2025

Rapport : Projet Réseau de Neurones

UNIVERSITé DE POITIERS

nicolas bouillon, ZOE ARTHAPIGNET, MELIE BORDAGE, ZACK HEBERT

Groupe 6

Table des matières

[1. Contexte 2](#__RefHeading___Toc2325_2729266023)

[2. Chronologie 2](#__RefHeading___Toc2327_2729266023)

[3. Répartition des tâches 2](#__RefHeading___Toc2329_2729266023)

[4. Choix de conception 3](#__RefHeading___Toc2331_2729266023)

[Structure de données (types.h) 3](#__RefHeading___Toc2333_2729266023)

[Structure d’un point d’une spirale 3](#__RefHeading___Toc2335_2729266023)

[Génération du jeu de données (neurone.c) 3](#__RefHeading___Toc2337_2729266023)

[Création et sauvegarde du réseau de neurones (neurone.c) 3](#__RefHeading___Toc2339_2729266023)

[Perceptron multicouche 4](#__RefHeading___Toc2341_2729266023)

[Architecture du réseau de neurone (neurone.c) 4](#__RefHeading___Toc2343_2729266023)

[Apprentissage du réseau de neurone(neurone.c) 4](#__RefHeading___Toc2345_2729266023)

[Affichage à l’aide de SDL2 4](#__RefHeading___Toc2347_2729266023)

[5. Analyse des performances 4](#__RefHeading___Toc2349_2729266023)

[6. Résultats 5](#__RefHeading___Toc2351_2729266023)

[Bibliographie 5](#__RefHeading___Toc2353_2729266023)

# 1. Contexte

Le projet consiste à concevoir un réseau de neurones de type perceptron multicouche, capable de distinguer deux populations de points (rouges et bleus) formant chacun une spirale. L’objectif est d’entraîner ce réseau afin qu’il puisse, à partir de coordonnées (x, y), déterminer à quelle classe (rouge ou bleu) le point appartient. Nous utilisons pour cela la fonction de transfert tangente hyperbolique et l’algorithme de rétropropagation du gradient pour ajuster les poids des neurones dans notre réseau.

# 2. Chronologie

1. **Mise en place du réseau** : création d’une structure NeuralNetwork avec le nombre de couches, la taille de chaque couche, les poids, ainsi que les sorties et les deltas associés.
2. **Expérimentations** : ajustement de paramètres (nombre de couches, neurones par couche, taux d’apprentissage, nombre d’époques, etc.).
3. **Optimisation** : encapsulation du code pour faciliter la réutilisation et la configuration, puis séries de test pour améliorer la précision et réduire le temps de calcul.
4. **Documentation et finalisation** : génération de la documentation et consolidation du code après avoir validé les performances.
5. **Présentation et rapport** : préparation des supports (diapositives, rapport) et synthèse des résultats obtenus.

# 3. Répartition des tâches

* **Code et réglage des paramètres** : Zack et Nicolas
* **Benchmark** (tests de performance et comparaison) : Zoé
* **Documentation** : Mélie
* **Présentation et rédaction du rapport** : Nicolas, Mélie et Zoé

# 4. Choix de conception

Le perceptron a été retenu pour sa simplicité et son efficacité en classification binaire (ici deux classes : rouge et bleu). Son entraînement rapide répondait à nos contraintes de temps et de ressources (exécution sur nos PC personnels).

## Structure de données (types.h)

Structure du réseau de neurones (NeuralNetwork) :

* nbLayers : nombre total de couches (incluant l’entrée et la sortie).
* layerSizes : tableau spécifiant le nombre de neurones par couche.
* weights : tableau 3D contenant les poids synaptiques entre les couches successives.
* outputs : tableau 2D stockant les sorties calculées de chaque neurone.
* delta : tableau 2D stockant les erreurs nécessaires à la rétropropagation.

### Structure d’un point d’une spirale

* x et y : Les coordonnées du point
* target : tableau 2D stockant la couleur du point

## Génération du jeu de données (neurone.c)

Dans ce projet, le jeu de données sont deux spirales symétriques, pour obtenir des spirales symétrique ont utilise la formule des spirales d’Archimède. (Archimède, s.d.) Pour calculer le step entre 2 points de la spirale, on utilise . La rotation est , car c’est la distance entre 2 spires de la spirale d’Archimède (Paloma & LEAUVA , 2010). est le facteur de grandeur de notre spirales proportionnel à la taille de fenêtre du rendu et du nombre de points dans la spirale.

## Création et sauvegarde du réseau de neurones (neurone.c)

Pour créer et ensuite sauvegarder le réseau de neurones, il faut alloué de la mémoire pour chaque variable définie dans notre structure du réseau de neurones. C’est ce que fait la fonction CreateNetwork, la fonction freeNetwork libère la mémoire. La fonction saveNetwork permet de conserver dans un fichier txt les données du réseau de neurones, loadNetwork permet le charger le dernier réseau sauvegardé .

## Perceptron multicouche

La phase principale du projet consiste à utiliser le perceptron et à l’entraîner, pour cette partie, on s’est aidé du sujet du projet qui nous donnait les éléments fondamentaux du projet donc la fonction de transfert et d’un document sur le (Réseaux de neurones) du département mathématique de l’Université de Toulouse.

### Architecture du réseau de neurone (neurone.c)

Le perceptron consiste à lier les couches de neurones entre elles, pour cela, on fait une passe-avant, la fonction **ForwardPass**. La couche d’entrée du réseau de neurones n’est pas modifiée. Pour les couches suivantes, on utilise la fonction de transfert pour recalculer les poids de chaque neurone en fonction des neurones de la couche précédente à laquelle ils sont liés.

### Apprentissage du réseau de neurone(neurone.c)

Pour l’apprentissage du réseau de neurones, après avoir fait la passe-avant, on fait la retropropagation, c’est la fonction **Backpropagation** on utilise l’algorithme fournit dans le sujet, elle nous permet de calculer le gradient et le delta.

Ensuite, on choisit d’entraîner sur un grand nombre d’itération, c’est **TrainNetwork** définie par NB\_EPOCHS et aussi un maxDelta qu’on recalcule à chaque itération, qui nous doit être inférieure au seuil d’erreur MIN\_ERROR.

## Affichage à l’aide de SDL2

Pour l’affichage en sdl de la fenêtre du rendu et de la texture, on s’est aidé (utiliser-la-sdl-en-langage-c.pdf, 2020) pour les principes de bases du SDL.

# 5. Analyse des performances

Plusieurs configurations ont été testées : de 4 à 6 couches, avec entre 5 et 80 neurones par couche, pour mesurer le compromis entre précision et temps de calcul. Finalement, la structure de réseau la plus équilibrée en termes de performances et de rapidité s’est avérée être un réseau de 7 couches avec la configuration [2, 48, 8, 24, 8, 32, 2].

# 6. Résultats

Le réseau ainsi entraîné distingue efficacement les spirales rouge et bleue, avec un temps d’apprentissage moyen compris entre 20 et 30 secondes. La décision du réseau est précise : elle suit bien les deux spirales, même éloignées du centre (où la densité de points est moins élevée et plus étendue). Les tests ont confirmé la robustesse du réseau sur différentes tailles d’échantillons et un large éventail de paramètres.

# Bibliographie

Archimède. (s.d.). *Spirale d'Archimède*. Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Spirale\_d%27Archim%C3%A8de

Paloma, C.-O., & LEAUVA , D. (2010). *Quadature de la spirale d'archimède.* Récupéré sur Université Paris-Saclay: https://www.imo.universite-paris-saclay.fr/~daniel.perrin/Projet-geometrie/Daena-Paloma.pdf

*Réseaux de neurones.* (s.d.). Récupéré sur math.univ-toulouse: https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

*utiliser-la-sdl-en-langage-c.pdf.* (2020). Récupéré sur zestedesavoir: https://zestedesavoir.com/tutoriels/pdf/1014/utiliser-la-sdl-en-langage-c.pdf