Université d'Avignon – Master Intelligence Artificielle
Apprentissage supervisé – TP 2
Juan-Manuel Torres - 2025 Binômes ou monômes

Langages acceptés : C/C++ ; python ; awk, rust, perl ; ruby – Exclus : R, java, javascript
Interdit d'utiliser des fonctions/librairies des RN natives (R, python, matlab, etc).
IA interdite, IN à volonté

Théorie

Le serveur UCI (machine learning datasets) https://archive.ics.uci.edu contient beaucoup des données pour tester les algorithmes d'apprentissage supervisé. Dans ce TP nous allons utiliser l'ensemble de données des échos de sonar codées en plusieurs dimensions, proposé par Tierry Sejnowsky. Ceci est un ensemble classique dans le domaine des Réseaux de neurones. Les données et leur description se trouvent ici :
https://archive.ics.uci.edu/dataset/151/connectionist+bench+sonar+mines+vs+rocks

Pour ce TP vous pouvez consulter le papier *Torres & Gordon, Characterization of the Sonar Signals Benchmark*, https://link.springer.com/article/10.1023/A:1009605531255 qui est disponible sur le site du cours dans le fichier SONAR_torres.pdf

Travail pratique

1 **Données**. Télécharger les données du Sonar depuis UCI. Construire 3 ensembles d'apprentissage : train (P=104 données marquées avec une *), test (P=104 données) et ensemble complet (208 données). Le nombre de dimensions est N=60. Formater un écho par ligne, séparation de colonnes par des tabs.

PARTIE I

2 **Apprentissage sur « train »**. Utiliser l'algorithme du perceptron (justifier le choix **version batch** vs **online** selon votre TP1**)** pour apprendre l'ensemble « train », puis tester sur l'ensemble de « test ».

a) Calculer les erreurs d'apprentissage Ea et de généralisation Eg ;
b) Afficher les N+1 poids **W** du perceptron ;
c) Calculer les stabilités des P exemples de « test » selon la formule de gamma (distance a l'hyperplan séparateur avec les poids normés)
d) Graphique des stabilités

3. **Apprentissage sur « test », cad inverser les ensembles :** Apprendre sur l'ensemble « test », puis généraliser sur l'ensemble « train ». Calculer a), b) et c) du point précédent 2.

PARTIE II

4 Programmer l'algorithme d'apprentissage Pocket : il garde le meilleur résultat de l'algorithme du perceptron en fonctions des itérations et en fonction d'un Nb d'erreurs prédéfini.
Apprendre sur l'ensemble « train » en stoppant l'erreur d'apprentissage Ea fixé à l'avance, puis tester la généralisation Eg sur l'ensemble de test.
Tester aussi l'initialisation aléatoire vs une initialisation de Hebb. Refaire les expériences en échangeant les données « train » et « test ». Que pouvez vous dire de Ea et Eg en fonction des ensembles? et en fonction de l'initialisation? En fonction du eta (pas d'apprentissage)? Montrer vos résultats sous forme de tables ou de graphiques (bonus).

PARTIE III

5 Apprentissage sur « train + test ». Utiliser l'algorithme du perceptron pour apprendre l'ensemble fusionné L = L(train)+ L(test) (avec leur classe tau)
L'ensemble L est-il LS ou pas LS ? Justifier votre réponse en fonction des calculs de vos programmes.

PARTIIE IV

6. Early Stopping. Utilisez l'ensemble L : Train+Test (208 patrons) pour créer 3 ensembles au hasard:
LA = Apprentissage 50 % des données
LV = Validation 20 % des données
LT = Test 30 % des données

Faites l'apprentissage sur LA, validez l'erreur sur LV et avec Early Stopping testez sur LT.
Répétez cette expérience plusieurs fois afin d'obtenir des statistiques sur la moyenne des erreurs Ea, Et, et Ev (validation).

6 Rapport. Pour chaque partie : Ea, Eg, (Ev en plus pour la partie IV), P stabilités, N+1 poids, réponses aux questions. Rendu PDF du Latex et codes sources compressés ZIP/GZIP su ENT.

Que la force soit avec vous !