Réseaux de neurones multicouches 1

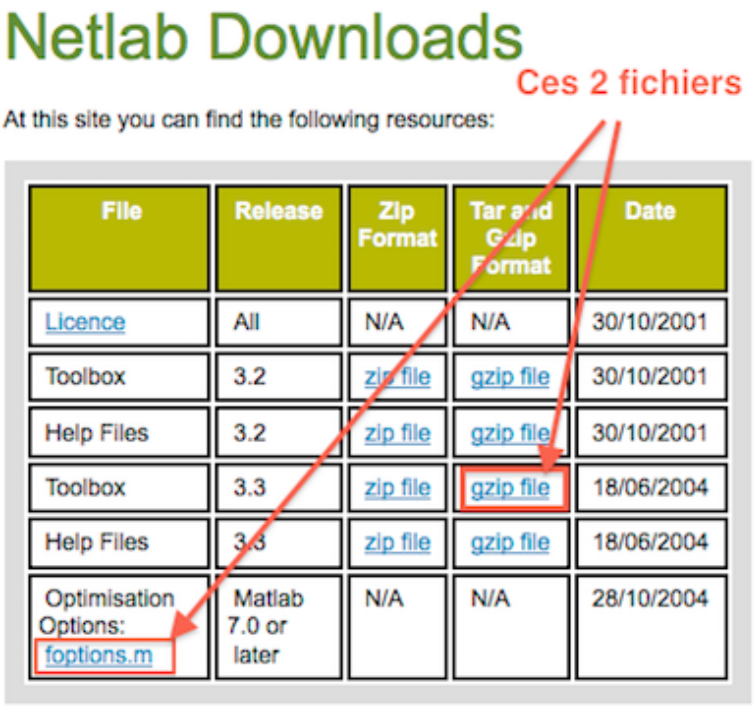
1 Chargement de la toolbox Netlab

Adresse de la Toolbox Netlab Cliquez sur le lien suivant :

`http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/downloads/`

- Charger le fichier **netlab3_3.tar**
  C'est-à dire, cliquez sur **"gzip file"** ⟶ ligne : Toolbox 3.3 (Date : 18/06/2004)
  → comme indiqué sur la figure ci-dessous.
- Charger aussi le fichier *foptions.m*
- Décompresser le fichier **netlab3_3.tar** dans un un répertoire **Netlab**
  Sous linux :
  tar xvf netlab3_3.tar
  (Si on veut supprimer le fichier netlab3_3.tar ⟶ **rm** netlab3_3.tar)
- Déplacer le fichier **foptions.m**
  dans le répertoire **Netlab**

☞ Vous disposez alors de la **toolbox Netlab** (dans le répertoire **Netlab**)



2 Utilisation de la toolbox Netlab avec octave

Si vous utilisez **octave** alors Il faut ajouter dans le répertoire **Netlab** une fonction **fcnchk.m** contenant les instructions suivantes :

```
function f=fcnchk(x, n)
f = x;
end
```

Pour les auditeurs qui utilisent octave, je pense que le lien suivant est intéressant (il y a peut être d'autres liens) :

`http://www.ices.utexas.edu/sysdocs/Octave-Matlab/index.html`

Il donne les différences entre matlab et octave. C'est là que j'ai trouvé l'information concernant la fonction **fcnchk.m** (j'ai testé et ça marche !). Voici ce que dit le document :

- If something (like Netlab) need a function called fcnchk you can just put the following into a file called fcnchk.m and put it somewhere Octave can find it:

```
function f=fcnchk(x, n)
  f = x;
end
```

3 Perceptron sans couche cachée

3.1 Les données

1. Ecrire (ou faire copier-coller) les instructions suivantes dans un fichier **donnee1.m**.

```
a=2;
b=3;
n=200;
X=4*(rand(n,1)-.5);
X=sort(X);
Y=a*X+b;
Yb=Y+.2*randn(size(Y,1),1);
figure
plot(X,Y,'b-',X,Yb,'c.')
legend(': Y',': Yb');
save data1 X Y Yb
```

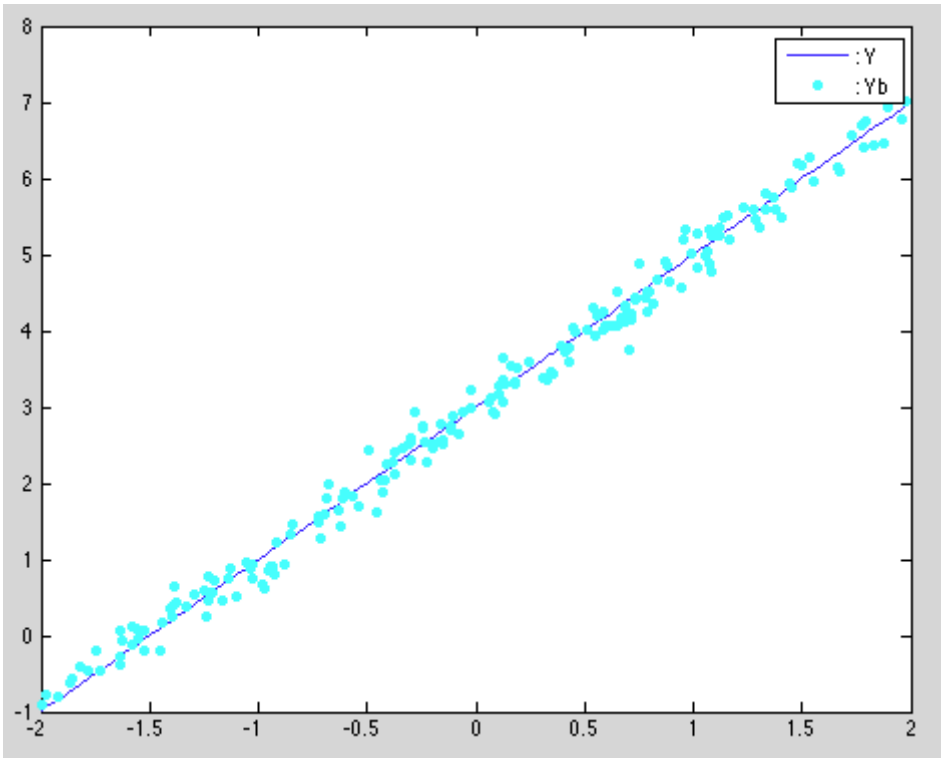
Fichier donnee1.m avec des commentaires

```
a=2; b=3; n=200;
% n points triés par ordre croissant
X=4*(rand(n,1)-.5);
X=sort(X);
Y=a*X+b; % équation d'une droite
Yb=Y+.2*randn(size(Y,1),1); % Yb : Y bruitée
% Vidualisation (voir figure ci-dessous)
figure
plot(X,Y,'b-',X,Yb,'c.')
legend(': Y',': Yb');
save data1 X Y Yb % sauvegarde
```

2. Placez-vous dans la fenêtre de commande Matlab et tapez

   >> **donnee1** ⟶ pour générer les données **data1.mat**

On obtient la figure suivante :



3.2 Apprentissage

1. Taper (ou faire copier-coller) les instructions suivantes dans un fichier **perceptron1.m**

```
global options
addpath Netlab
load data1
Xapp=X(1:2:end);
Yapp=Yb(1:2:end);
Xtest=X(2:2:end);
Ytest=Yb(2:2:end);
Net = glm(size(Xapp,2),size(Yapp,2),'linear');
options     = foptions;
options(1)  = 1;
options(14) = 100;
options(18) = 0.001;
[Net options errlog] = netopt(Net, options, Xapp, Yapp, 'graddesc');
figure
plot(errlog)
Yc=glmfwd(Net,X);
figure
plot(X,Y,'b-',X,Yc,'r-')
legend(': Y',': Yc');
Etest=glmerr(Net,Xtest,Ytest)
```

Fichier perceptron1.m avec des commentaires

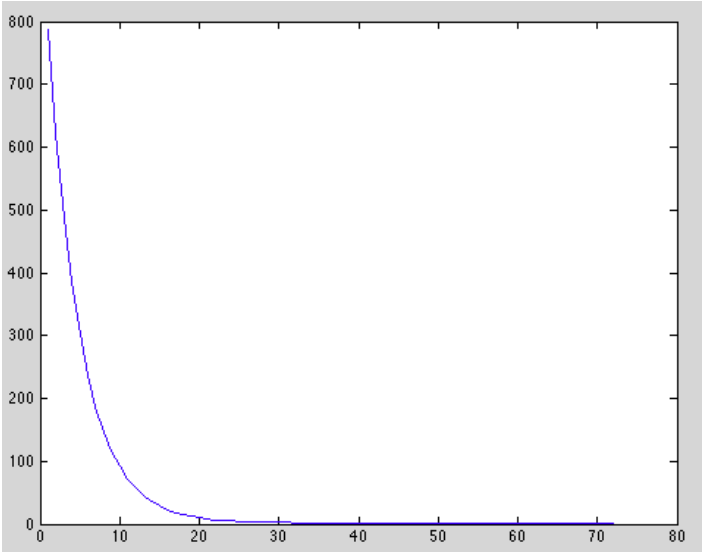
```
global options
addpath Netlab % À modifier selon l'emplacement de Netlab
               % addpath : pour ajouter le chemin d'accès aux fonctions Netlab
               % faire help addpath pour plus d'informations
load data1 % Chargement des données ( à modifier selon l'emplacement de data1)
Xapp=X(1:2:end); Yapp=Yb(1:2:end); % Base d'apprentissage
Xtest=X(2:2:end); Ytest=Yb(2:2:end); % Base de test
% Choix d'une architecture et initialisation des poids
Net = glm(size(Xapp,2),size(Yapp,2), 'linear'); % Taper Net pour voir le résultat
options = foptions; % vecteur contenant les paramètres d'apprentissage
options(1) = 1; % on active l'affichage des erreurs
options(14) = 100; % nombre de cycles d'apprentissage
options(18) = 0.001; % le pas d'apprentissage
% Apprentissage
[Net options errlog] = netopt(Net, options, Xapp, Yapp, 'graddesc');
% Vidualisation : voir les figures ci-dessus
figure
plot(errlog) % Variation de l'erreur durant l'apprentissage
Yc=glmfwd(Net,X); %Yc : sortie calculée par le perceptron Net (après apprentissage) pour l'entrée X
figure
% Maintenant on va montrer :
% - en utilisant la couleur bleu : la droite d'équation Y= 2 X + 3 (celle qu'on veut apprendre)
% - en utilisant la couleur rouge : Yc (celle que le réseau a appris)
plot(X,Y,'b-',X,Yc,'r-')
legend(': Y',': Yc');
% Maintenant on va calculer l'erreur sur la base de test
% entre   Ytest = 2*Xtest + 3   et   Ytestcal=glmfwd(Net,Xtest)
% Ytestcal : sortie calculée par le perceptron Net (après apprentissage) pour l'entée Xtest
Etest=glmerr(Net,Xtest,Ytest)
```

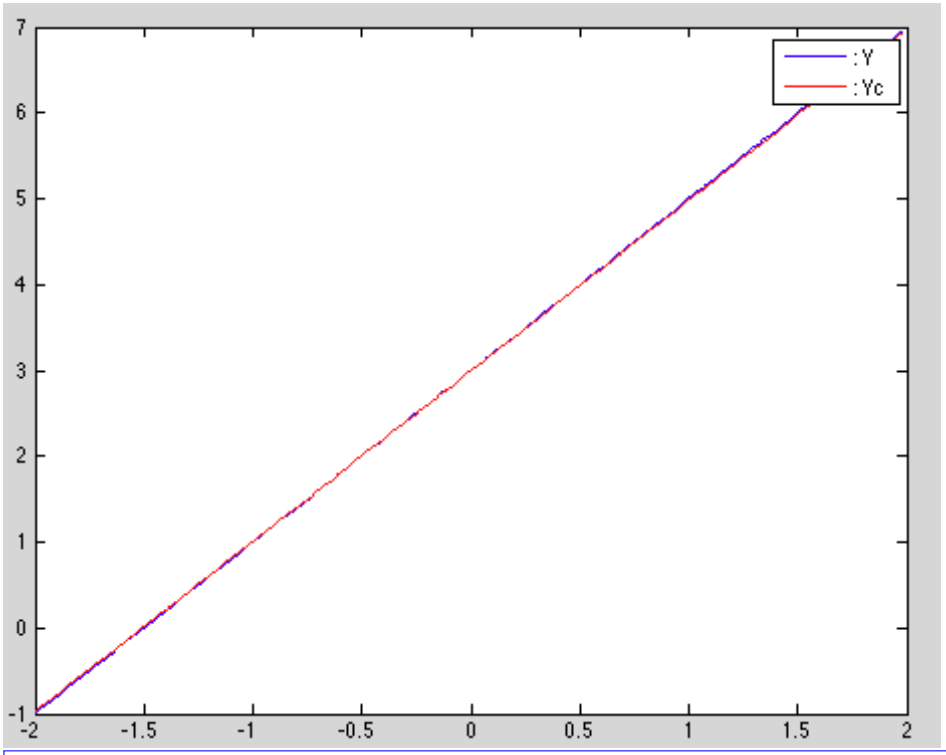
2. Dans la fenêtre de commande Matlab, tapez

   >> **perceptron1** ⟶ pour faire un premier apprentissage

   On obtient les 2 figures suivantes :

   Variation de l'erreur durant l'apprentissage





- **En bleu** la droite d'équation : **Y=a∗X+b** ⟶ avec **a=2** et **b=3** (voir fichier **donnee1.m**)
- **En rouge** la droite d'équation : **Yc = Net.w1∗X + Net.b1**
  ☞ équation de le droite apprise par le réseau

- Taper

  >> **Net.w1**

  ⟶ Comparer **Net.w1** avec **a** (**a=2** dans le fichier **donnee1.m**)

- Taper

  >> **Net.b1**

  ⟶ Comparer **Net.b1** avec **b** (**b=3** dans le fichier **donnee1.m**)

- Pour voir les autres champs de **la structure Net**, taper (dans la fenêtre de commande Matlab)

  >> **Net**

4 Problème réel de classification : les Iris de Fisher

Les Iris de Fisher correspondent à **150 fleurs** décrites par **4 variables** quantitatives :
longueur du sépale, **largeur du sépale**, **longueur du pétal** et **largeur du pétal**.
Les 150 fleurs sont réparties en **3 différentes espèces** : **iris setosa**, **iris versicolor** et **iris virginica**.
Chaque classe est composée de **50 fleurs**. La classe setosa est linéairement séparable des deux autres, alors que versicolor et virginica ne le sont pas.

Les données sont disponibles à l'adresse suivante : `http://deptinfo.cnam.fr/new/spip.php?article851`
Il faut télécharger les deux fichiers **iris_don.mat.zip** et **iris_cls.mat.zip**
Vous pouvez charger directement ces données en cliquant sur les liens suivants :
→ pour charger **iris_don.mat.zip** : `http://deptinfo.cnam.fr/new/spip.php?pdoc5291`
→ pour charger **iris_cls.mat.zip** : `http://deptinfo.cnam.fr/new/spip.php?pdoc5292`
Ensuite il faut décompresser ces deux fichiers (sous linux unzip iris_don.mat.zip puis unzip iris_cls.mat.zip).

Le fichier **iris_don.mat contient les vecteurs d'entrée en dimension 4 décrivant les 150 iris**.
Le fichier **iris_cls.mat contient les classes des 150 iris**.

- Pour lire ces fichiers sous Matlab faire :

load -ascii iris_don.mat
load -ascii iris_cls.mat

(→ Compte-rendu)

☞ **À rendre au plus tard pour le lundi 4 Février 2019**

1. Utiliser 120 iris pour apprendre une carte topologique.
2. Labeliser chaque cellule (neurone) de la carte en utilisant le vote majoritaire.
3. Classer les 30 fleurs restantes en utilisant la carte.
4. Evaluer l'erreur de classification au moyen de la matrice de confusion.

Il faut envoyer par email à **meziane.yacoub@cnam.fr** ☞ un petit rapport (**format pdf**) contenant :

- La carte (visualisation de la carte) obtenue à la fin de la phase 2 (phase de convergence)
- La qualité de cette carte (l'erreur de quantification et l'erreur topologique)
- Le pourcentage de bonne classification sur l'ensemble des 30 fleurs, non utilisées durant l'apprentissage.
- Le pourcentage de bonne classification sur l'ensemble des 150 fleurs (toutes les fleurs).
- La matrice de confusion sur l'ensemble des 150 fleurs (toutes les fleurs) et les indices des instances (fleurs) mal classées par la carte.
- Vos programmes matlab (octave).