UTILISATION DE PERCEPTRON MULTICOUCHE POUR L'INVERSION DES DONNEES RADIOMETRIQUE SUR LES OCEANS

I. INTRODUCTION

La lumière émise par le soleil est diffusée et absorbée par l'atmosphère avant d'être captée par les radiomètres embarqués à bord des satellites. La diffusion et l'absorption sont dues aux molécules présentes dans l'atmosphère mais aussi aux aérosols, qui sont de fines particules en suspensions dans l'air. De plus le satellite reçoit de la lumière provenant de la surface du sol ou de l'océan.

Nous nous plaçons au dessus des océans, et on peut alors écrire la mesure radiométrique pour une longueur d'onde λ donnée :

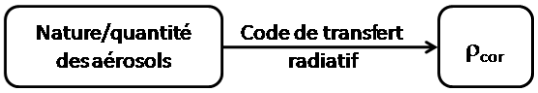
Où ρ désigne la reflectance qui est la quantité optique mesurée par le capteur. Les indices *mes*, *aer*, *mol*, et *oce* correspondent respectivement à la grandeur mesurée par le satellite, à la contribution des aérosols, des molécules et de l'océan.

Pour certaines longueurs d'onde λ du proche infrarouge (λ>600nm), on peut supposer que $\rho_{oce}(\lambda)$=0 (ce qui se traduit par le fait que, du point de vue du satellite, l'océan est noir et ne renvoie pas de lumière)

La distribution des molécules dans l'atmosphère est bien connue et permet de modéliser avec précision ρ_{mol}. On peut donc calculer une quantité corrigée ρ_{cor} qui ne dépend que de la présence d'aérosol :

Le but de la télédétection satellitaire est d'utiliser la mesure ρ_{cor} pour en déduire la quantité et la nature des aérosols présents dans l'atmosphère. L'étude des aérosols est un problème clé en environnement et est relié aux processus de désertifications, à la pollution due à l'homme ou aux feux de forêt. En effet, les aérosols sont produits sur les continents et sont transportés ensuite par les vents sur de larges zones de l'océan.

Afin d'exploiter les données satellitaires, on utilise un modèle de transfert radiatif qui à partir d'une description des aérosols permet de prédire la mesure satellitaire



Dans un premier temps, on décrit les données utilisées, puis on présente les objectifs principaux de ce TP. Enfin, on détaille les différentes étapes à réaliser. Un formulaire des différentes fonctions matlab à utiliser dans les scripts est donné en annexe.

LES DONNEES

LES AEROSOLS

Les aérosols sont décrits par 2 caractéristiques : leur type, et leur épaisseur optique.

L'épaisseur optique qu'on notera tau peut-être vue comme la concentration de ces aérosols dans l'atmosphère.

Le type est une notion complexe qui fait intervenir un certain nombre de propriétés optiques (réfraction, absorption) ou physique (taille). Généralement, on définit plusieurs classes d'aérosols type : les aérosols maritimes, les aérosols désertiques, et les aérosols continentaux. Ces classifications seront vues dans un TP ultérieur.

LA GEOMETRIE

La reflectance rho_cor issue des aérosols dans l'atmosphère dépend de la position du satellite, et de celle du soleil. Généralement, on décrit ces positions par 3 angles, l'angle zénithal solaire thetas, l'angle zénithal de vue thetav et la différence des angles azimutaux dphi.

LA « LOOK-UP TABLE »

Un modèle numérique a été utilisé pour générer les reflectance. Ce modèle a pour entrées l'épaisseur optique, le type d'aérosol et les 3 angles de géométrie. En sortie, il donne la ρ_{aer} (=ρ_{cor} dans notre cas).

Le fichier LUT_NIR.dat contient donc 7 colonnes :

Colonne 1 (Type) : type d'aérosol utilisé

Colonne 2 (Tau) : épaisseur optique

Colonne 3 (Thetas) : angle zénithal solaire

Colonne 4 (Thetav) : angle zénithal de vue

Colonne 5 (Dphi) : différence des angles azimutaux

Colonne 6 (Lambda) : longueur d'onde (=670nm, 765nm ou 865nm)

Colonne 7 (Rho_aer) : Reflectance aérosol.

TRAVAIL A REALISER

APPRENTISSAGE DU PERCEPTRON MULTICOUCHE

1°) Construire la base des entrées et des sorties du perceptron multicouche. Le réseau prend en entrée : Thetas, Thetav, Dphi, Rho_aer(670), Rho_aer(765), Rho_aer(865) où 670, 765 et 865 sont les différentes longueurs d'onde Lambda. En sortie, le réseau donne Tau.
fonction à utiliser : mix_base

2°) Séparer la base aléatoirement en base d'apprentissage (80%)/base de test (10%)/base de validation (10%).
Questions:

- Pourquoi séparer la base en 3 sous-bases ?
- A quoi servent chacune des nouvelles bases ainsi crées apprentissage/test/Validation ?
- Qu'est-ce que le sur-apprentissage ?
- Qu'est-ce qu'une architecture ?
- Comment séparer la base ?

3°) Remarques sur l'apprentissage
La performance d'un réseau de neurone dépend de la complexité de la relation que l'on cherche à modéliser (ici l'épaisseur optique TAU en fonction des réflectances et de la géométrie) et de l'apprentissage lui-même.

- Afin de réduire la complexité de la relation à apprendre, il est souvent nécessaire de normaliser les paramètres d'entrée et de sortie.
- L'apprentissage va dépendre de l'initialisation et de l'architecture choisie pour le réseau. En l'absence d'autre information a priori sur le problème traité, on testera les performances de réseaux appris avec différentes architectures et différentes initialisations.

4°) Normalisation des paramètres.

Dans cet exemple nous allons centrer et réduire les paramètres d'entrée et de sortie selon la formule suivante :

$$x_n = 2/3\frac{x - \bar{x}}{\sigma(x)}$$

Le facteur 2/3 devant permet de ramener la majorité des paramètres dans l'intervalle [-1 1] qui est l'intervalle pour laquelle la fonction d'activation du réseau est la plus variable.

- Faire la normalisation des 3 bases créées et vérifier la répartition des données à l'aide d'histogrammes.
  Fonction à utiliser : cenred, hist

Exemple d'histogramme montrant la répartition des données et l'effet de la normalisation

5°) Apprentissage

Préparer un script permettant de faire l'apprentissage pour plusieurs initialisations et plusieurs architectures. Voici l'algorithme de ce script. On essaiera par exemple 5 architecture archi=[20 40 60 80 100] et pour chacune de ces architectures, 2 initialisations.
Fonctions à utiliser : MLPinit, MLPfit, MLPval

Script d'apprentissage

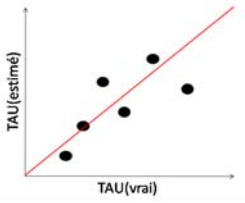
```
Archi=[20 40 60 80 100]

Init = [1 2]

Pour n allant de 1 à la taille de Archi

      Pour i allant de 1 à la taille de Init

            -  Initialise le réseau R(n,i)avec Archi(n)
            -  Apprend le réseau R(n,i)
            -  Calcule les performances sur la base de validation P(n,i)

      Fin pour i

Fin pour n
```

- Lancer le script d'apprentissage
- La fonction MLPfit.m prend en entrée la base de validation, pourquoi ?

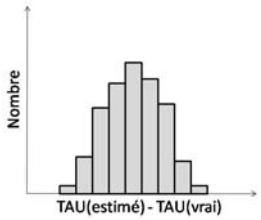
A la fin de cette phase d'apprentissage on doit pouvoir compléter le tableau suivant. Dans chacune des cases du tableau, on portera la RMS (root mean square error). Important : toutes les performances d'erreurs doivent être données sur des données NON normalisées. Pourquoi ?

	Initalisation 1	Initialisation 2
20 neurones		
40 neurones		
60 neurones		
80 neurones		
100 neurones		

6°) Détermination de l'architecture optimale.
A l'aide du tableau ci-dessus, donner l'architecture optimale pour le réseau et calculer ses performances sur la base de test. On donnera la RMS, l'erreur relative, le coefficient de corrélation entre les valeurs de TAU retrouvées par le réseau et les valeurs de TAU « vraies » dans la base. Enfin on affichera un diagramme de dispersion entre les valeurs « vraies » et les valeurs retrouvées et un histogramme des erreurs.
Fonction à utiliser : MLPval, plot, hist



Exemple de diagramme de dispersion



Exemple d'histogramme des erreurs

TEST SUR UNE IMAGE

On va tester l'inversion sur de vraies mesures radiométriques de SeaWiFS sur la mer Méditerranée. On prend par exemple l'image du 8 Septembre 1999. Les données sont stockées dans le fichier base_image_251.dat (le nombre 251 correspond au 251ème jour de l'année 1999). Le fichier est organisé comme suit :

Colonne 1 : Thetas

Colonne 2 : Thetav

Colonne 3 : Dphi

Colonne 4 : Rho_cor(670)

Colonne 5 : Rho_cor(765)

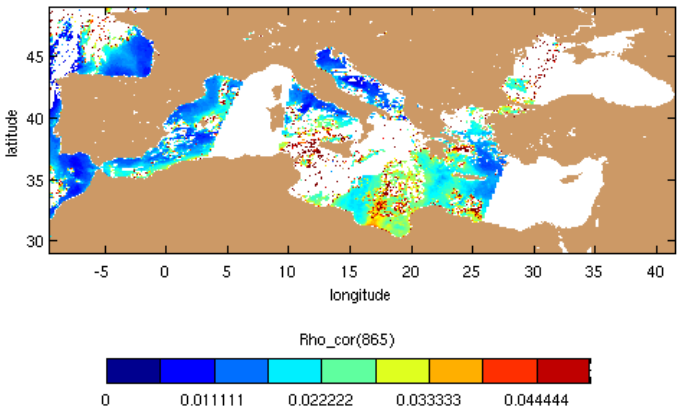
Colonne 6 : Rho_cor(865)

En plus de ces données il y a 3 fichiers permettant l'affichage :

- Valid_Pixels_251.dat : indices des pixels pour lesquels il y a une mesure radiométrique valide
- Val_Pix_Terre_251.dat : indices des pixels correspondants à la Terre
- Val_Pix_Nuages_251.dat : indices des pixels n'ayant pas été mesuré (souvent à cause des nuages)

Ces 3 derniers fichiers sont utilisés pour la fonction *plot_image.m*.

1°) Afficher les cartes des différents paramètres (Thetas, Thetav, Dphi, Rho_cor, …)
fonction à utiliser : plot_image



Carte de Rho_cor(865) sur la mer Méditerranée pour le 8 Septembre 1999

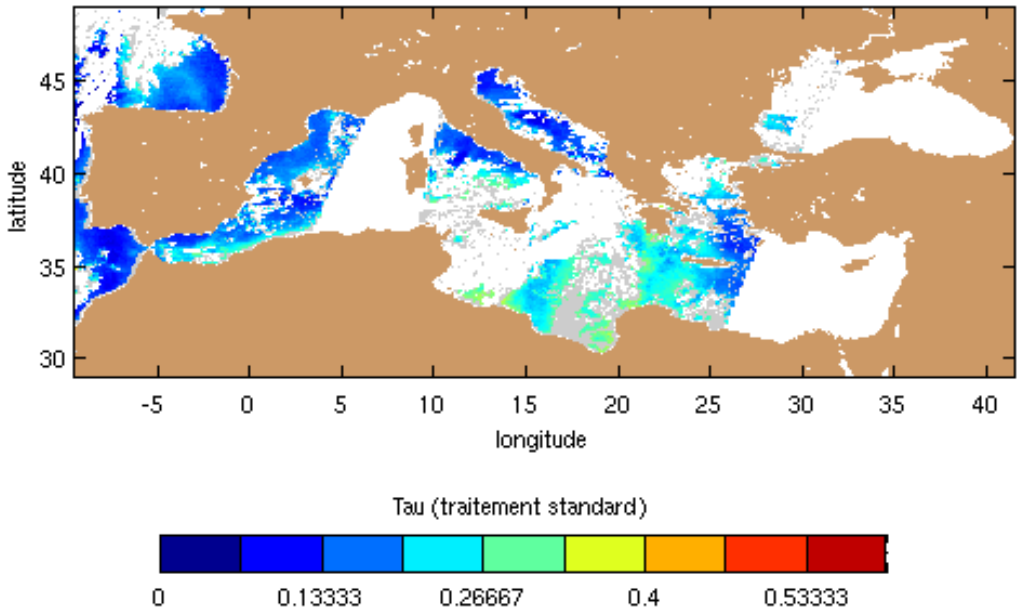
2°) Certains pixels ne peuvent pas être traiter par le réseau de neurone car les valeurs sont en dehors de l'intervalle de valeurs déterminée dans la base d'apprentissage. Il est donc nécessaire dans un premier temps de sélectionner les pixels qui sont susceptibles d'être traiter par le réseau.
fonction à utiliser : find

3°) Normaliser les données satellites afin de pouvoir utiliser le perceptron multicouche

4°) Utiliser le perceptron multicouche optimum obtenu à l'étape précédente afin d'estimer la valeur de Tau pour cette image. Afficher le résultat obtenu.
fonctions à utiliser : MLPval, plot_image

5°) Comparaison avec le traitement standard

Pour traiter les images radiométriques SeaWiFS, il existe des algorithmes standards de traitement. Nous allons donc comparer les restitutions de Tau obtenues par le réseau de neurone avec les restitutions standards. Le fichier de Tau traité par l'algorithme standard est Tau_standard.dat et contient une colonne de Tau au même format qu'une colonne du fichier base_image_251.dat



Traitement standard pour le 8 Septembre 1999. Les pixels en gris sont non traités et les pixels en blanc sont non mesurés.

Questions :

- Peut-on comparer tous les pixels ?
- Que donnent les résultats (diagramme de dispersion/histogramme) sur les pixels comparables
- Combien de pixel sont traitées uniquement par les réseaux de neurones, uniquement par le traitement standard, par les deux méthodes de traitement ?

LA VALIDATION CROISEE

Un facteur important pour estimer la qualité du modèle est le choix de la base de test et de la base d'apprentissage. Si la base de test n'est pas représentative de la base d'apprentissage (c'est-à-dire que sa distribution statistique diffère), l'erreur calculée sur cette base de test ne sera **pas** représentative de l'erreur du modèle. Une façon de traiter ce problème est de faire une validation croisée décrite de la façon suivante :

Algorithme de validation croisée

```
Base <- base totale
Ndiv <- Nombre de division de la base
Base(1 ..Ndiv) <- base totale divisée en 10 sous-base
Delta <- vide
Pour n allant de 1 à Ndiv
      Base_test <- Base(n)
      Base_app  <- Base(1..(n-1) (n+1)..Ndiv)
      -  Faire l'apprentissage avec Base_app et Base_test
      Erreur <- sortie_calculee –
      sortie_desiree
      Delta <- union (Delta,Erreur)
Fin n
```

La validation croisée permet à toutes les données de la base d'avoir été des données test à un moment.

1°) Faire le script de la validation croisée pour 60 neurones dans la couche cachée en prenant Ndiv=10 et en faisant de apprentissage courts (100 epochs).

2°) Comparer les résultats avec les performances données en première partie.