TP Apprentissage
Autour du perceptron multicouches

1 Objectif

Le but de ce TP est de construire un perceptron multicouches à une couche cachée, dans le but de résoudre des problèmes de classification et de régression. Dans un premier temps, on s'intéressera à la construction, au test et à la compréhension d'un perceptron simple.

L'archive à votre disposition sur l'ent (ressources pédagogiques>cours en ligne>statistical Learning>PMC) contient un ensemble de sources matlab, pour certains à compléter, et des jeux de données pour tester vos algorithmes. La quasi intégralité du code vous est fournie, quelques lignes restent à écrire (ça ne serait pas un TP sinon...) qui vous permettront de tester et valider le comportement de vos réseaux. Bien entendu, pour valider vos résultats, vous devrez faire tourner plusieurs fois votre code en apprentissage et en test (les réseaux obtenus à chaque itération étant tous différents, les poids étant initialisés de manière aléatoire). La moyenne des résultats de vos réseaux pourra être affichée sous forme de courbes, à l'aide de fonctions fournies.

A l'issue de ce TP, il vous est demandé un compte rendu présentant les réponses aux questions posées, agrémentées de capture d'écran des résultats (classification, régression, courbes des statistiques d'apprentissage) et de commentaires pertinents.

2 Le perceptron

Le fichier **neuronMain.m** (listing 1) contient le squelette du programme principal permettant de construire et d'utiliser un perceptron Vous utiliserez les jeux de données linéairement séparables symétrique (linsym.txt) et dissymétrique (lindissym.txt) dans vos expériences.

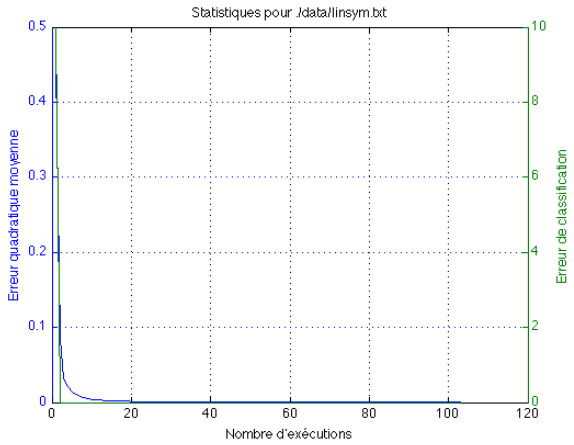
```
% Jeu de donnees au format suivant
% Type du jeu de données (linéairement séparable symétrique ou
% dissymétrque, non linéairement séparable)
% Dimension d'un exemple (dim_exemples)
% Nombre d'exemples (nb_class)
% exemples + labels

function NeuronMain(nomFichier)


% Chargement du jeu de données
learning_set = readData (nomFichier) ;

% Définition du perceptron
neuron = [] ;
neuron.dim_exemples  = learning_set.dim_exemples ;

% Lancement de l'apprentissage
n_epochs      = 200 ; % nombre de présentation de l'ensemble d'apprentissage
mse           = 0.0001 ; % test d'arrêt sur l'erreur quadratique
learning_rate = 0.1 ; % taux d'apprentissage initial
disp (' Apprentissage Perceptron linéaire :') ;
[neuron stat] = ApprentissagePerceptron (neuron , learning_set , learning_rate , mse , n_epochs) ;

% Affichage des statistiques
figure(1) ;
[ax,h1,h2]=plotyy(1:neuron.n_epochs,[stat(:).mse] ,1:neuron.n_epochs,[stat(:).err_exemple],'plot') ;
set(get(ax(1),'Ylabel'),'String','Erreur quadratique moyenne') ;
set(get(ax(2),'Ylabel'),'String','Erreur de classification') ;
xlabel ('Nombre de présentation de l''ensemble d''apprentissage') ;
title (sprintf('Statistiques pour %s' , nomFichier)) ;
grid on

stat(end).mat_conf

% Affichage de l'ensemble d'apprentissage
figure(2) ;
axis ([0 1 0 1]) ;
axes = [1 2] ;
plot_exemple (axes , learning_set.matExemple , learning_set.label) ;

% Génération de points dans le carré unité
nb_points = 70 ;
pointsTest = rand(nb_points,learning_set.dim_exemples);

% test du perceptron sur l'ensemble généré
for i = 1:nb_points
    o = neuronPropagation(neuron , pointsTest(i,:)) ;
    plot_output (axes , pointsTest(i,:) , (o >= 0.0)*1 , o) ;
end

% Affichage de l'hyperplan séparateur
plot_hyperplane (neuron.weights , [0 1] , [0 1]) ;
```

Listing 1 – Programme principal perceptron linéaire

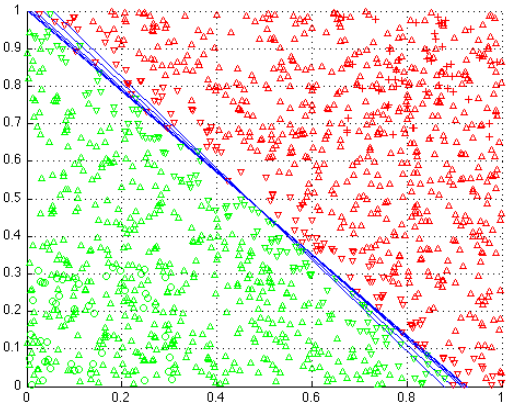
2.1 Apprentissage

Le fichier **ApprentissagePerceptron.m** réalise l'apprentissage du perceptron linéaire. Dans ce fichier, vous avez à compléter deux parties.

1. la partie de code relative à l'apprentissage. Vous testerez les trois options suivantes :
   - tanh
   - sigmoïde : $\sigma(x) = \frac{1}{1+e^{-\lambda x}}$. Pour cette fonction vous évaluerez l'influence du paramètre λ sur les statistiques d'apprentissage et les valeurs de sortie du perceptron.
   - seuil (fonction de Heaviside). Dans ce cas, qu'observe t'on pendant la phase d'apprentissage ?



Erreur quadratique



Hyperplans

FIGURE 1 – Exemples de résultat d'exécution des codes du perceptron linéaire

2. la partie de code relative à la mise à jour des poids.
3. Quel est l'impact du paramètre learning_rate sur l'apprentissage ?
4. quelle différence notable constatez vous dans la phase d'apprentissage entre les jeux de données symétrique et dissymétrique ? En phase de test, comment se comporte alors le perceptron ?

2.2 Propagation d'un exemple

Le fichier **neuronPropagation.m** décrit comment le perceptron calcule la sortie suite à la présentation d'un exemple donné.

5. Ecrire cette fonction en utilisant la tangente hyperbolique comme fonction d'activation du neurone. Bien penser au biais....

Pour illustration, la figure 1 présente l'évolution de l'erreur quadratique moyenne en fonction du nombre de présentation de l'ensemble d'apprentissage, et 5 hyperplans calculés sur 5 exécutions du perceptron.

2.3 Cas non linéairement séparable

Le jeu de données nonlinxor.txt contient un problème de séparation non linéaire. Tester votre perceptron sur ce jeu de données... Comment améliorer vos résultats ?

3 Le perceptron multicouches

Comme nous l'avons vu en cours, un moyen de séparer des données non linéairement séparables est de rajouter une couche cachée au perceptron, et de construire ainsi un perceptron multicouches. Nous nous intéressons ici au cas d'un PMC à une seule couche cachée, dans le cas de la classification du jeu de données XOR.

En construisant un réseau de neurones avec une couche cachée, l'idée est de laisser les neurones de la couche cachée effectuer la séparation entre sous ensembles d'exemples, tandis que la couche de sortie en réalise une combinaison afin de fournir en sortie la classe de l'exemple présenté en entrée.

3.1 Construction du PMC

Dans ce cas, il est raisonnable d'envisager qu'un réseau à une seule couche cachée, avec deux neurones sur cette couche est suffisant. Dans la suite, nous allons construire et apprendre ce réseau. La fonction d'activation retenue pour les neurones est la tangente hyperbolique.

Le code **PMCMain.m** (listing 2) vous donne le squelette du programme principal.

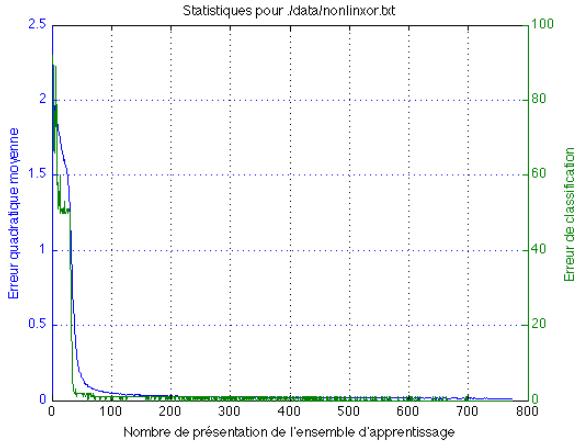
```
filename = './data/nonlinxor.txt' ;
learning_set = readData (filename) ;

% Définition de l'architecture du Perceptron Multicouches
net = [] ;
net.dim_exemples  = learning_set.dim_exemples ;
net.dim_cachees = 2 ;
net.dim_output = learning_set.nb_class ;

% Lancement de l'apprentissage
n_epochs      = 1000 ;
mse           = 0.01 ;
learning_rate = 0.01 ;
disp ('Apprentissage PMC :') ;
[net stat] = PMCLearning (net , learning_set , learning_rate , mse , n_epochs) ;

% Affichage des statistiques
figure(1) ;
[ax,h1,h2] = plotyy (1:net.n_epochs,[stat(:).mse] ,1:net.n_epochs,[stat(:).err_exemple],'plot') ;
set(get(ax(1),'Ylabel'),'String','Erreur quadratique moyenne') ;
set(get(ax(2),'Ylabel'),'String','Erreur de classification') ;
xlabel ('Nombre de présentation de l''ensemble d''apprentissage') ;
title (sprintf('Statistiques pour %s' , filename)) ;
grid on

stat(end).mat_conf

% Affichage de l'ensemble d'apprentissage
figure(2) ;
axis ([0 1 0 1]) ;
axes = [1 2] ;
plot_exemple (axes , learning_set.matExemple , learning_set.label) ;

% Génération de points dans l'espace d'entrée [0,1]x[0,1]

nb_points = 70 ;
pointsTest = rand(nb_points,learning_set.dim_exemples);

% test du perceptron sur l'ensemble généré
for i = 1:nb_points
    o = PMCPropagation(net , pointsTest(i,:)) ;
    [v cl] = max(o);
    plot_output (axes , pointsTest(i,:)  , cl-1 , max(o)) ;
end


% Affichage des hyperplans séparateurs
for (i = 1:net.dim_cachees)
    plot_hyperplane (net.IH(:,i) , [0 1] , [0 1]) ;
end
```

Listing 2 – Programme principal perceptron multicouches

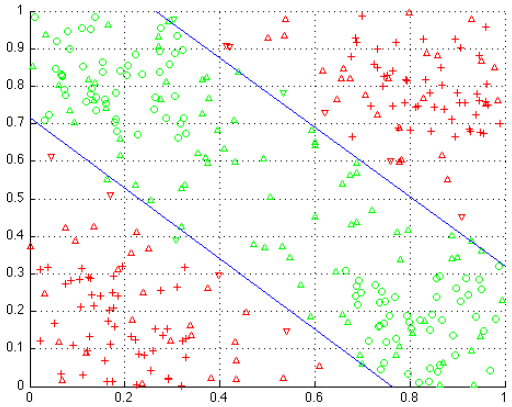
3.2 Apprentissage du PMC

Le fichier **PMCLearning.m** donne le squelette de l'algorithme d'apprentissage du PMC. Complétez le code pour réaliser :

6. le calcul de la mise à jour des poids de HO (couche cachée -> couche de sortie)
7. le calcul de la mise à jour des poids de IH (couche d'entrée -> couche cachée)
8. la mise à jour de la matrice des poids



Erreur quadratique



Hyperplans

FIGURE 2 – Exemples de résultat d'exécution des codes du perceptron linéaire

3.3 Propagation d'un exemple

Le code **PMCPropagation** propage un exemple au travers d'un PMC et calcule la valeur retournée par le réseau. Il appelle la fonction **PMCLayerPropagation** qui assure la propagation au travers d'une seule couche d'un réseau.

9. Ecrire la fonction **PMCLayerPropagation**

3.4 Validation du PMC

Lancez plusieurs apprentissages sur la base d'apprentissage. Observez et commentez :

10. l'influence du paramètre learning_rate sur les statistiques de convergence de l'algorithme
11. la variabilité de la position des hyperplans séparateurs, en fonction des conditions initiales

Pour illustration, la figure 2 présente l'évolution de l'erreur quadratique moyenne en fonction du nombre de présentation de l'ensemble d'apprentissage, et les hyperplans calculés sur les données XOR.

4 Etude d'un cas réel

Les données sont issues de la base de données USPS. Cette base de données contient des images représentant des chiffres manuscrits isolés sur des codes postaux. Cette base est utilisée classiquement pour l'évaluation des méthodes d'apprentissage.

Les données se trouvent dans les fichiers digit10_16x16_learn.txt et digit10_16x16_test.txt. Un exemple dans ces fichiers est une image binaire d'un chiffre codé sur 16×16 pixels. Par exemple, la figure 3 présente un exemple d'un chiffre 3 de la base USPS.

Le code **PMCUSPS.mat** contient le squelette complet pour l'apprentissage d'un PMC à 10 couches cachées, sur ces données.

12. Complétez ce code pour tester le PMC appris sur les données de test, et calculez les statistiques de reconnaissance des chiffres de la base de test.
13. Faites varier le nombre de neurones sur la couche cachée et le taux d'apprentissage. Qu'observez vous ?

5 Regression

Nous allons enfin utiliser les réseaux de neurones pour un problème de régression de données. Le but plus précisément est de réaliser l'apprentissage d'une surface dans $\mathbb{R}^3$ à l'aide d'un ensemble de points se trouvant sur cette surface (problème d'interpolation donc).

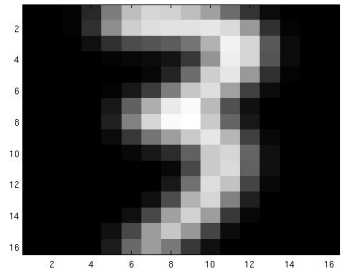
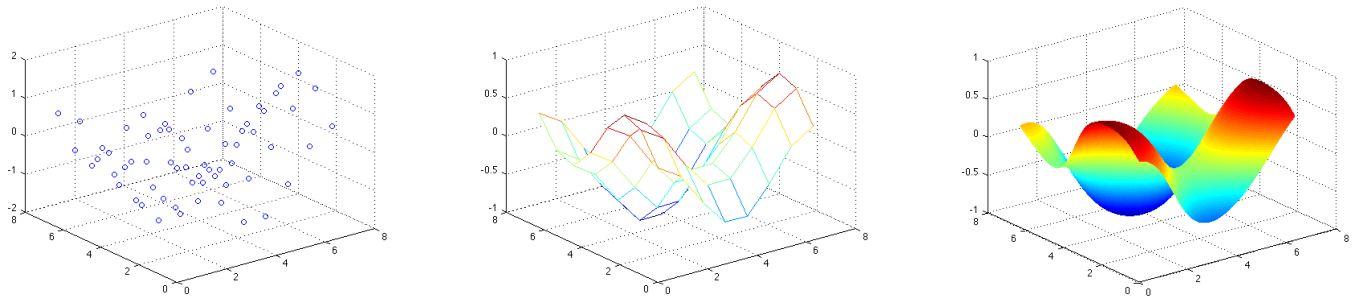


FIGURE 3 – Exemple d'un chiffre présent dans la base USPS



Points d'interpolation | réseau d'interpolation | surface interpolée

FIGURE 4 – Interpolation de surface par PMC

Le fichier **PMCInterpolation.m** propose un squelette de code pour l'interpolation de surfaces à l'aide d'un PMC. La création de l'ensemble d'apprentissage sur $[0, 2\pi] \times [0, 2\pi]$ vous est fournie, la procédure d'apprentissage également.

14. Générez des données de test, et affichez la surface interpolée sur $[0, 2\pi] \times [0, 2\pi]$.
15. Que donne l'extrapolation de données (i.e. le calcul de la valeur de la fonction en un point en dehors de $[0, 2\pi] \times [0, 2\pi]$) ?
16. Faites varier les paramètres du réseau (nombre de neurones sur la couche cachée, taux t'apprentissage). Commentez les changements.

La figure 4 présente un exemple d'interpolation pour la fonction