Apprentissage Artificiel ENSTA / INT-22

A. Cornuéjols

Connexionnisme : l'algorithme de Rétro-propagation.

Nous vous proposons dans ce TP de faire des expériences avec l'algorithme de rétro-propagation de gradient dans un Perceptron multi-couches.

Pour cela, nous allons utiliser le simulateur de réseaux de neurones javaNNS (une implémentation en java du Stuttgart Neural Network Simulator).

Vous trouverez le site donnant des détails sur ce simulateur (et un manuel en ligne) à l'adresse :

http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/

Vous avez l'option de charger le programme correspondant pour processeur intel ou AMD. Il semble que la bonne configuration pour vos machines soit : processeur AMD.

1. Prise en main sur une base de lettres

Dans un premier temps, vous allez vous familiariser avec l'utilisation de SNNS en utilisant une base de données fournie avec le logiciel.

Il s'agit d'une base de données sur les 26 lettres de l'alphabet décrites sur une matrice 5x7. La couche d'entrée du réseau comporte donc 35 neurones, tandis que la couche de sortie en compte 26 (un neurone par lettre à reconnaître).

Copiez les fichiers `letters.pat` et `letters.net` dans le répertoire :

http://www.lri.fr/~antoine/Courses/ENSTA/ResNeuro/Letters/

Lancez Windows, ou bien Linux.

Puis utilisez la commande : `java -jar JavaNNS.jar`

Le reste des manipulations est expliqué dans le manuel en ligne sur le site mentionné. Cependant, en voici les grandes lignes.

Dans la **fenêtre de contrôle** :

- Copier les fichiers d'extension `.net` et `.pat` liés à lettres. Ces fichiers décrivent respectivement la configuration du réseau (combien y a-t-il de couches cachées ? et combien de neurones par couches ?), et la manière de coder les données (comment est-ce fait ?)

  À l'issue de cette étape, une fenêtre montrant le réseau (ayant déjà été appris) devrait s'afficher en pressant le bouton `<display>`.
- Ouvrir la fenêtre permettant de suivre la courbe d'apprentissage par le bouton `<graph>`.
- Ouvrir la fenêtre de contrôle permettant de contrôler l'apprentissage par `<control>`
- Dans cette fenêtre, mettre par exemple le paramètre « cycle » à 100

- Appuyer sur `<all>`. L'apprentissage se fait sur 100 présentations de la base d'apprentissage.
- Tester le résultat de l'apprentissage par `<test>`. Que constatez-vous ?
- Répéter.

On peut aussi suivre l'évolution des poids grâce à la fenêtre s'ouvrant avec le bouton `<weights>`. Le codage couleur permet de visualiser le poids associé à chaque connexion. Lorsque vous pointez votre souris sur un pixel de cette fenêtre, les coordonnées de la connexion correspondante et son poids s'affichent.

1.1 Familiarisez vous avec l'outil javaNNS en modifiant les paramètres (nombre de couches cachées, nombre de neurones en couches cachées), l'initialisation, éventuellement les fonctions d'apprentissage, etc.

1.2 Faites des expériences en bruitant certaines des entrées, c'est-à-dire en modifiant aléatoirement des valeurs des descripteurs.

2. Construction de Perceptrons Multi-Couches

Pour la construction de Perceptrons Multi-Couches (PMC) avec JavaNNS, il faut se reporter au manuel d'utilisation « JavaNNS-manual.pdf » disponible sur le site de JavaNNS, en page 10.

1. Dans un 1er temps, vous créerez un PMC à 2 entrées, une couche cachée de 2 neurones et une couche de sortie de 1 neurone afin d'apprendre la fonction XOR, dont les exemples d'apprentissage sont disponibles dans le fichier XOR.pat.
   Vous regarderez en moyenne combien il faut de cycles d'apprentissage pour apprendre cette fonction.
2. Vous testerez ensuite d'autres architectures (ex : 2-1-1 ; 2-3-1 ; 2-5-1 ; 2-10-1) et vous examinerez à nouveau le temps d'apprentissage.
   Vous créerez un jeu de données bruitées et vous examinerez les performances en généralisation des différentes architectures sur ce jeu de données.
3. Faites la même étude sur une fonction XOR à 3 entrées (vous trouverez les jeux de données sur le site : 'xor_rec1.pat' et 'xor_rec2.pat')
4. Généralisez l'étude précédente à une fonction de 5 variables d'entrée qui retourne 1 quand le nombre de variables d'entrée à 1 est pair et 0 sinon. C'est ce que l'on appelle une *fonction parité*. Vous construirez des bases d'apprentissage et de test pour ce problème.

3. Devoir à la maison : apprentissage de lettres dans un cadre plus général

A RENDRE POUR LE **JEUDI 11 JUIN matin**

Vous trouverez une base de données décrivant des lettres définies sur des matrices 6x8 et 7x9 et systématiquement décalées ou bruitées à l'adresse suivante :

http://www.lri.fr/~antoine/Courses/ENSTA/ResNeuro/Letters-IIE/

Copiez ces données.

Les explications figurent dans le fichier « Source download »
(http://www.lri.fr/~antoine/Courses/ENSTA/ResNeuro/Letters-IIE/Source download).

Puis réfléchissez à la structure d'un réseau de neurones permettant de les exploiter. Quel codage pour les entrées (et donc combien de neurones sur la couche d'entrée) ? Quel codage pour la sortie ? Combien de couches cachées et combien de neurones sur chacune ?

Construire le réseau correspondant dans un fichier d'extension .net

Créer le fichier de description des données d'extension .pat

Créer le fichier des exemples (suivant les spécifications du fichier d'extension .pat créé).

Faites vos expériences.

Qu'en concluez-vous ?

3. Compléments sur l'utilisation de SNNS

Lancement 'xgui'

Cliquer sur 'bignet' option 'feed_forward'

Couche d'entrée 'input' dimensions 4x1 'enter', 'insert'

'type'

hidden 2x5 'enter', 'insert'

hidden 2x5 'enter', 'insert'

'type'

ouput 1x1 'enter', 'insert'

'full connection'

'create net' 'done'

(retour au snns-manager)

Signaler les cellules cachées et 'u s f a' (le curseur doit être dans la fenêtre 'display') (Unit Set Activation function)

Choisir 'Act-Tanh' 'done'

Signaler les cellules de sortie et taper 'u s f a'

Choisir 'Act-Identity' 'done'

Désélectionner les cellules avec shift click

Visualiser les liens : dans fenêtre 'display' faire 'setup', devant 'link' cliquer 'on', 'done'

Associer les fichiers de données :

Dans le snns-manager : 'file', 'PAT', double-cliquer sur le fichier 'load' pour l'apprentissage

Lancement de l'apprentissage

Dans le snns-manager : 'control' et 'graph'

Choix de la fonction de rétropropagation pour LEARN (dans control) :

'SEL_FUNC' en face de LEARN, choisir par exemple 'SCG' pour le gradient conjugué.

Dans LEARN écrire le pas d'apprentissage (ex. 0.2) et 0.0 pour les autres cases.

L'ordre topologique de propagation du signal d'entrée :

'SEL_FUNC' en face de UPDATE, choisir 'topological order'

Initialisation des poids

'SEL_FUNC' en face de INIT, choisir 'randomize_Weights', réduire l'intervalle d'initialisation (ex: [-0.02, 0.02])

Cliquer 'INIT'

Mettre le nombre total de cycles et le nombre de cycles avant le test (CYCLES et VALID)

Cocher 'shuffle' pour une présentation aléatoire des exemples durant l'apprentissage (gradient stochastique)

Sélection du fichier d'apprentissage et de test :

'USE', choisir le fichier correspondant de l'apprentissage

'USE', devant VALID choisir le fichier correspondant au test.

Lancer l'apprentissage

'RESET' pour reprendre le cycle à zéro.

ALL

(ajuster CYCLES et VALID, lancer ALL jusqu'à ...)

Récupération des résultats

Les poids : (dans snns-manager) : 'FILES', NET, donner un nom et 'save'