Université de Jijel
Département d'électrotechnique
Master II Electrotechnique Industrielle
Module : Techniques d'intelligences artificielles

TP-2_ Réseaux de Neurones

1. Utilisation de la Boîte à outils Matlab réseaux de neurones (Neural Network Toolbox «nntool»)

1.1. Création d'un réseau

La fonction de création d'un réseau est spécifique au modèle de réseau utilisé (newc, newp, newrb, etc).
Pour les réseaux multicouches, la création du réseau est commandée par la fonction newff :
reseau=newff(PR, [S1 S2.....SN1] , {TF1 TF2.....TFN1}, BTF , BLF , PF) ;
Avec :
PR : Plage des variations des entrées (affichage par minmax(p)).
Si : nombre des neurones dans la couche i, pour N1 couches.
TFi : fonction d'activation dans la couche i, par défaut la fonction d'activation est 'tansig', elle peut être :

- Hardlim : Fonctions Heaviside.
- hardlims: Fonctions signe.
- logsig : Fonction logarithme sigmoïde.
- tansig : Fonction tangente sigmoïde.
- pureline : fonction linéaire.
- satlins : Fonction linéaire a seuil.

BTF : l'algorithme d'apprentissage par paquets du réseau, la fonction BTF peut être :

- trainlm : apprentissage par l'algorithme de Levenberg-Marquardt
- trainbfg : apprentissage par l'algorithme BFGS.
- trainoss : apprentissage par l'algorithme « one-step BFGS Method »
- trainbr : version de trainlm avec modération automatique des poids.
- trainrp : apprentissage par l'algorithme RPROP.
- trainscg : apprentissage par scaled conjuguate gradient (SCG)
- traincgf : apprentissage par la méthode du gradient conjugué+Fletcher-Reeves.
- traincp : apprentissage par la méthode du gradient conjugué+Polak-Ribiere.

BLF : l'algorithme d'apprentissage incrémental du réseau, la fonction BLF peut être :

- Learngd : l'algorithme d'apprentissage sera la descente de gradient à taux d'apprentissage fixe.
- Learngdm : version de learngd avec momentum.

PF : fonction du coût.

- mae : erreur absolu moyen
- mse : erreur quadratique moyen
- msereg : version de mse avec modérations des poids
- sse : somme des carrés des erreurs

1.2. Apprentissage

Il existe 2 types d'apprentissage :

- Incrémental : fonction adapt.
- Par paquets : fonction train.

Apprentissage incrémental (en-ligne, *on-line*) : les poids sont modifiés à chaque présentation d'une entrée.

Apprentissage par paquets (hors-ligne, *off-line*, *batch mode*) : les poids sont modifiés uniquement après présentation de toutes les entrées.

1.3. Simulation (ou activation) d'un réseau

A = sim(net, p) ;

où net est le pointeur retourné par une fonction de création de réseau.

1.4. Simulation par un block simulink

gensim(net,st) simule le réseau de neurone avec un block en simulink.

1.5. Options

Le taux d'apprentissage

Vous pouvez modifier le taux d'apprentissage de la méthode du gradient de descente par :

net.trainParam.lr = valeur''nouvelle'';

La valeur du Momentum

Le paramètre dynamique dans l'apprentissage du gradient de descente peut être changé :

net.trainParam.mc = valeur'' nouvelle'';

Le nombre d'itérations

Le nombre d'itérations maximum peut être changé par :

net.trainParam.epochs = valeur''nouvelle'';

Valeur de performance

La procédure de minimisation peut être stoppée lorsque l'erreur a atteint un certain «seuil». Vous pouvez changer cela par :
net.trainParam.goal = valeur''nouvelle'';

2. Problèmes de classification des données et d'approximation de fonction

2.1. Problèmes de classification

On peut créer un perceptron par la fonction: net = newp(P,T,TF,LF)
Elle prend comme entrées:,

P - RxQ matrix of Q1 représente le vecteur d'entrée.
T - SxQ matrix of Q2 représente le vecteur de sortie.
TF - fonction de Transfert, par défaut = 'hardlim'.
LF - fonction d'apprentissage, par défaut = 'learnp'.

Questions:

1- Créer un programme d'un perceptron qui fait la classification linéaire de deux classe de données différentes générées aléatoirement. Les deux classes sont générer par :

```
p1=7*rand(2,50) ;
p2=7*rand(2,50) +5;
p=[p1,p2] ;
t=[zeros(1,50),ones(1,50)] ;
```

2.2. Problèmes d'approximation de fonction

1- Mettez en place un réseau de neurones utilisant la méthode de la rétro-propagation du gradient, afin d'élaborer un programme Matlab d'identification de courbes pour la fonction sinus cardinale (sinc).
2- Utilisez le même réseau de neurones pour approximer les fonctions : f(x)=exp(-x)

Une consigne dans ce problème est d'utiliser qu'une seule couche cachée de neurones. L'intérêt sera ensuite de jouer sur le nombre de neurones cachés et d'en déduire leur influence.

3. Evaluation continue

3.1. Expérience d'interpolation de surface

L'objectif est réaliser l'apprentissage d'une surface par un réseau de neurones MLP à partir d'un certain nombre de points de cette surface. Ce réseau doit être capable d'interpoler les coordonnées de n'importe quel point de cette surface.

1- Quelles modifications faut-il apporter à l'architecture du réseau précédent pour se placer dans le cadre d'une interpolation de surface ?
2- Visualiser les surfaces obtenues (apprentissage et test). Vous pouvez utiliser la fonction Matlab mesh pour afficher une surface en 3D.
3- Essayer d'utiliser le réseau en extrapolation, c'est à dire fournir en phase de test des points dont les coordonnées (x,y) sont en dehors de l'espace qui a servi à l'apprentissage. Qu'observez-vous?

On pourra utiliser comme surface de travail $z = \cos(x) + \sin(y)$ avec $(x,y) \in [0\ 2\pi]x[0\ 2\pi]$.

Chargé du module
Pr. T. Hacib