

PFE

Khaled MEDJKOUH
Davidson Lova RAZAFINDRAKOTO

4 avril 2023

Table des matières

1	Introduction	3
1.1	Réseau de neurones	3
1.2	Source d'incertitudes dans un réseaux de neurones	3
1.2.1	Acquisition des données	3
1.2.2	Structure du model	3
1.3	Prédiction	3
1.3.1	Incetitude aléatoire	4
1.3.2	Incetitude épistémique	4
2	Réseau de neurones bayésiens (BNN)	5
2.1	Méthodes variationelles	5
2.2	Méthode de Laplace	5
2.3	Méthodes par échantillonnage ou Monte Carlo	5
3	Algorithme BNN-ABC-SS	7
3.1	ABC (Approximate Bayesian Computation)	7
3.2	SS (Subset Simulation)	7
3.3	Pseudo - Code	8
4	Réalisation	10
4.1	Cosinus perturbé	10
4.2	Sinus perturbé	10
5	Conclusion et perspective	10

1 Introduction

Incertitudes utile dans plusieurs domaines et la quantification duquel est nécessaire pour des évaluations de risques, détection d'anomalie et prise décision.

1.1 Réseau de neurones

- + Historique sur les réseaux de neurones
- + Explication des poids et biais et fonction d'activation
- + Quelque résultats et application
- + Ce qui nous amène à un problème

1.2 Source d'incertitudes dans un réseaux de neurones

On modélise le réseaux de neurones comme la fonction non linéaire

$$f : (x, \theta) \in \mathcal{X} \times \Theta \mapsto f(x, \theta) \in \mathcal{Y}$$

où

- $\mathcal{X} \subset \mathbb{R}^{n_e}$, l'espace des variables d'entrées
- $\mathcal{Y} \subset \mathbb{R}^{n_s}$, l'espace des variables de sorties
- $\Theta \subset \mathbb{R}^{n_p}$, l'espace des paramètres

On se donne maintenant une base de données d'entraînement $D = \{(x_i, y_i)\}_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$.

1.2.1 Acquisition des données

Si on prend comme données d'entrées x une mesure d'une quantité réel \tilde{x} .

Il y a une variabilité sur la mesure en fonction des circonstances et conditions $\omega \in \Omega$ dans lequel la mesure a été effectué.

La sortie y peut aussi subir des erreurs de labelisation (pour le cas d'une tâche de classification) ou aussi de mesure si c'est une quantité mesuré.

En somme on a une incertitude sur l'entrée $x|\omega \sim p_{x|\omega}$ et $y|\omega \sim p_{y|\omega}$.

S'ajoute à ça, l'incertitude sur x qui peut se propager sur le y .

1.2.2 Structure du model

Les paramètres θ et donc l'espace Θ est variable en fonction du choix de model s .

On a $\theta|D, s \sim p_{\theta|D,s}$

1.3 Prédiction

La distribution de la prédiction y^* sachant une entrée x^* est données par

$$p(y^*|x^*, D) = \int_{\Theta} \underbrace{p(y^*|x^*, \theta)}_{\text{Données}} \underbrace{p(\theta|D)}_{\text{Modèle}} d\theta$$

1.3.1 Incertitude aléatoire

Insert definition here

Elle affecte la partie $p(y^*|x^*, \theta)$ de la tâche de prédiction, elle est due à la variabilité (précision et erreurs) lors des mesures. Elle est donnée et inhérent au problème.

1.3.2 Incertitude épistémique

Insert definition here

Affecte la partie $p(\theta|D)$ de la tâche de prédiction, elle due :

- à la complexité du modèle,
- aux erreurs durant la phase d'entraînement,
- à la manque d'information à cause de données manquantes ou la capacité de représentation des données d'entraînement.

2 Réseau de neurones bayésiens (BNN)

Les réseaux de neurones bayésiens sont des réseaux de neurones dont les poids sont, non pas des quantités déterministes (comme dans le cas d'un NN normale) mais des distributions.

A l'initialisation, les poids suivent une loi a priori $p(\theta)$, et l'entraînement consiste à évaluer l'a posteriori de cette loi conditionnée aux données d'entraînement $p(\theta|D)$.

On ne dispose pas dans le cas générale, d'une formule analytique de cette distribution a posteriori.

Voici trois familles de méthodes pour approcher cette distribution :

- Méthodes variationnelle
- Méthodes par échantillonnage ou Monte Carlo (qu'on va voir dans la suite)
- Méthodes de Laplace

2.1 Méthodes variationnelles

Ici on approche $p(\theta|D)$ par une famille de distribution paramétrique $\{q^\gamma(\theta)\}_\gamma$ (souvent des gaussiennes). Le but est de choisir γ qui rapproche $q^\gamma(\theta)$ le plus de $p(\theta|D)$. La distance choisit ici est la divergence de Kullback-Leibler :

$$KL(q||p) = \mathbb{E}_q \left[\log \frac{q^\gamma(\theta)}{p(\theta|D)} \right]$$

Comme on ne connaît pas $p(\theta|D)$, on utilise l'ELBO (evidence lower bound) qui est égal à la divergence à une constante paramètres

$$L = \mathbb{E}_q \left[\log \frac{p(y|D, \theta)}{q^\gamma(\theta)} \right]$$

(on a en effet $KL(q||p) = -L + \log p(y|D)$)

2.2 Méthode de Laplace

$\hat{\theta}$ l'estimateur de maximum d'a priori

$$\log p(\theta|D) \approx \log p(\hat{\theta}|D) + \frac{1}{2}(\theta - \hat{\theta})^T (H + \tau I) (\theta - \hat{\theta})$$

$$p(\theta|D) \sim \mathcal{N}(\hat{\theta}, (H + \tau I)^{-1})$$

2.3 Méthodes par échantillonnage ou Monte Carlo

La formule de Bayes nous donne

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

- $p(D|\theta)$ la vraisemblance des données D sachant le paramètre θ ,
- $p(\theta)$ la distribution a priori de θ ,
- $p(D)$ la distribution des données d'entraînement.

$$p(\theta|D) \propto$$

3 Algorithmme BNN-ABC-SS

3.1 ABC (Approximate Bayesian Computation)

La méthode ABC consiste à évaluer $p(\theta|D)$ sans évaluer la vraisemblance qui peut s'avérer coûteux.

Posons $\hat{y} = f(x, \theta)$ la sortie d'une évaluation de x par réseaux de neurones f avec paramètre θ .

La formule de Bayes nous donne

$$p(\theta, \hat{y}|D) \propto p(D|\hat{y}, \theta)p(\hat{y}|\theta)p(\theta)$$

Pour simuler selon la distribution du second membre, on applique l'algorithme de rejet.

- On tire $\theta \sim p(\theta)$
- On évalue $\hat{y} = f(x, \theta) \sim p(\hat{y}|\theta)$
- On accepte le θ si et seulement si $\hat{y} = y$

Comme \hat{y} est une quantité réel (a priori à distribution continue), obtenir exactement $\hat{y} = y$ est une condition trop forte pour être atteinte (en un temps raisonnable).

On introduit alors une tolérance ϵ , on remplace $\hat{y} = y$ par $|\hat{y} - y| < \epsilon$.

On remarque que plus ϵ est petit, plus on se rapproche de la condition $\hat{y} = y$, et donc le mieux notre approximation sera.

On note $p_\epsilon(\theta, \hat{y}|D)$ la distribution issue du tirage précédent, et qui approche $p(\theta, \hat{y}|D)$, on a

$$p_\epsilon(\theta, \hat{y}|D) \propto \mathbb{1}_{\mathcal{N}_\epsilon(D)}(\hat{y})p(\hat{y}|\theta)p(\theta)$$

où

$$\mathcal{N}_\epsilon(D) = \{\hat{y} \in \mathcal{Y}, \rho(\eta(\hat{y}), \eta(y)) \leq \epsilon\}$$

où η est une statistique qui caractérise une distribution (par exemple les moments ou les quantiles) et ρ est une mesure de dissimilarité.

En intégrant des deux cotés par \hat{y} on obtient notre approximation de $p(\theta|D)$

$$\begin{aligned} p_\epsilon(\theta|D) &= \int_{\mathcal{Y}} p_\epsilon(\theta, \hat{y}|D) d\hat{y} \propto \int_{\mathcal{Y}} \mathbb{1}_{\mathcal{N}_\epsilon(D)}(\hat{y})p(\hat{y}|\theta)p(\theta) d\hat{y} \\ &= p(\theta) \int_{\mathcal{Y}} \mathbb{1}_{\mathcal{N}_\epsilon(D)}(\hat{y})p(\hat{y}|\theta) d\hat{y} = \mathbb{P}(\hat{y} \in \mathcal{N}_\epsilon(D)|\theta)p(\theta) \end{aligned}$$

Cependant lors de l'algorithme de rejet, tirer les θ de manière générique donne trop rarement des \hat{y} qui tombe dans $\mathcal{N}_\epsilon(D)$ ce qui fait que l'approximation est prend beaucoup de temps à converger. On utilise alors SS (Subset Simulation) pour faire des tirages plus fins.

3.2 SS (Subset Simulation)

Soient $\epsilon_1 > \epsilon_2 > \dots > \epsilon_m = \epsilon$ des seuils.

Il est clair que $\mathcal{N}_{\epsilon_m}(D) \subset \mathcal{N}_{\epsilon_{m-1}}(D) \subset \dots \subset \mathcal{N}_{\epsilon_2}(D) \subset \mathcal{N}_{\epsilon_1}(D)$.

De plus et en conséquence, $\bigcap_{j=1}^m \mathcal{N}_{\epsilon_j} = \mathcal{N}_{\epsilon_m} = \mathcal{N}_\epsilon$, ce qui nous donne

$$\begin{aligned}
\mathbb{P}(\hat{y} \in \mathcal{N}_\epsilon(D)|\theta) &= \mathbb{P}\left(\hat{y} \in \bigcap_{j=1}^m \mathcal{N}_{\epsilon_j}(D)|\theta\right) \\
&= \mathbb{P}(\hat{y} \in \mathcal{N}_{\epsilon_1}(D)|\theta) \prod_{j=2}^m \mathbb{P}(\hat{y} \in \mathcal{N}_{\epsilon_j}(D)|\hat{y} \in \mathcal{N}_{\epsilon_{j-1}}(D), \theta)
\end{aligned}$$

L'idée de la SS, est de faire des tirages itératifs de plus en plus fins.

Initialement on tire de manière générique avec un ϵ_1 grand.

A chaque itération, on tire à partir (conditionnées) des tirages précédents avec un ϵ_i plus fin.

Après n itérations, on aura tiré avec ϵ_n beaucoup plus petit que ϵ_0 .

Cette méthode exploite la décomposition d'une probabilité d'un ordre très petit en un produit de probabilité d'un ordre assez grand pour être calculable en un temps raisonnable.

3.3 Pseudo - Code

Algorithme 1 : Pseudo code BNN - ABC - SS

Entrées : $N \in \mathbb{N}^*$: le nombre de tirage à chaque itération

$l_{\max} \in \mathbb{N}^*$: le nombre d'itération maximale

P_0 la proportion de nos tirages qu'on garde pour générer à la prochaine itération

ϵ_0 : la tolérance initiale

ϵ : la tolérance finale

x : la variable d'entrée

y : la variable de sortie

Sorties : $\left[\theta_1^{(n)}, \dots, \theta_N^{(n)} \right]$

$NP_0 \leftarrow N * p_0$;

$iP_0 \leftarrow p_0^{-1}$;

$\left[\theta_1^{(0)}, \dots, \theta_N^{(0)} \right]$, N tirage de $\theta \sim p(\theta)$;

$\left[\hat{y}_1^{(0)}, \dots, \hat{y}_1^{(0)} \right] \leftarrow \left[f(x, \theta_1^{(0)}), \dots, f(x, \theta_N^{(0)}) \right]$;

$\left[\gamma_1^{(0)}, \dots, \gamma_1^{(0)} \right] \leftarrow \left[\rho(\eta(y), \eta(\hat{y}_1^{(0)})), \dots, \rho(\eta(y), \eta(\hat{y}_N^{(0)})) \right]$;

$\epsilon_j \leftarrow \epsilon_0$;

pour $j \in \{1, \dots, l_{\max}\}$ **faire**

 On ordonne $\left[\gamma_1^{(j-1)}, \dots, \gamma_1^{(j-1)} \right]$ dans l'ordre croissant ;

 On réordonne $\left[\theta_1^{(j-1)}, \dots, \theta_N^{(j-1)} \right]$ dans cette ordre ;

$\epsilon_j \leftarrow \gamma_{NP_0}^{(j-1)}$ (ou $\frac{1}{2}(\gamma_{NP_0}^{(j-1)} + \gamma_{NP_0+1}^{(j-1)})$) ;

pour $k \in \{1, \dots, NP_0\}$ **faire**

 On choisit une graine $\theta_k^{(j-1),1} = \theta_k^{(j-1)}$ tq $\hat{y}_k^{(j-1)} \in \mathcal{N}_{\epsilon_j}(D)$

 On génère iP_0 états d'une chaîne de Markov de θ tq $\hat{y} \in \mathcal{N}_{\epsilon_j}(D)$:

$\left[\theta_k^{(j-1),1}, \dots, \theta_k^{(j-1),iP_0} \right]$ et avec ça $\left[\gamma_k^{(j-1),1}, \dots, \gamma_k^{(j-1),iP_0} \right]$

fin

$\left[\theta_1^{(j)}, \dots, \theta_N^{(j)} \right] \leftarrow \left[\theta_k^{(j-1),l}, k \in \{1, \dots, NP_0\}, l \in \{1, \dots, iP_0\} \right]$

$\left[\gamma_1^{(j)}, \dots, \gamma_N^{(j)} \right] \leftarrow \left[\gamma_k^{(j-1),l}, k \in \{1, \dots, NP_0\}, l \in \{1, \dots, iP_0\} \right]$

si $\epsilon_j \leq \epsilon$ **alors**

 Fin de l'algorithme ;

fin

fin

4 Réalisation

+ On se donne une base de données à étudier avec une comparaison avec des méthodes déjà existantes¹.

4.1 Cosinus perturbé

4.2 Sinus perturbé

5 Conclusion et perspective

1. MATSUMOTO et al. 2013.

Références

- [Mat+13] Haru MATSUMOTO et al. « Tracking beaked whales with a passive acoustic profiler float ». In : *The Journal of the Acoustical Society of America* 133.2 (2013), p. 731. ISSN : 00014966. DOI : 10.1121/1.4773260.