UNIVERSITÉ DE STRASBOURG

Projet d'Intelligence artificielle licence 3 informatique

Apprentissage bout-en-bout sur des données

Analyse de données synthétiques par Réseau de Neurones Artificiel

BELARIBI NADJIB
MAMMADOV ALI

03/05/2020

1. *Introduction*

Le projet couvre un algorithme d'apprentissage automatique***: le perceptron multicouches MLP***. La méthode est appliquée à un ensemble de données contenant des échantillons de 3 classes, ce qui est un problème de classification multi-classes. Le but du projet est de construire et entrainer le classifieur sur l'ensemble de données afin d'obtenir la meilleure précision possible .

2. *Préparation* des données

Réponses

- Ces données comportent 2 attributs *X1 et X2* .
- Ces données sont séparables en 3 classes différentes *(class-0 , class-1 , class-2*) .
- Non , les données ne sont pas linéairement séparables. En effet la fonction correspondante est non linéaire : $X1^2 + X2^2 = L^2$ (L c'est la distance entre l'origine et le point (X1,X2))

Chargement et préparation des données

- ***test_final_df*** contient 20% des données choisies d'une manière aléatoire pour le test et ***gaussian_df*** contient le reste (80% des données initiales) pour l'entraînement du modèle

3. *Construction du modèle*

1. Le modèle comporte une liste de 4 matrices de poids de la forme dans l'ordre : [4,2] , [3,4] , [2,3] , [3,2] et une liste de 4 vecteurs de biais de taille 4,3,2,3 respectivement .

2.Quand le taux d'apprentissage $\eta = 0.01$ (bas), le changement des paramètres (poids et biais) progressent très lentement car on effectue de très petites mises à jour des paramètres . Cependant, si le taux d'apprentissage passe à $\eta = 0.1$ (élevé) , cela peut entraîner des comportements divergents indésirables dans la fonction de perte (*loss function*) .

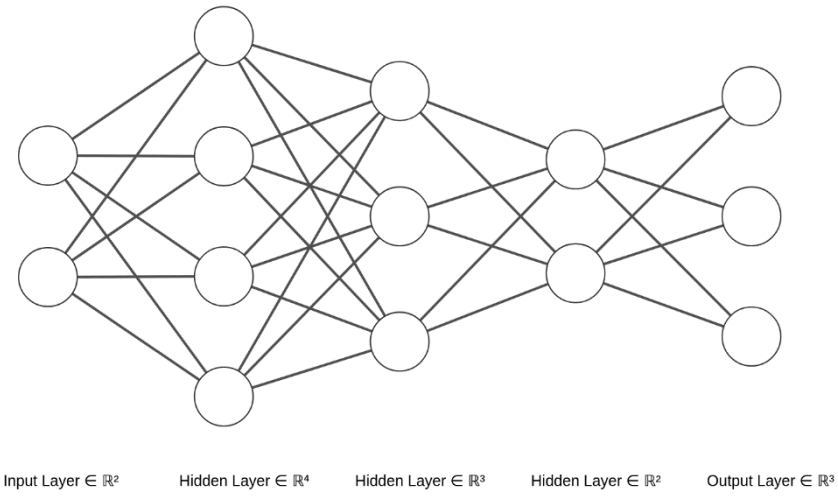

Figure 1 : Architecture MLP

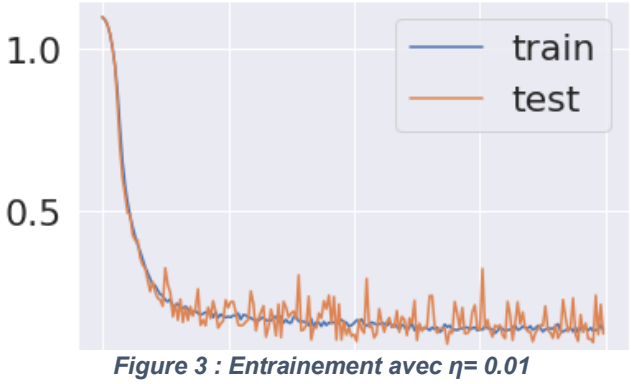

Figure 3 : Entrainement avec η= 0.01

Figure 2 : Entrainement avec η= 0.1

4.Analyse du modèle

1. La ***class-2*** est prédite pour l'instance i = 0 , oui elle est correcte car la probabilité correspondante est la plus haute .

2. On a obtenu un pourcentage de précision (**93%** - ***97.7%***) **96%** au moyen , de prédictions correctes réalisées par notre modèle

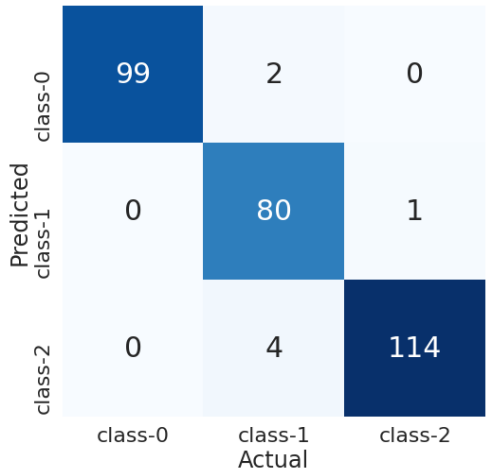

Figure 4 : Maximum Accuracy confusion matrix

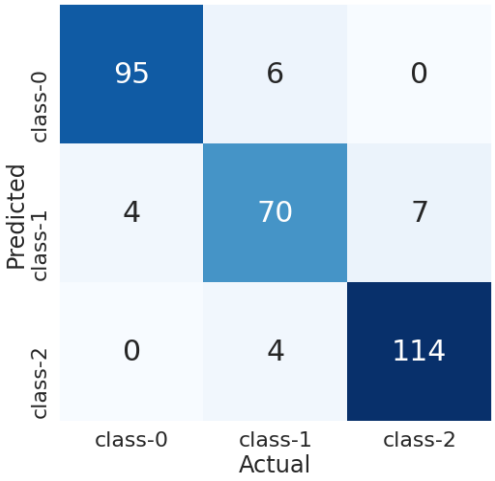

Figure 5 : Minimum Accuracy confusion matrix

3. Les erreurs les plus fréquentes sont liées à la ***class-1***, car ses données se situent entre la ***class-0*** et la ***class-2***. Le modèle prédit de manière erronée les valeurs qui se situent aux extrêmes (proches des points de ***class-0*** et ***class-2***).

4. Les valeurs placées sur la diagonale sont prédites correctement et celles qui sont en dehors la diagonale ne le sont pas . La somme des éléments diagonaux est le nombre de prédictions correctes , ce qui revient à la formule de ratio .

5. Démarches suivies et conclusion :

Tout d'abord, nous avons suivi les instructions données, pour la fonction d'activation et la fonction de transfert. Le premier problème que nous avons eu était les sorties négatives de la fonction ***tanh()*** pour la couche de sortie. Nous avions deux méthodes pour résoudre ce problème, la première c'est utiliser la fonction *MSE*, la seconde c'est remplacer la fonction d'activation sur la couche de sortie par ***softmax() ,*** avec ***cross_entropy_loss()*** pour calculer l'erreur. les résultats obtenus ont montré que la seconde méthode était meilleure et plus adaptée a notre problème de classification .

Pour l'initialisation des poids, nous avons utilisé l'initialisation *Xavier **:** gaussien avec moyenne = 0 et std = 1 / sqrt (n_input_neurons)*, La précision que nous avons obtenue était assez bonne, mais avec des recherches sur Internet, nous avons remarqué que nous pouvions améliorer cela . *Comment ? on m*ultiplie le delta avec la dérivée de ***softmax():*** *delta [L] = (pred - y) * df [l] .* C'était un peu délicat car la dérivée de ***softmax()*** renvoie une matrice, mais nous avons résolu cela en faisant le produit scalaire de delta et la dérivée de ***softmax*()**.

On pourra ainsi , améliorer la précision en ajoutant **:**

1. Taux d'apprentissage décroissant. Par exemple, nous commençons avec 0.1 et nous le réduisons après chaque 50 époques.
2. *Early stopping*, meilleure solution pour éviter le sur-apprentissage. nous pouvons considérer comme négligeable la réduction de l'erreur après l'époque 160-170.
3. Choisir la meilleure partie de la trame de données pour l'entraînement à l'aide de la fonction "*Cross Validation"* qui peut donner comme précisions sur 5 trames :[0.96, 0.98, 0.98, 0.97, 0.97].