Reconnaissance de chiffres manuscrits et PMC
Lançons-nous

Quelques indications pour se lancer dans la programmation d'un PMC.

1 Structures de données

Matrices. Le fonctionnement d'un PMC repose pour une grande part sur des calculs matriciels (des produits matrice×vecteur, notamment). Il peut donc être très utile d'avoir une structure de données appropriée pour représenter des matrices ; pour rappel, un vecteur est une matrice dont le nombre de colonnes (ou de lignes) est égal à 1. Afin d'avoir le plus de flexibilité possible, ces matrices doivent pouvoir être de taille variable, sans spécification au préalable d'une taille maximale (il faut donc faire des allocations et désallocations de la mémoire).

PMC. Evidemment, la structure de données essentielle de ce programme est celle permettant de représenter un perceptron multi-couches. Comme vu à plusieurs reprises, un PMC est simplement défini à partir de la taille de sa couche d'entrée, de sa couche cachée et de sa couche de sortie, ainsi que des poids des connexions, qui peuvent être stockées dans des matrices de taille adéquate.

2 Fonctions principales

Quelques en-têtes de fonctions. On suppose que les types utilisées (pmc et matrice, notamment) sont des pointeurs sur des **struct**.

2.1 Matrices

```
/* Crée une matrice de n lignes et m colonnes */
matrice cree_matrice(int n, int m);

/* Fait le produit de deux matrices */
matrice produit(matrice a, matrice b);

/* Fait le produit d'un scalaire par une matrice */
matrice produit_scalaire(float s, matrice a);

/* Fait la somme de deux matrices */
matrice somme(matrice a, matrice b);
```

2.2 PMC

```
/*
    Crée un PMC avec
    e neurones sur la couche d'entrée
    c neurones sur la couche cachée
    s neurones sur la couche de sortie
```

```
    et initialise les poids du pmc aléatoirement
*/
pmc cree_pmc(int e, int c,int s);

/* Réinitialise les poids du pmc */
pmc reinitialise(pmc p);

/*
    Calcule le vecteur de sortie à l'aide des équations données
    dans le descriptif du sujet. X est le vecteur d'entrée (il fait
    donc la même taille que la couche d'entrée)
*/
matrice propage(pmc p, matrice X);

/*
   Fait une mise à jour du pmc en fonction d'un couple (X,Y)
   en utilisant l'algorithme de rétro-propagation du gradient
*/
pmc backprop(pmc p, matrice X, matrice Y);
```

2.3 Fichiers

```
/*
  Sauvegarde un pmc dans un fichier et renvoie 1 si
  l'opération a réussi et 0 sinon
*/
int enregistre(char *filename, pmc p);

/*
  Charge un pmc à partir d'un fichier et renvoie 0 si l'opération
  a échoué et un pointeur correspondant au pmc sinon
*/
pmc charge(char *filename);
```