Contents

Chapter 1

Contexte du stage

Dans ce chapitre, il s'agit pour nous de décrire les contours du stage. Pour ce faire, nous commencerons par présenter le projet LISA et terminerons par une présentation de la méthode de développement.

Contents

1.1 Présentation de LISA

Le projet Laser Interferemoter Space Antenna (LISA) est un observatoire spatial dont le lancement est prévu en 2034 qui permettra la détection directe d'ondes gravitationnelles indétectables par les interféromètres au sol tels que LIRGO ,VIRGO.
Cet observatoire doit être capable de détecter un grand nombre de signaux(onde gravitationnelle) provenant d'environs 60 millions de système binaires galactiques. Concrètement LISA doit permettre de caractériser ces systèmes binaires galactiques composé de trous noirs, laines blanches etc.
D'autres part, caractériser ces systèmes revient tout simplement à estimer leurs paramètres (intrinsèques et extrinsèques) à partir des données issues des ondes gravitationnelles. Pour ce faire, des méthodes classiques d'estimations paramétriques telles MCMC ont été abordé

Ainsi, le but de mon stage est d'utiliser les méthodes d'apprentissage automatique en particulier les réseaux de neurones pour une estimation rapide de ces paramètres.

Dans ce document, nous allons implémenter deux types de réseaux de neurones à savoir le Perceptron Multi-Couches (MLP) et les Réseaux de Neurones Convolutifs (CNN).

1.2 Les données

Pour faire de l'apprentissage automatique, nous avons besoin d'une base d'exemple pour l'entraînement et l'évaluation de notre modèle. Ainsi, les données qui m'ont été fournir sont celles du simulateur LISAcode.

1.2.1 Base d'apprentissage

Un élément essentiel de l'apprentissage automatique est de disposer de données d'apprentissages nombreuses et de qualité.
Notre base d'exemple est constitué de $N = 40000$ formes d'ondes observées sur 256 fréquences dans 3 canaux (X,Y,Z) . Concrètement, une donnée est constituée d'une forme d'onde, de 256 fréquences et d'un canal.

1.2.2 Environnement de travail

L'analyse du problème nous emmène à porter notre choix sur le langage de programmation Python en utilisant principalement la librairie **JAX** pour l'implémentation de notre modèle.

Chapter 2

Le MLP

Dans cette partie, nous allons présenter la description du réseau, des données , les paramètres à estimer et les différents graphiques permettant d'évaluer notre modèle.

Contents

2.1 Les données d'entrées

Notre réseau de neurones prends en entrée un vecteur de dimension 3

2.1.1 Préprocessing sur les données d'entrées

Après avoir faire une première analyse de nos données , nous obtenons les résultats suivant:

- Il y a 100 bandes de fréquences différentes chacune ayant une fréquence centrale f_0.
- Nous n'avons pas les mêmes fréquences *Min* et *Max* dans chaque bande.
- Dans chaque bande nous avons 400 formes d'ondes chacune ayant 256 points d'observation.
- Le pas d'échantillonnage n'est pas le même dans toutes les bandes de fréquences.

Dans un premier temps, nous allons considérer que la première bande de fréquence pour la suite de ce travail.

2.1.1.1 Normalisation

Après avoir extraire les données sur lesquelles nous devons travailler, j'ai par la suite entamer la phase de normalisation de celles-ci afin de réduire la complexité de notre modèle. Ainsi, la transformation effectuer sur chaque forme d'onde est la suivante:

$$\overline{S_i} = \frac{S_i}{\|S_i\|_2}, \quad \forall i \in \overline{0, 400}$$

où S_i représentent nos données brutes.

2.1.1.2 Génération de variables aléatoires uniforme

Pour rendre les observations plus réaliste, nous avons juger important d'augmenter les amplitudes des différentes formes d'ondes en vue d'atteindre un SNR de 20 dB. A cet effet, pour chaque forme d'onde, nous avons générer 5 formes d'ondes à amplitude différente toutes ayant un SNR maximal de 20 dB.

Dans la pratique, cela revient à multiplier la forme d'onde normalisée $\overline{S_i}$ par une variable aléatoire uniforme X_j.

$$\tilde{S_{i,j}} = \mathrm{X}_j \times \overline{S_i}, \quad \forall i \in [0, 1, ..., 400], \quad j \in \overline{0, 5} \quad avec \quad \mathrm{X}_j \to \cup(1, 10)$$

2.1.2 Mise en forme des paramètres à estimer

Nos paramètres à estimer eux aussi présentent quelques anomalies du point de vu de leur différence d'écart remarquables. Ils sont au nombre de 8 au total à savoir:

- La latitude Ecliptic β
- la longitude Ecliptic λ
- L'amplitude h
- La fréquence f_0
- La dérivée

Dans un premier temps, nous allons estimé 3 d'entre eux.

2.1.2.1 Paramètres estimés

D'après la section précédente, les paramètres suivants seront estimés dans un premier temps.

- La latitude Ecliptic β
- La longitude Ecliptic λ
- L'amplitude h

2.1.2.2 Transformation

La transformation faite sur les paramètres consiste à les mettre tous dans un intervalle [-1;1] afin de palier au problème d'ordre de grandeur. En effet , l'amplitude présente un ordre de grandeur plus important par rapport aux autres paramètres ce qui pourrait avoir un impact sur l'erreur d'estimation au cours de l'apprentissage.

NB: Pour la phase de test, nous écrirons une fonction permettant de prendre en compte les paramètres physiques et non les paramètres transformés.

2.2 Apprentissage

2.2.1 Split data

Pour ne pas utiliser toutes les données disponibles pour l'entraînement, nous allons la diviser en deux parties : données d'apprentissages et de tests.

- L'ensemble d'entraînement pour entraîner notre modèle.
- L'ensemble de test pour mesurer la performance de notre modèle sur les données apprises.

2.2.2 Architecture réseau

Les architectures ont leurs forces et faiblesses et peuvent être combinées pour optimiser les résultats. Le choix de l'architecture s'avère ainsi crucial et il est déterminé principalement par l'objectif.
Les architectures de réseaux neuronaux peuvent être divisées en 4 grandes familles :

- Réseaux de neurones Feed fowarded
- Réseaux de neurones récurrent (RNN)
- Réseaux de neurones à résonance
- Réseaux de neurones auto-organisés

La famille de MLP dont nous allons implémenter est le réseau de neurones de type Feed Forwarded (Propagation avant) qui signifie tout simplement que les données traversent le réseau d'entrée à la sortie sans retour en arrière de l'information.

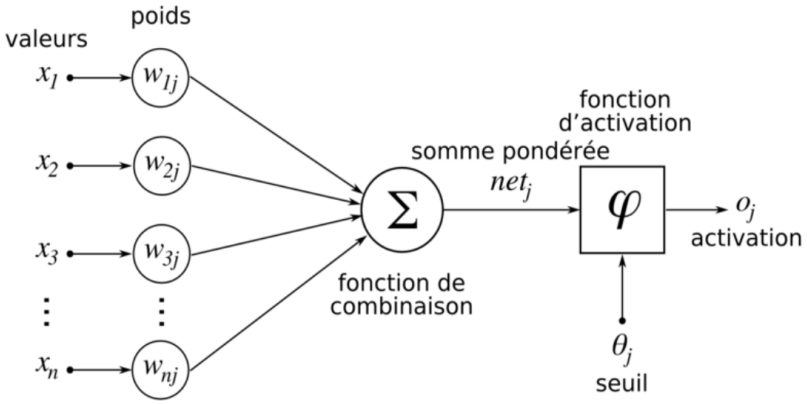

Figure 2.1: Architecture réseau
Où valeurs désigne les données d'entrées, poids et fonction d'activation sont des paramètres liés au réseau.

2.2.2.1 Initialisation des paramètres

Nous allons à présent initialiser les paramètres liés au réseau à savoir les poids (w) et le biais(b). En effet, une meilleur initialisation nous évitera de faire de l'overfitting sur nos données; Pour cela nous avons initialisé ces paramètres principalement les poids du réseau à l'aide de **Glorot** (Glorot Normal). Glorot Normal , permet de maintenir la variance constante le long de toutes les couches du réseau.

Note: w $\rightarrow N(0, s^2), avec$ s=$\sqrt{\frac{2}{n_i+n_o}}$

2.2.2.2 Construction du réseau de neurones

Le réseau mise en place dans un premier est composé d'une couche d'entrée de dimension 3 , de trois couches cachées chacune ayant 10 neurones et d'une couche de sortie avec trois neurones faisant référence aux trois paramètres à estimer.

2.2.3 Feedforward function

On note x_{mjt} les données d'entrée à trois dimensions. Le neurone artificiel effectue une somme pondérée à l'aide de la relation suivante:

$$y_{mik} = \sum_{jt} w_{ikjt} \times x_{mjt} + b_{mik}$$

.
Ensuite, nous utilisons comme fonction d'activation la fonction **Relu** sur toutes les couches de notre réseau sauf la dernière qui prends elle la fonction **Identité** et dont le vecteur de sortie s'écrit sous la forme :
$out_{ms} = \sum_{ik} w_{sik} \times y_{mik} + b_{ms}$.

2.2.4 Loss function

La fonction de perte est la fonction qui nous permettra d'évaluer les prédictions réalisées par notre réseau et les valeurs réelles des observations utilisées lors de l'apprentissage.
Le réseau mis en place effectue une erreur de **0.76720226** sur les données d'entraînement.

2.2.5 Backpropagation

Pour la phase de propagation arrière, nous avons utilisé l'hyper-paramètre Adam Optimizer avec un learning rate de 10^{-3}

2.2.6 Training loop

2.2.7 Enregistrement des hyperparamètres dans un fichier

2.3 Analyse des résultats sur le TrainSet

2.3.1 Visualisation de la courbe de déscente de gradient

2.3.2 Prédiction et observation des résultats

2.4 Evaluation du modèle sur le TestSet

2.4.1 Remise en forme des paramètres à estimés

2.4.2 Visualisation des prédictions sur les trois paramètres

2.4.3 Histogramme des erreurs de prédiction sur le TestSet

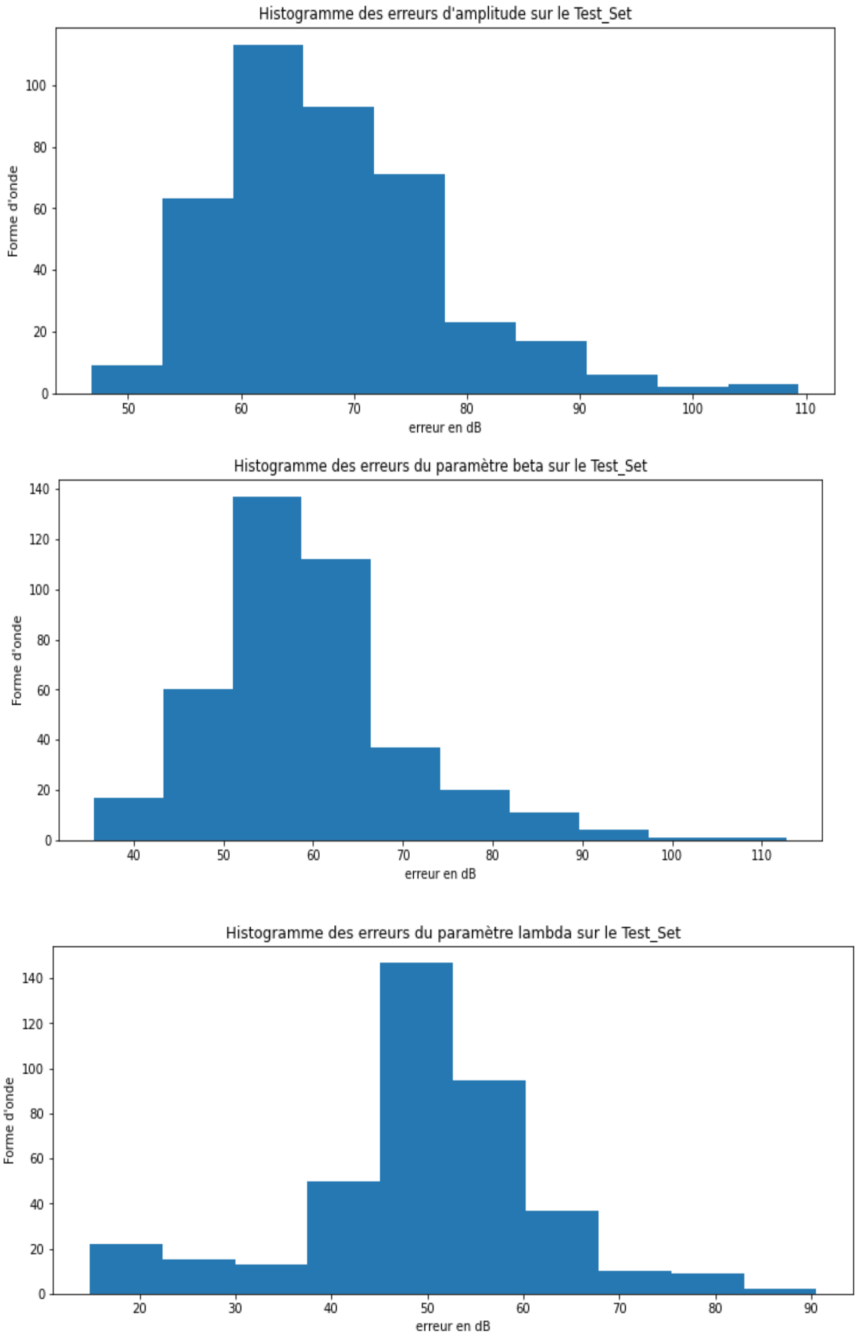

Figure 2.2: Histogramme des erreurs sur les trois paramètres

Chapter 3

Le CNN