sommes principalement penchés sur l'approche paramétrique pour la régression. Nous avons découvert que la représentation d'une distribution par ses paramètres permet une approche algorithmique simplifiée mais engendre des complexifications mathématiques liées à la topologie de l'espace des distributions et aux propriétés des fonctions de coût utilisées.

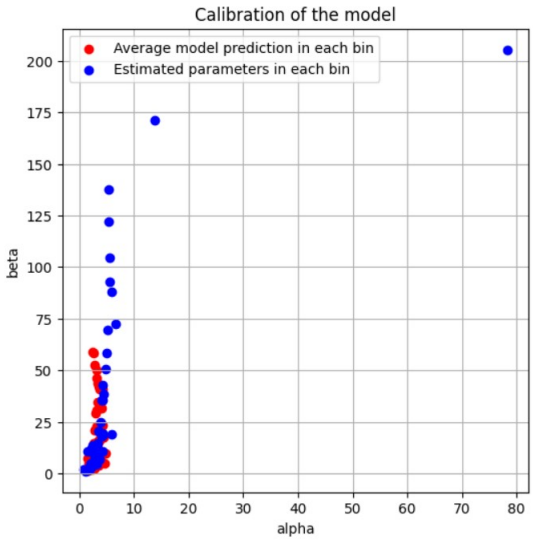
Ces découvertes nous ont poussé à étudier un formalisme de calibration en régression pour juger de la qualité des modèles. A nouveau, nous avons étudié les implications de la paramétrisation des distributions dans ce cadre, en basant notre méthode sur ces propriétés.

Enfin, nous avons étendu notre problématique aux modèles de deep learning. Nous avons centré notre étude sur la méthode *MC Dropout* qui est particulièrement populaire dans ce cadre, et avons montré que la variabilité dans les prédictions n'étaient pas liées à de la calibration probabiliste, mais était plutôt le résultat d'une variabilité artificielle du modèle.

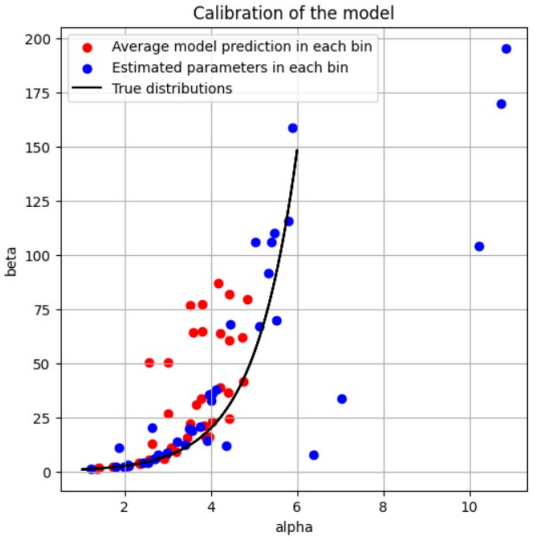
5 ANNEXES

5.1 Calibration 2D pour la loi Gamma

On considère $Y|(X = x) \sim \Gamma(\alpha(x), \beta(x))$, avec $\alpha(x) = |x| + 1$ et $\beta(x) = e^{|x|}$. La méthode généralisée que nous avons défini donne les graphiques :



(a) Représentation de la fonction de calibration de notre modèle.



(b) Zoom et ajout des vraies distributions $(\alpha(x), \beta(x))$

Figure 12 – Résultats pour une loi Gamma, en utilisant l'algorithme basé sur le gradient naturel

La méthode nous permet d'observer que le modèle est moins performant dans le cas de la loi Gamma que dans le cas de la loi normale.

5.2 Effet de la profondeur du réseau sur les propriétés du *MC Dropout*

On note L la profondeur du réseau et N le nombre de neurones sur chaque couche. On réalise des expériences similaires à celles faites précedemment, et on observe le comportement des différentes réseaux.

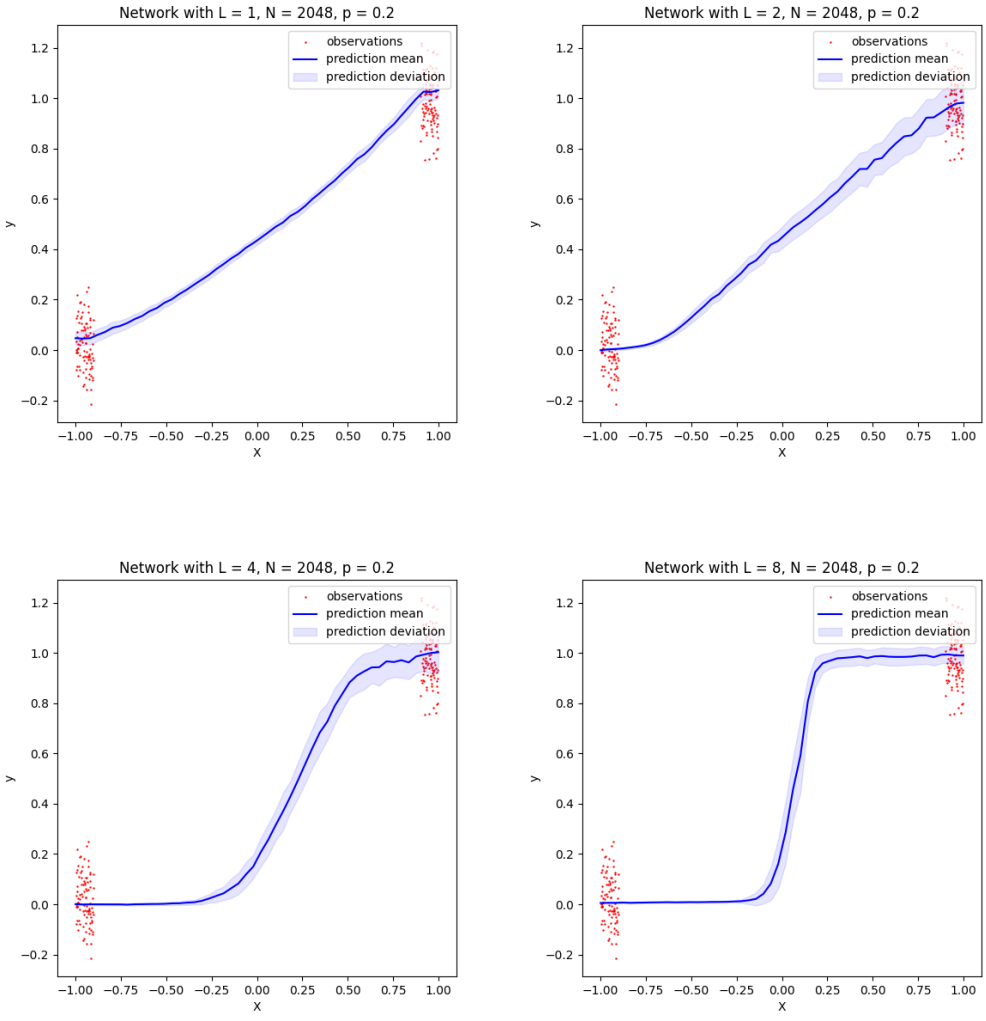


Figure 13 – Expérience avec différentes architectures de réseaux. Les propriétés remarquées précedemment se retrouvent dans tous les cas.

La profondeur du réseau permet d'apprendre des modèles plus complexes (en l'occurence, une discontinuité), mais on observe toujours que l'incertitude augmente avec la prédiction, et qu'elle ne rend pas compte de la variabilité des données.

5.3 Posterior bayésien en multi-modal

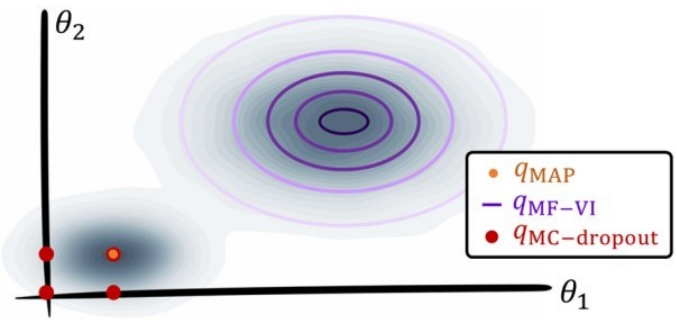


Figure 14 – Illustration des différentes méthodes de *variational inference* pour un mélange gaussien multi-modal

Le *MC Dropout* et le *MAP* se basent toujours sur le point maximal de la distribution, bien que l'essentiel de la "masse" soit centré ailleurs.

RÉFÉRENCES

[1] S. Amari et S.C. Douglas. « Why natural gradient ? » In : *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*. T. 2. 1998, 1213-1216 vol.2. doi : 10.1109/ICASSP.1998.675489.

[2] A. Philip Dawid. « The geometry of proper scoring rules ». In : *Annals of the Institute of Statistical Mathematics* 59 (2007), p. 77-93.

[3] Loic Le Folgoc et al. *Is MC Dropout Bayesian ?* 2021. arXiv : 2110.04286 [cs.LG].

[4] Yarin Gal et Zoubin Ghahramani. « Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning ». In : *Proceedings of The 33rd International Conference on Machine Learning*. Sous la dir. de Maria Florina Balcan et Kilian Q. Weinberger. T. 48. Proceedings of Machine Learning Research. New York, New York, USA : PMLR, 20–22 Jun 2016, p. 1050-1059. url : https://proceedings.mlr.press/v48/gal16.html.

[5] Yuki Mae, Wataru Kumagai et Takafumi Kanamori. « Uncertainty propagation for dropout-based Bayesian neural networks ». In : *Neural Networks* 144 (2021), p. 394-406. issn : 0893-6080. doi : https://doi.org/10.1016/j.neunet.2021.09.005. url : https://www.sciencedirect.com/science/article/pii/S0893608021003555.

[6] Hao Song et al. « Distribution calibration for regression ». In : *Proceedings of the 36th International Conference on Machine Learning*. Sous la dir. de Kamalika Chaudhuri et Ruslan Salakhutdinov. T. 97. Proceedings of Machine Learning Research. PMLR, sept. 2019, p. 5897-5906. url : https://proceedings.mlr.press/v97/song19a.html.

[7] Francesco Verdoja et Ville Kyrki. *Notes on the Behavior of MC Dropout*. 2021. arXiv : 2008.02627 [cs.LG].