

**Université d'Avignon – Master 1****Intelligence Artificielle – Approches Neuronales – TP 3****Juan-Manuel Torres - 2024**

**Travaillez individuellement ou en binôme. Langages acceptés : C/C++ ; python ; awk, rust, perl ; ruby – Exclut : R, java, javascript**  
**Interdit d'utiliser des fonctions/librairies des RN natives (R, python, matlab, etc)**

**Théorie**

L'algorithme d'apprentissage Minimererror utilise une fonction de coût qui dépend d'un hyper-paramètre appelé température  $T > 0$ ; qui peut être écrit comme  $\beta = 1/T$ . Il a les propriétés de trouver un hyperplan séparateur de stabilité maximale si l'ensemble d'apprentissage est LS ou s'il ne l'est pas, de trouver le nombre minimum de fautes ayant des stabilités maximales. Utilisez la fonction de coût, sa dérivée et une stratégie de recuit déterministe pour programmer l'algorithme minimerror.

Pour ce TP, en particulier pour les paramètres (initialisation, delta de températures, delta du recuit, etc) consulter thèse de M *Torres et les slides Minimererror*, disponibles sur ENT.

**Il s'agit d'un algorithme difficile : Il n'y a pas UNE seule implantation de cette algorithme : construisez votre propre version de l'implémentation**

**Travail pratique. Données au choix selon****PARTIE I**

**Minimererror avec 1 température  $\beta = 1/T$ .** Programmer l'algorithme minimerror pour apprendre sur l'ensemble « train », puis tester sur l'ensemble de « test ». Les données à tester NE DOIVENT PAS CONTENIR LEUR CLASSE.

- a) Calculer les erreurs d'apprentissage  $E_a$  et de généralisation  $E_g$  ;
- b) Afficher les  $N+1$  poids  $\mathbf{W}$  du perceptron ;
- c) Calculer les stabilités des  $P$  exemples de « test » (distance a l'hyperplan séparateur avec les poids normés)
- d) Graphique des stabilités

Note : Il est possible que  $E_a > 0$  en fonction des paramètres

**PARTIE II**

**Minimererror avec 2 températures  $\beta_+ / \beta_- = \text{rapport de températures}$ .** Modifier l'algorithme minimerror pour apprendre avec 2 températures sur l'ensemble « train », puis tester sur l'ensemble de « test ». Les données à tester NE DOIVENT PAS CONTENIR LEUR CLASSE.

- a) Calculer les erreurs d'apprentissage  $E_a$  et de généralisation  $E_g$  ;
- b) Stocker les  $N+1$  poids  $\mathbf{W}$  du perceptron dans un fichier;
- c) Calculer les stabilités des  $P$  exemples de « test » (distance a l'hyperplan séparateur avec les poids normés)
- d) Graphique des stabilités en fonction de  $\beta = 1, 2, \dots, 10$

**Rapport.** Pour chaque partie : Votre programme,  $E_a$ ,  $E_g$ ,  $P$  stabilités,  $N+1$  poids, réponses aux questions. Rendu PDF et codes sources compressés sur ENT