DÉPARTEMENT SYSTÈMES FORMELS ET INTELLIGENTS

RÉSEAUX DE NEURONES ARTIFICIELS – TP N° 3

16 février 2009

MÉTHODES DE RÉTROPROPAGATION AVEC javaNNS et SNNS

L'objectif du TP est l'étude des méthodes de rétropropagation à l'aide de deux ensembles d'exemples : restitution d'un codage binaire et reconnaissance des lettres pixelisées.

Exemple 1.- Soit un RNA qui accepte en entrée un mot binaire de 8 bits, qu'il restitue à la sortie.On utilisera

– pour l'architecture du système le fichier `codeur.net`, et
– pour les données en entrée-sortie le fichier `codeurTr.pat`.

Ces deux fichiers font partie du fichier compacté `codeur.zip` que vous trouverez sur le site du cours.

Partie A Étude des quatre méthodes d'apprentissage par rétropropagation.

1. Première méthode à tester, la rétropropagation standard (`Std_bakpropagation`). On prendra
   – taux d'apprentissage $\eta = 2.0$ (En cours on a vu que $\eta \in]0, 1]$. Néanmoins pour des problèmes avec peu d'exemples on accélère la convergence en prenant $\eta > 1$. (cf. page 69 du manuel SNNS)).
   – La différence maximale tolérée entre sortie calculée et sortie désirée $d_{\max} = 0$. Au délà de cette limite, on considère qu'il y a erreur et on procède à l'adaptation des poids.
2. Deuxième méthode, la rétropogation avec moment (`BakpropMomentum`) avec
   – taux d'apprentissage $\eta = 0.8$;
   – moment $\alpha = 0.6$;
   – constante ajoutée à la dérivée de la fonction d'activation pour permettre au réseaux de dépasser les régions plates de la surface d'optimisation $c = 0.1$, et
   – $d_{\max} = 0$.
3. Troisième méthode, la méthode `Quickprop`. Cette méthode, comme aussi la suivante, est une méthode de rétropropagation en ligne. Les deux méthodes précédentes sont des méthodes de rétropropagation batch. Il y a adaptation des poids après l'examen de chaque exemple. Au contraire, pour cette méthode et la suivante l'adaptation se fait après examen de tous les exemples (cf. pages 70-71 du manuel SNNS). On prendra
   – taux d'apprentissage $\eta = 0.2$;
   – moment $\alpha = 0.1.75$ et
   – taux de décroissance des poids $\nu = 0.0001$.
4. Quatrième méthode, la méthode `Rprop` avec facteur de décroissance de l'adaptation des poids $\eta_1^- = 0.2$, les autres paramètres étant mis à zéro.

Pour chacune de ces méthodes

1. Vous chargerez le fichier `codeur.net` ;
2. Vous prendrez `nombre de cycles = 50` ;
3. Vous choisirez l'ordre topologique pour l'entrée des exemples (`topological_order`)
4. Vous n'utiliserez pas les boutons `SUFFLE` et `INIT`.
5. Vous appuierez sur le bouton `ALL`.

Comparer la vitesse de convergence.

Partie B Comparaison entre les méthodes batch et en ligne.

Pour les deux prémières méthodes (méthodes batch), réfaire les cinq étapes précédentes, en utilisant maintenant `SUFFLE` mais en n'utilisantt toujours pas `INIT`.

Comparer la vitesse de convergence avec les méthodes en ligne et commenter.

Partie C Étude de l'influence des valeurs initiales.

Repéter les étapes 1 à 5 précédentes, en utilisant `SUFFLE` pour les deux premières méthodes et `INIT` pour toutes les méthodes.

Comparez et commenter.

Partie D Étude de la capacité de généralisation.

Nous allons maintenant s'intéresser à la capacité prédictive du réseau. On créera un fichier test `codeurTs.pat` avec les patterns en entrée et sortie suivants :

`00000001, 11100000, 00011100, 00110000, 00001100, 10100000, 0000101(`

Pour cette partie il est obligatoire d'utiliser SNNS et faire du calcul du réseau et de la validation en même temps.

Exemple 2.- Considérons un RNA qui est capable de reconnaître les 26 lettres majuscules de l'alphabet latin.L'architecture du réseau est donnée par le fichier `lettres.net` et sa configuration par le fichier `lettres.cfg`. Les données sont dans le fichier `lettres.pat`. Ce trois fichiers sont compactés dans le fichier `lettres.zip` que vous trouverez sur le site du cours.

Partie A On utilisera la méthode de rétropropagation standard avec $\eta = 0.5$ et $d_{\max} = 0$. On utilisera aussi le bouton `SUFFLE` et on prendra `CYCLES=10000`. On appuiera sur `INIT` et `ALL`. Si la convergence n'est pas bonne à la fin de 10000 itérations, on appuiera de nouveau sur `ALL` et ceci jusqu'au moment où on aura une convergence acceptable. Nous pouvons utiliser le bouton `TEST` pour vérifier la qualité de la reconnaissance.

Partie B On étudiera maintenant la tolérance du réseau au bruit. On copiera le fichier `lettres.pat` sur le fichier `lettresTs.pat`. À l'aide d'un éditeur de texte on mettra du bruit sur les lettres soit en transformant quelques 0 à 1, soit en transformant quelques 1 à 0. Ensuite charger ce fichier et, en utilisant le bouton `TEST`, examiner le comportement du réseau.

En particulier étudier l'influence du bruit sur la reconnaissance

- des lettres C, D, O et Q ;
- des lettres E et F ;
- des lettres I et J ;
- des lettres P et R.