Rapport Mini Projet
Machine Learning

Alexis Araujo

Introduction

L’objectif de ce projet était de modéliser le processus d'admission des étudiants en se basant sur leurs performances en mathématiques et en français. Je devais créer un dataset avec ces données puis utiliser un algorithme d’apprentissage supervisé : perceptron, un modèle capable de prédire si un étudiant serait admis ou non en fonction des critères que je définis.

Création du Dataset

J’ai débuté par la création d’un ensemble de données simulées représentant les résultats de 1000 étudiants (paramètre m). Chaque étudiant était caractérisé par deux notes : en français et en mathématiques qui sont des caractéristiques (labels). Les notes ont été générées de manière aléatoire pour chaque étudiant, avec des valeurs allant de 0 à 20 pour chacune des deux caractéristiques.

Ces notes aléatoires ont ensuite été soumises à une pondération. J’ai mis des poids particuliers pour représenter des coefficients à chaque matière pour l’importance d’une matière à une autre dans la décision d'admission. Dans cet exemple, j'ai choisi un poids de 3 pour les notes de mathématiques et un poids de 2 pour les notes de français, avec une plus grande importance accordée aux matières scientifiques dans le processus d'admission.

Une fois ces données pondérées obtenues, j'ai calculé la moyenne pondérée pour chaque étudiant en liant les notes de chaque matière avec leurs poids respectifs. Cette moyenne pondérée a ensuite été divisée par la somme des poids, permettant pour obtenir une évaluation plus équilibrée des performances des étudiants de chaques différentes matières.

Pour déterminer les résultats d'admission, j'ai établi un critère pour qu’un élève soit admis ou non, à une moyenne pondérée de 10. Si la moyenne pondérée d'un étudiant est supérieure ou égale à cette valeur, l'étudiant est décrit comme "admis" avec une valeur de 1, sinon il est "non admis" avec une valeur de 0.

Pour visualiser ces données, j'ai affiché la matrice des notes de français et de mathématiques, où chaque ligne représente les notes d’un élève et chaque colonne représente les notes de chaque matière. Ensuite, j'ai affiché le tableau de booléens, où chaque nombre représente un étudiant ; un 1 signifie qu'il est admis et un 0 qu'il est non admis. Cette représentation m'a permis d'avoir une vue sur les résultats d'admission en fonction des notes des étudiants.

Apprentissage avec Perceptron

Après avoir défini les résultats d'admission et obtenu les données de caractéristiques et de résultats, j'ai effectué un apprentissage avec Perceptron.

J'ai commencé par modifier la forme des étiquettes d'admission “y” pour m'assurer de leur compatibilité avec le modèle. En les “reshapant” à (m, 1), où m représente le nombre d'étudiants, j'ai organisé ces étiquettes pour qu'elles soient de même dimension et mieux utilisées par l’apprentissage.

Ensuite, pour obtenir une première vue sur la dispersion des données, j'ai affiché un diagramme en utilisant plt.scatter(). J'ai représenté graphiquement les données des étudiants en utilisant les notes en mathématiques sur l'axe des abscisses et les notes en français sur l'axe des ordonnées. Les points ont été colorés en fonction des résultats (y), cela permet d’observer la séparation entre les étudiants admis et non admis en fonction des moyennes pondérées de leurs notes. Les points verts sont les étudiants pour lesquels la moyenne pondérée des deux caractéristiques dépassent 10 et sont donc admis et les points bleus sont les étudiants pour lesquels la moyenne pondérée des deux caractéristiques ne dépassent pas 10 et sont donc non admis.

Par la suite, j'ai défini les différentes fonctions nécessaires pour l'implémentation du Perceptron :

- initialisation(): La fonction est utilisée pour initialiser aléatoirement les poids (W) et le biais (b) du modèle.
- model(): Créée pour calculer les prédictions du modèle en utilisant la fonction sigmoïde.
- logLoss(): Cette fonction calcule la perte logistique pour évaluer les performances du modèle.
- gradients(): Cette fonction calcul les gradients pour les poids et le biais.
- update(): La fonction permet de mettre à jour les poids et le biais avec la descente de gradient.

La fonction principale perceptron() calcul l'entraînement du modèle sur un nombre fixé d'itérations à 5000. Pendant chaque itération, la prédiction est calculée, les gradients aussi, la mise à jour les poids et le biais est faite, mais aussi enregistre la perte, et évalue les prédictions par rapport aux étiquettes réelles pour obtenir la précision du modèle.

Par la suite, j’ai fait appel à cette fonction perceptron qui m’a permis également de voir le score de la prédiction du modèle qui tourne autour de 97%, mais également cet appel a permis l’affichage de la fonction de coût au fil des itérations. Cette fonction de coût diminue fortement lors des premières itérations, cela est dû à la vitesse de descente de chaque itération que j’ai fixé à 0.05. Malgré

ça on peut observer une petite montée du coût de la prédiction, comme si l’apprentissage du modèle à reconnu une erreur à un moment. Cependant la fonction d’erreur diminue après cette petite montée.

Pour mieux comprendre comment le modèle sépare la classe des admissions à celle des non admis, j'ai tracé la frontière de décision. Cette frontière a été calculée à partir des poids et du biais obtenus par le modèle, et elle a été ajoutée au nuage de points représentant les données des étudiants, cela permet une visualisation claire de la séparation attendue entre les deux classes. Mais la séparation n’est pas quasi parfaite, elle montre quelques légers écarts, cela est dû à l'imperfection de l'apprentissage du modèle. Pour améliorer cette séparation, il est nécessaire d'augmenter le nombre d'itérations pour préciser l'apprentissage.

Pour terminer, j'ai simulé la création d'un nouvel étudiant en définissant une note de mathématiques à 2 et une note de français à 1. En utilisant cette "nouvelle plante" avec la fonction de prédiction, j'ai appelé les poids pondérés et le biais. La fonction de prédiction a alors fait appel à la fonction modèle pour calculer les prédictions et déterminer si l'étudiant était admis ou non. Une sortie de "True” signifie que l'étudiant est admis, tandis qu'une sortie de "False" indique le contraire. De plus, elle a fourni le pourcentage de chance que l'étudiant soit admis ou non, offrant ainsi un aperçu des probabilités associées à cette prédiction.

Conclusion

Ce mini projet m'a permis de mettre en pratique l'apprentissage supervisé, en particulier l'utilisation du Perceptron pour la classification. J'ai pu observer comment l'algorithme ajustait ses paramètres pour mieux séparer les étudiants admis de ceux non admis en fonction de leurs notes et au fil des itérations. Cette expérience a renforcé ma compréhension sur l'apprentissage automatique.