Rapport TP 1 réseaux de neurones

Maxime FROISSANT
Guillaume MEDARD
Younes KARMOUCHE
Gabriel MONCZEWSKI

Ce rapport à été fait en collaboration avec Maxime FROISSANT, Younes KARMOUCHE, Guillaume MEDARD et Gabriel MONCZEWSKI.

1. Introduction

Le perceptron est un algorithme d'apprentissage pour la classification binaire. Il s'agit d'un réseau de neurones simplifié, avec une couche d'entrée, des poids, un biais et une fonction d'activation (souvent la fonction de Heaviside), permettant de prendre une décision entre deux classes.

2. Objectifs du TP

1. Comprendre le fonctionnement mathématique et algorithmique du perceptron.
2. Implémenter l'algorithme d'apprentissage.
3. Appliquer le modèle à des jeux de données simples.
4. Observer les limites du perceptron.

3. Questions d'analyse

Exercice 2

Pourquoi la fonction de Heaviside pose-t-elle problème pour l'apprentissage par gradient ?

Parce qu'elle n'est pas dérivable (discontinue) et sa dérivée est nulle partout sauf en un point, ce qui empêche la rétropropagation du gradient.

Dans quels cas utiliser sigmoid vs tanh ?

Sigmoid : en sortie de réseau pour une classification binaire (valeurs entre 0 et 1).
Tanh : en couches cachées car centrée en 0 (valeurs entre -1 et 1), ce qui accélère la convergence par rapport à sigmoid.

Pourquoi ReLU est-elle si populaire dans les réseaux profonds ?

Elle est simple, rapide à calculer, évite le problème du vanishing gradient (pour x > 0), et

favorise un apprentissage plus efficace dans les réseaux profonds.

Quel est l'avantage du Leaky ReLU ?

Elle résout le problème des neurones morts de ReLU en gardant une petite pente non nulle pour les valeurs négatives, assurant un flux de gradient même quand $x < 0$.

Exercice 3

Que se passe-t-il si η est trop grand ?

L'apprentissage devient instable : les poids peuvent osciller ou diverger sans jamais converger vers une solution.

Et s'il est trop petit ?

L'apprentissage est trop lent : le modèle mettra beaucoup de temps à converger, voire stagnera dans un minimum local.

Existe-t-il une valeur idéale de η ?

Non, il n'existe pas de valeur universelle idéale. Elle dépend du problème, des données, de l'architecture du modèle et de la fonction de coût.

Peut-on faire varier η au cours du temps ?

Oui, c'est même couramment utilisé pour améliorer la convergence.

Quelle stratégie pouvez-vous imaginer ?

Une stratégie classique est le learning rate decay (décroissance du taux d'apprentissage), par exemple :

$\eta_t = \eta_0 / (1 + k \cdot t)$ (avec t = époque, k = constante)

Ou une décroissance exponentielle ou par palier. Cela permet de commencer avec un grand η pour apprendre vite, puis de le réduire progressivement pour affiner les poids.

Exercice 8

Que se passe-t-il lorsque η est très petit ?

Lorsque le taux d'apprentissage (η) est très petit (ex. : 0.0001 ou 0.001), le perceptron effectue des modifications infimes des poids à chaque erreur. Cela entraîne une convergence très lente, un risque de stagnation si l'amélioration est trop progressive et une grande stabilité, mais peu d'efficacité à court terme.

Que se passe-t-il lorsque η est trop grand ?

Avec un η trop élevé (ex. : 1.0, 3.0, 10.0), les ajustements deviennent trop brusques, ce qui provoque des sauts excessifs dans l'espace des poids, une oscillation de l'erreur, voire une divergence complète et pas d'apprentissage stable : le perceptron "oublie" d'une époque à l'autre.

Existe-t-il un η optimal dans votre cas ?

Oui, généralement une valeur intermédiaire (comme 0.01 ou 0.1) permet une convergence relativement rapide et une stabilité suffisante pour éviter les oscillations. Le η optimal dépend du problème et nécessite souvent une recherche.

Comment la structure des données interagit-elle avec η ?

La structure des données influence fortement la sensibilité à η, si les données sont bien séparables et peu bruitées un η plus grand peut convenir, si elles sont bruitées ou dispersées il faut un η plus petit pour éviter de trop réagir à des points aberrants et dans les cas complexes (ex. données mal séparables ou bruit significatif), un mauvais η peut empêcher totalement la convergence.

Conclusion

Ces observations confirment les réponses de l'Exercice 3 :

- η doit être choisi avec soin, en fonction des données et de la tâche.
- Une stratégie adaptative ou un décroissement progressif de η peut améliorer la robustesse de l'apprentissage.

Exercice 9

Que se passe-t-il si plusieurs perceptrons prédisent positivement pour le même exemple ?

Plusieurs perceptrons peuvent effectivement activer (prédire « positif ») pour le même exemple, créant une zone de conflit, dans ce cas, la stratégie habituelle consiste à :

Choisir le perceptron avec le score le plus élevé (avant activation), i.e. le plus « confiant ». Cela revient à prendre le maximum des sorties pondérées (sommation) avant la fonction d'activation.

Avantage : cela permet de trancher même en cas de prédictions multiples.
Limite : cela ne garantit pas que la prédiction soit fiable si les scores sont proches.

Comment gérer le cas où aucun perceptron ne prédit positivement ?

Cela peut se produire si tous les scores sont négatifs (sous Heaviside), ou très faibles. On peut alors :

Utiliser les scores bruts (avant activation) et prendre celui avec la plus grande valeur, même si elle est négative.
Alternativement, considérer un seuil adaptatif ou ne pas classer (ex. : retour "inconnu" ou "incertain").

Bonne pratique : ne pas s'appuyer uniquement sur la sortie binaire, mais exploiter le score de confiance.

Comment l'approche "Un contre Tous" gère-t-elle le déséquilibre naturel qu'elle crée ?

Chaque perceptron apprend à reconnaître sa propre classe (positive) contre toutes les autres (négatives). Cela crée un déséquilibre : la classe positive est sous-représentée par rapport à la somme des autres.

Conséquences :
Risque que le perceptron favorise la classe majoritaire (négative), apprenant à prédire « non » par défaut. Cela peut nuire à la sensibilité du modèle à sa propre classe.

Stratégies d'atténuation :
Rééchantillonnage (oversampling de la classe positive ou undersampling des négatives).
Pondération des erreurs (pénaliser plus fortement les erreurs sur la classe minoritaire).
Choix judicieux de η pour éviter une trop forte inertie d'apprentissage.

Réponses aux questions sur le perceptron

1. Convergence : Le perceptron est garantit de converger si les données sont linéairement séparables et si le taux d'apprentissage est positif mais suffisamment petit. Le théorème de convergence du perceptron assure qu'il trouve une frontière de séparation en un nombre finis d'itération dans ce cas.
2. Initialisation : L'initialisation des poids n'influence pas la solution finale si les données sont linéairement séparables, car le perceptron convergera vers une solution (pas forcément unique). Cependant, elle peut affecter la vitesse de convergence.
3. Taux d'apprentissage : Le taux d'apprentissage optimal dépend du problème. Un taux trop élevé cause des oscillations, un taux trop faible ralentit la convergence. Une approche courante est de choisir un taux constant petit (ex. 0,01 à 0,1) ou d'utiliser un taux adaptatif (ex. décroissance progressive).
4. Généralisation : La capacité de généralisation du perceptron est évaluée via un ensemble de tests distinct de l'ensemble d'entraînement. On mesure l'erreur de classification (ou précision) sur cet ensemble pour estimer la performance sur des données non vues.
5. XOR Revisité : Le perceptron simple ne peut pas résoudre le problème XOR (non linéairement séparable). Solutions :
    - Utiliser un perceptron multicouche (MLP) avec une couche cachée.
    - Transformer les données (ex. ajouter des caractéristiques polynomiales).
    - Combiner plusieurs perceptrons (ex. réseau avec non-linéarités).
6. Données bruitées : Le perceptron peut être sensible au bruit, entraînant une convergence instable ou une frontière de séparation sous-optimale. Une

régularisation (ex. marge maximale) ou un prétraitement (réduction du bruit) peut aider.

7. Classes déséquilibrées : Si une classe est très minoritaire, le percepteur peut biaiser vers la classe majoritaire, ignorant la minoritaire. Solution : rééchantillonnage (sur-échantillonnage de la classe minoritaire ou sous-échantillonnage de la majoritaire) ou poids de classe dans la fonction de coût.
8. Normalisation : Oui, il est recommandé de normaliser les données (ex. mise à l'échelle [0,1] ou standardisation) pour éviter que des caractéristiques à grande échelle dominent l'apprentissage et pour accélérer la convergence.

Conclusion : Synthèse des apprentissages

Ce TP sur le perceptron nous a permis de comprendre son fonctionnement pour la classification binaire, ses limites et ses besoins. Le perceptron converge pour des données linéairement séparables avec un taux d'apprentissage bien calibré, mais échoue sur des problèmes non linéaires comme XOR, nécessitant un MLP avec au moins deux neurones cachés. L'initialisation des poids influence la vitesse de convergence, et des fonctions comme ReLU ou Leaky ReLU sont préférables à Heaviside pour l'apprentissage par gradient. Normalisation, gestion des classes déséquilibrées et traitement du bruit sont essentiels pour une bonne généralisation. Ce travail pose une base solide pour explorer des modèles plus complexes.

ANNEXES VISUALISATION/GRAPHES