AVIGNON UNIVERSITÉ

Approches Neuronales – TP 2 Perceptron

Sarra Bensafi

25 février 2022

Informatique
Master1 Intelligence Artificielle
UCE Approches Neuronales

Responsables
Juan-Manuel Torres

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Lecture des Données

En premier j'ai lu les données Mines pour train/test, puis j'ai lu les données Rocks train/test et j'ai fusionné les deux, ensuite j'ai créé les fichiers contenant les étiquettes M ou R selon la source de la donnée si rocks ou mines.

2 Partie I

2.1 Apprentissage sur « train » et « test »

J'ai utilisé l'algorithme du perceptron online puisque le coût de calcul est plus faible en utilisant ce dernier et le nombre des itérations est moindre et toujours inférieur à ceux de la version BATCH.

2.1.1 Calcul des erreurs(Ea Eg) , stabilité et N+1 poids

Ea qui est l'erreur d'apprentissage nous l'obtenons en calculant l'erreur de classification d'un modèle à partir des mêmes données sur lesquelles le modèle a été entraîné. Dans la partie apprentissage avec Train le Ea=2 et pour l'apprentissage avec Test Ea=1.

Eg qui est l'erreur de généralisation nous l'obtenons en utilisant deux ensembles l'un pour entraîner le modèle et l'autre pour calculer l'erreur de classification un ensemble de test qui ne doit pas avoir été utilisé pour l'apprentissage.Dans la partie apprentissage avec Train le Eg=0.8 avec 23 erreurs et pour l'apprentissage avec Test Eg=0.74 avec 28 erreurs.

J'ai pu observer que le nombre d'itérations lors de l'apprentissage de train est de 110286 est extrêmement élevé par rapport à celui de test 32832.

On peut remarquer pour la stabilité dans le premier graphic, la plus haute colonne appartient à l'intervalle 0.0,0.1, les valeurs entre 0.0 et 0.2 c'est-à-dire la majorité dès valeurs sont positives c'est à dire bien classées, ce qui prouve qu'on a une grande robustesse de la réponse du perceptron.le reste est négatif ce qui nous laisse comprendre que les données sont mal classées(23 valeurs).

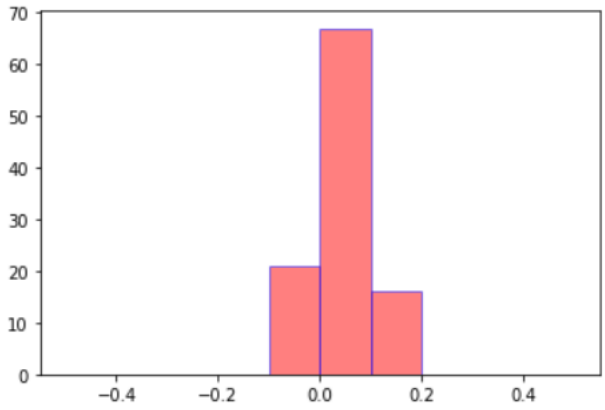

Figure 1. Graphique des stabilités pour l'apprentissage sur Train.

Pour la stabilité dans le deuxième g graphic, les valeurs entre 0.1 et 0.2 sont positifs et bien classée (76 valeurs), ce qui prouve qu'on a une grande robustesse de la réponse du

perceptron aussi.le reste est négatif ou proche de 0.0 ce qui nous laisse comprendre que les données sont mal classées represetant ainsi 28 erreurs.

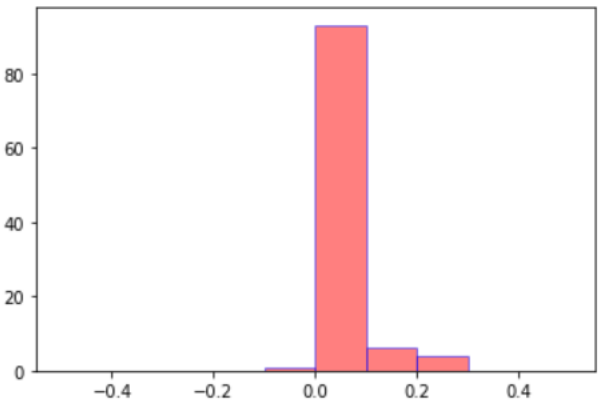

Figure 2. Graphique des stabilités pour l'apprentissage sur Test.

Les W poids après Apprentissage sur train [-0.57 1.04276 1.512476 -0.107876 0.898778 0.445828 0.977096 -0.59398 -0.942866 0.014884 0.666954 0.131872 0.210206 0.053092 0.022598 0.61268 -1.537268 -0.31116 1.241562 -0.159616 0.016476 -0.536048 0.582988 -0.426124 0.73657 -0.039444 -0.02566 -0.0568 -0.019098 -0.348296 1.56466 -2.232274 1.606986 -1.058436 0.52115 -0.340012 -0.107696 -0.281232 -0.103466 0.963358 -0.835666 0.315268 -0.137372 0.102532 -0.539004 0.768824 0.451216 -0.265826 1.080314 1.972866 -0.471236 0.51434 0.24726 0.049444 0.995906 0.179026 0.14788 -0.544252 -0.379126 0.669384 -1.016136] Les W poids après Apprentissage sur test [-0.24 0.668386 -0.691352 -0.79171 0.484454 0.191868 0.197314 -1.22146 -0.545696 0.528988 -0.38749 1.274684 0.483058 0.324872 -0.325992 -0.408296 0.394126 0.334876 -0.53687 -0.489306 0.602444 -0.01076 0.016736 0.152616 0.319944 -0.344132 -0.30433 -0.198678 0.777444 -0.86336 0.7464 -0.332136 -0.111936 0.09007 -0.494058 0.523572 -0.663472 -0.269456 -0.583902 0.771656 -0.485448 0.620942 -0.212654 0.708942 0.590702 0.618468 -0.19502 0.95661 -0.308336 -0.259476 0.623458 -0.471866 -0.37398 0.851418 0.615894 0.29474 0.577178 0.097188 -0.881556 -0.311892 -0.171424]

3 Partie II

En résumer de ce que j'ai compris de l'algorithme pocket, c'est qu'il garde en mémoire le meilleur résultat qu'il a rencontré. Si les poids nouvellement calculé produisent un plus petit nombre d'erreurs de classification que les poids déja calculé, alors on remplace les poids déja calculé par les nouveaux poids.

3.1 Variation de l'algorithme pocket avec Hebb et Random

Nous avons établi deux initialisations(heb et random) distinctes de l'algorithme pocket avec les modifications des hyperparamètres alpha et erreur d'apprentissage fixé (Ea) selon les ensembles Train et Test. Remarque : sachant qu'Erreur de généralisation = Nombre de bonnes classifications/nombre totales, mais tableau j'ai considéré le nombre d'erreurs commises par le système lors de la généralisation pour qu'il me soit facile de comparer Ea et Eg sur la même échelle.

3.1.1 Ensembles Train

Dans ce tableau si je fixe Alpha, l'erreur d'apprentissage diminue et l'erreur de généralisation augmente pour atteindre un maximum de 24 erreurs pour alpha 0,7 et 0,4. Pour alpha=0.01 l'erreur d'apprentissage augmente mais l'erreur de généralisation diminue légèrement pour atteindre 22 erreurs. Si je fixe ErrLimit et j'augmente Alpha je peux remarquer que pour ErrLimit=8 et 10 Eg diminuent.

ErrLimit\Alpha	Alpha=0.7	Alpha=0.4	Alpha=0.01
E=3	Ea=20/Eg=21	Ea=25/Eg=30	Ea=9/Eg=23
E=6	Ea=14/Eg=22	Ea=14/Eg=22	Ea=7/Eg=24
E=8	Ea=12/Eg=23	Ea=13/Eg=23	Ea=12/Eg=22
E=10	Ea=13/Eg=24	Ea=14/Eg=23	Ea=14/Eg=22

Table 1. Tableau des erreurs avec apprentissage sur Train Initialisation HEBB

Dans ce tableau si je fixe Alpha, l'erreur de généralisation diminue pour atteindre un minimum de 21 erreurs pour alpha 0,7 et 0,4. Pour alpha=0.01 Eg augmente. Si je fixe ErrLimit, Eg diminue pour ErrLimit =8,10.

ErrLimit\Alpha	Alpha=0.7	Alpha=0.4	Alpha=0.01
E=3	Ea=13/Eg=23	Ea=6/Eg=23	Ea=9/Eg=22
E=6	Ea=7/Eg=25	Ea=6/Eg=23	Ea=14/Eg=22
E=8	Ea=13/Eg=21	Ea=12/Eg=22	Ea=12/Eg=23
E=10	Ea=12/Eg=21	Ea=12/Eg=21	Ea=16/Eg=24

Table 2. Tableau des erreurs avec apprentissage sur Train Initialisation Random

Pour l'ensemble Train, je déduis que les deux initialisations sont appropriées, ce qui n'affecte pas beaucoup la capacité du modèle à généraliser 0.8 et avec Alpha petit entre à 0.4 et 0.1, ErrLimit=8,10 et plus l'alpha est petit, mieux notre modèle généralise.

3.1.2 Ensembles Test

Après avoir examiné les deux tableaux, je remarque qu'il n'y a pas un énorme changement en ce qui concerne l'erreur d'apprentissage par rapport au Train , mais le nombre d'erreurs commises pendant la généralisation est nettement supérieur arrivant jusqu'à 33 erreurs.

ErrLimit\Alpha	Alpha=0.7	Alpha=0.4	Alpha=0.01
E=3	Ea=6/Eg=22	Ea=6/Eg=22	Ea=6/Eg=22
E=6	Ea=7/Eg=22	Ea=8/Eg=23	Ea=9/Eg=27
E=8	Ea=10/Eg=28	Ea=11/Eg=28	Ea=13/Eg=32
E=10	Ea=11/Eg=27	Ea=11/Eg=27	Ea=10/Eg=27

Table 3. Tableau des erreurs avec apprentissage sur Test Initialisation RANDOM

ErrLimit\Alpha	Alpha=0.7	Alpha=0.4	Alpha=0.01
E=3	Ea=6/Eg=23	Ea=5/Eg=22	Ea=5/Eg=21
E=6	Ea=9/Eg=24	Ea=7/Eg=25	Ea=9/Eg=24
E=8	Ea=9/Eg=26	Ea=9/Eg=28	Ea=9/Eg=22
E=10	Ea=18/Eg=33	Ea=11/Eg=26	Ea=18/Eg=32

Table 4. Tableau des erreurs avec apprentissage sur Test Initialisation HEBB

3.1.3 Conclusion

En ce qui concerne les initialisations, les deux sont équivalentes, le type Hebb ou Random n'affecte pas les résultats. En ce qui concerne l'ensemble d'entraînement, par contre, nous remarquons qu'il y a une différence importante puisque pour l'ensemble Train est plus efficace que celui de Test, Je trouvais que c'était dû à une mauvaise construction de cet ensemble et qu'ils n'avaient pas la même variance que l'ensemble d'apprentissage. En ce qui concerne Alpha, si ce dernier est inférieur à 0.4/0.5 et supérieur à 0.01, meilleurs sont les résultats.

Nous pouvons affirmer qu'un perceptron avec un Eg=21 sur l'ensemble de test semble mieux généraliser qu'un perceptron avec un Eg de 33.Cela revient à dire que notre modèle est capabled'apprendre à partir des données et à appliquer les informations apprises sans sur-apprentissage ou sous-apprentissage.

4 Partie III

Je peux constater que mon ensemble L qui fussionne le train et le test est linéairement séparble, puisque il itére en un temps fini en trouvant toutes les étiquettes correctement en séparant les Rocks et les mines

Les W poids après Apprentissage sur train +Test [-20.37 44.67616 7.45938 -77.4384 31.8311 -14.25732 19.41642 -26.17504 -23.44302 33.03302 -14.41582 15.09324 26.65728 -14.1605 2.2862 9.27246 -14.907 -14.84644 17.3747 -12.08038 28.2079 -32.31402 39.77508 -29.11858 31.76712 -13.74618 -5.90664 12.28664 -5.84156 -4.36014 31.1778 -42.45838 16.98742 9.7227 -17.51696 12.68978 -2.5428 -19.06658 4.10518 17.54106 -24.85378 7.81882 4.02272 7.04068 6.23978 -5.76082 19.38624 -1.17 44.85146 54.66948 -273.85366 94.7755 165.95296 97.39164 19.77682 11.35596 -75.1454 -62.0724 83.55752 63.83316 12.40762]