alphai

L'ÉDUCATION À L'INTELLIGENCE ARTIFICIELLE

TP Détection d'intrus

Les mathématiques des réseaux de neurones

Introduction

Dans cette activité, nous allons découvrir le fonctionnement d'un **réseau de neurones** et s'intéresser aux différents objets mathématiques qui le composent :

1. L'opération linéaire réalisée par un neurone
2. Le biais neuronal
3. Les couches de neurones intermédaires
4. La fonction d'activation

Au cours de cette activité, nous nous baserons sur l'exemple d'un robot AlphAI utilisé comme *détecteur d'intrusion*. Nous allons donc découvrir ces différents aspects des réseaux de neurones en réalisant et en améliorant ce mode de fonctionnement. Les réseaux de neurones que nous allons utiliser dans cette activité sont très simples et ne contiennent que quelques neurones, mais ils partagent le même fonctionnement que les réseaux les plus complexes.

1 L'opération linéaire réalisée par un neurone

1.1 Configuration du logiciel

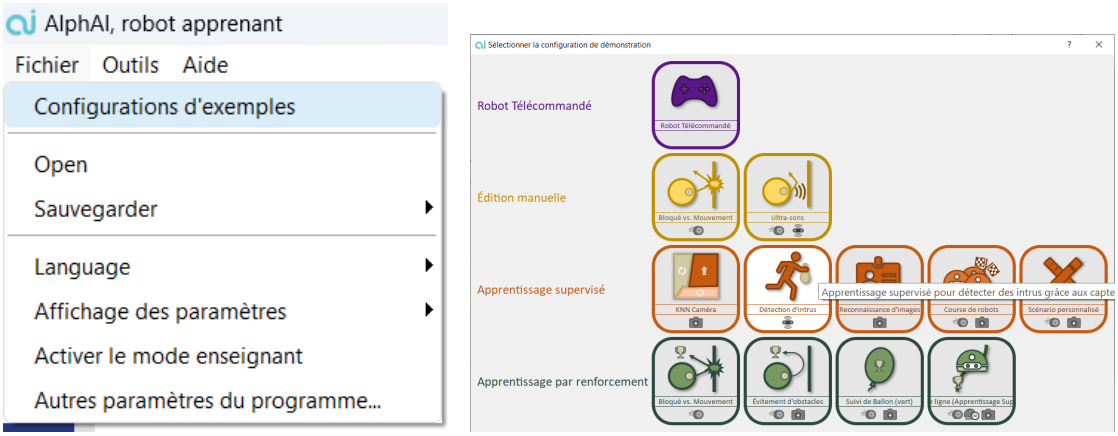

Figure 1 – Comment charger les paramètres d'exemple : détection d'intrus

1. Lancez le logiciel AlphAI et connectez-vous à un robot.
2. Placez le robot face à une surface plane (par exemple un mur ou une tour d'ordinateur) à environ 30 cm.
3. Chargez les paramètres d'exemple correspondant au scénario **Apprentissage supervisé - détection d'intrus** (voir figure 1).

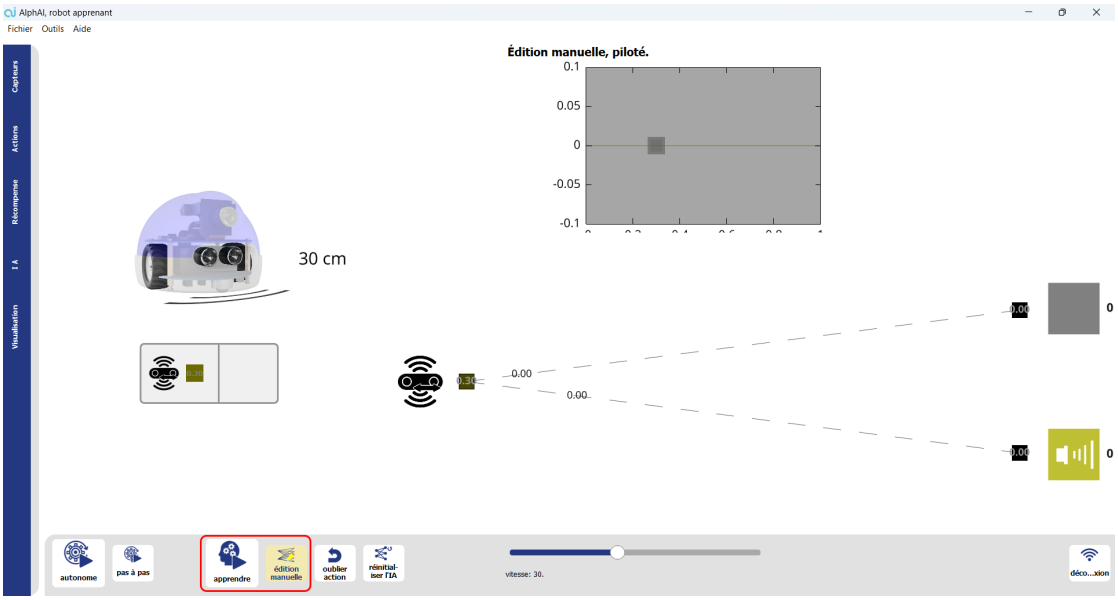

FIGURE 2 – Écran principal après configuration

4. Dans la barre inférieure, activez le bouton *édition manuelle* afin de pouvoir modifier vous-même les valeurs des connexions du réseau. Puis, cliquez sur le bouton *réinitialiser l'IA* afin de fixer les valeur des connexions à 0.
5. Si la configuration est effectuée correctement, votre écran doit ressembler à l'image de la figure 2.

1.2 EXPÉRIMENTATION

1. Le réseau de neurones en bas de l'écran comporte un unique **neurone d'entrée** (à gauche) qui correspond au capteur à ultra-sons du robot. Ce capteur mesure la distance qui sépare votre robot de l'obstacle qui lui fait face, en centimètres. *Quel est le lien entre cette distance, qui apparaît à côté de l'image du robot, et la* ***valeur d'activation*** *du neurone d'entrée, qui apparaît dans ce neurone ?*
2. Nous allons maintenant nous intéresser aux niveaux d'activation des deux **neurones de sortie** (à droite). Cliquez sur une des deux connexions du réseau et faites varier la valeur de cette connexion, en utilisant le clavier ou la molette de votre souris. Observez la valeur d'activation du neurone de sortie correspondant. *Quelle opération simple permet d'obtenir cette valeur* `y` *en fonction du niveau d'activation* `x` *du neurone d'entrée et du poids* `w` *de la connexion ?*
3. Modifiez de même le poids de l'autre connexion, puis activez le mode *autonome*. Faites varier les valeurs des connexions et observez quelle action est choisie par le robot. *En déduire la méthode utilisée par le robot pour choisir l'action à effectuer.*

À retenir :

- Un réseau de neurones est un ensemble de neurones organisé par **couches**. Chaque couche peut comporter un ou plusieurs neurones.
- Dans notre représentation, un neurone est considéré comme un nœud du réseau, contenant une valeur appelée **niveau d'activation**.
- Les neurones sont reliés entre eux par des connexions dont les valeurs sont appelées **poids**.
- Dans des réseaux plus complexes où des neurones ont plusieurs connexions entrantes, le niveau d'activation de chaque neurone est égal à la somme des niveaux d'activation des neurones de la couche précédente, pondérés par les poids des connexions correspondantes (comme illustré sur la figure 3) :

$$y = w_1 x_1 + \cdots + w_n x_n$$

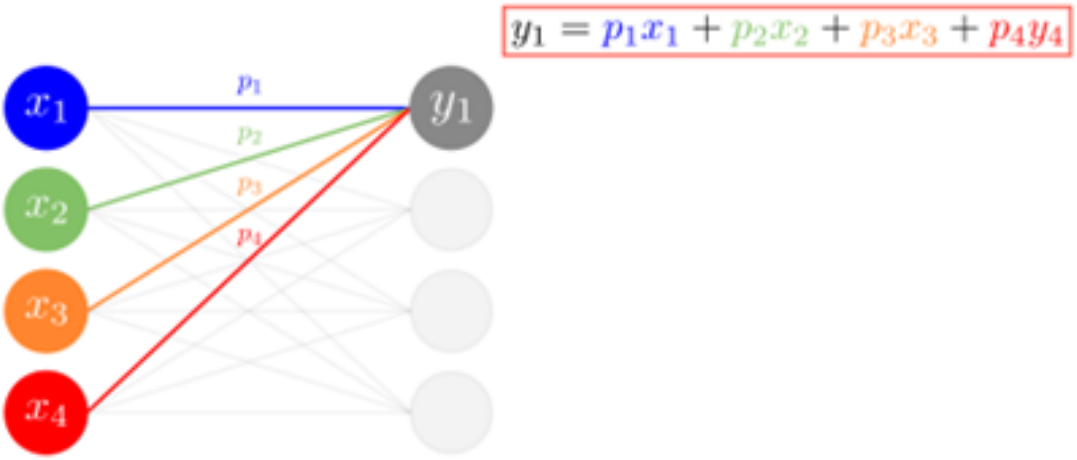

FIGURE 3 – Opération linéraire d'un neurone

1.3 AJOUT DE L'INTRUS

1. Replacez le robot face à une surface plane à environ 30 cm. La présence d'un intrus est symbolisée par un objet plat (par exemple un livre ou un téléphone) inséré entre le robot et le mur, de sorte que la distance mesurée par le robot devient inférieure à 30 cm. On voudrait que le robot émette un son lorsqu'il détecte un intrus (c'est-à-dire lorsqu'il mesure une distance inférieure à 30 cm) et cesse de bipper lorsque la distance est normale (environ 30 cm). *Est-il possible d'obtenir ce comportement en modifiant les valeurs des connexions ?*
2. Sur votre écran, au-dessus du réseau de neurones, se trouve la représentation de l'**espace d'états** de l'intelligence artificielle. Fixer le poids de la connexion correspondant à l'action *bip* à 1 et le poids de l'autre connexion à 2. *Que voit-on sur ce graphique ?*

3. Chaque droite correspond à la fonction calculée par un neurone de sortie. *À quelles valeurs correspondent l'abscisse et l'ordonnée de ce graphique ? Identifier à quel neurone correspond chaque droite. En déduire pourquoi l'action choisie par le robot est toujours la même.*
4. *Comment pourrait-on modifier ces droites pour obtenir le comportement souhaité du robot ?*

2 Le rôle du biais neuronal

Attention : pour accéder aux paramètres avancés décrits dans cette partie, rendez-vous dans le menu **Fichier > Affichage des paramètres** et sélectionnez le mode **Avancé**.

2.1 Choix des valeurs des paramètres du réseau

1. Dans la partie précédente, nous avons vu que chaque neurone de sortie réalise une fonction linéaire, qui est donc représentée par une droite passant par l'origine. Pour réaliser la détection d'intrus, nous avons besoin de fonctions affines, afin de pouvoir contrôler la position du point d'intersection des droites correspondantes. *Quel paramètre distingue une fonction affine d'une fonction linéaire ?*
2. Dans le contexte des réseaux de neurones, ce paramètre s'appelle le **biais neuronal**. Pour l'activer dans AlphAI, activez l'option *biais neuronal* dans l'onglet *IA*. Ces valeurs de biais apparaissent maintenant à côté de chaque neurone de sortie. Elles peuvent être modifiées à l'instar des connexions en plaçant votre souris sur le neurone correspondant. *Quelle est l'expression du niveau d'activation* `y` *d'un neurone de sortie en fonction du niveau d'activation* `x` *du neurone d'entrée, du poids* `w` *de sa connexion et de son biais* `b` *?*
3. *À quelle valeur d'abscisse souhaite-t-on positionner l'intersection entre les droites de nos deux neurones ?*
4. *Déterminer des valeurs des paramètres* w_1, w_2 *(les poids des connexions 1 et 2) et* b_1, b_2 *(les biais neuronaux) qui permettent d'obtenir le comportement souhaité.*
5. Configurez le réseau avec ces valeurs et vérifiez que le comportement obtenu permet bien de détecter les intrusions.

2.2 Ajout d'un intrus "malin"

1. Placez maintenant l'intrus (surface plane) entre le robot et le mur, mais en formant un angle de 45° avec celui-ci. Observez la valeur affichée par le

capteur à ultra-sons. *Comment expliquer cette valeur ?*

2. *Ce nouveau type d'intrus est-il détecté par votre robot ?* Passez en mode autonome pour le vérifier.
3. Jusqu'ici nous avions 2 **zones de décision** (en couleur dans AlphAI). *Combien en faudrait-il pour détecter ce nouveau type d'intrusion ?*

3 Les couches de neurones intermédiaires

3.1 Introduction

À retenir :

Les couches de neurones intermédiaires se trouvent entre la couche d'entrée et la couche de sortie. Les neurones de ces couches se comportent *presque* comme nous avons vu jusqu'à présent : chacun calcule son niveau d'activation comme étant une combinaison linéaire des niveaux d'activation des neurones de la couche précédente, plus le biais :

$$y = w_1 x_1 + ... + w_n x_n + b$$

Cependant, sans ajout supplémentaire, ces couches intermédiaires n'auraient pas d'intérêt, puisque les fonctions calculées par les neurones de sortie seraient alors toujours des fonctions affines, et ne permettraient pas d'obtenir plus de **zones de décision**. En effet, (voir cours d'algèbre linéaire) la composition de deux fonctions linéaires est toujours linéaire !

Il faut donc un ingrédient supplémentaire : c'est la **fonction d'activation** ! Il s'agit d'une fonction f *non linéaire* qui permet au réseau de neurones de calculer des fonctions plus complexes et donc d'augmenter le nombre de **zones de décision** de la couche de sortie. Avec cette fonction d'activation, le calcul réalisé par chaque neurone est donc maintenant non linéaire :

$$y = f(w_1 x_1 + ... + w_n x_n + b)$$

Dans les questions ci-dessous, nous allons utiliser comme fonction d'activation la fonction de Heaviside, dont la valeur est 1 lorsque son entrée est positive, et 0 sinon.

3.2 Mise en place de la couche intermédiaire

1. Dans **AlphAI**, le nombre et la taille des couches intermédiaires sont contrôlés par le paramètres *couches de neurones intermédiaires* dans l'onglet *IA*. Entrez la valeur 2 pour ce paramètre. On obtient une couche de neurones

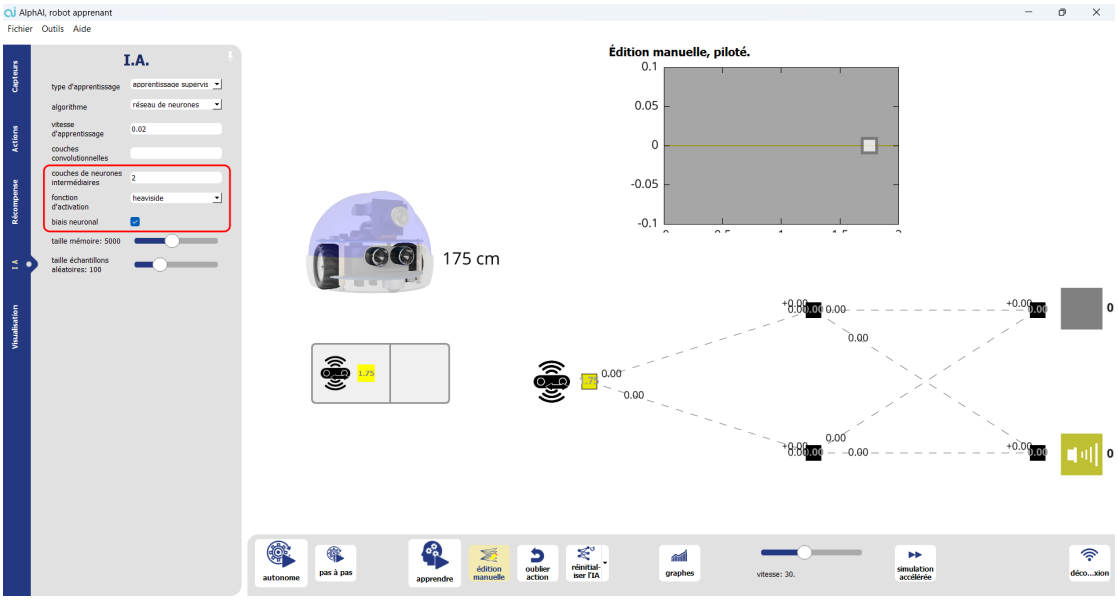

FIGURE 4 – Paramétrage de la couche intermédiaire

intemédiaire contenant 2 neurones, comme dans la figure 4. Sélectionnez également la fonction d'activation *heaviside*.

2. Pour le premier neurone de la couche intermédiaire, on souhaite qu'il s'active uniquement lorsque la distance mesurée par le capteur est trop faible (par exemple inférieure à 20 cm). *Déterminez des valeurs pour la connexion et le biais de ce neurone qui permettent d'obtenir ce résultat.* **Astuce :** en reliant chaque neurone intermédiaire à un neurone de sortie avec une connexion de poids 1 (et biais 0), on peut visualiser le niveau d'activation des neurones intermédiaires sur le graphe d'états.
3. Pour le second neurone de la couche intermédiaire, on souhaite qu'il s'active uniquement lorsque la distance mesurée par le capteur est trop grande (par exemple supérieure à 40 cm). *Déterminez des valeurs pour la connexion et le biais de ce neurone qui permettent d'obtenir ce résultat.*
4. Pour la couche de sortie, on souhaite que le robot bippe lorsque la valeur mesurée par le capteur est trop grande ou trop faible, et ne fasse rien si la distance est "normale". *Déterminez des valeurs pour les connexions et les biais de la partie droite du réseau afin d'obtenir ce résultat.*
5. Tester les valeurs choisies en vérifiant que le robot parvient à détecter à la fois les intrus "standard" et les intrus "malins".
6. Observez les courbes et les zones de décision affichées dans l'espace d'états au dessus du réseau. Comment se manifestent les choix des paramètres et de la fonction d'activation ?

3.3 LES "VRAIS" RÉSEAUX DE NEURONES

À retenir :

Les réseaux de neurones que nous avons étudiés dans cette activité sont simplifiés afin de bien comprendre les bases de leur fonctionnement. Quelles sont les différences avec les "vrais" réseaux de neurones (par exemple ceux présents dans votre téléphone) ?

- **Le nombre de neurones de la couche d'entrée**. Pour un réseau de neurones faisant de la classification d'images, le nombre de neurones de la couche d'entrée est très grand. Il est égal au nombre de pixels de l'image, multiplié par 3 pour les images en couleurs (pour les canaux rouge, vert, bleu). Pour traiter des images de 5 mégapixels, il faut donc 15 millions de neurones sur la couche d'entrée ! (ou réduire la définition de l'image)
- **Le nombre et la taille des couches intermédiaires**. La taille des couches intermédiaires est le plus souvent proportionnelle à la taille de la couche d'entrée. Le nombre de couches intermédiaires peut être lui aussi très grand : on parle alors d'apprentissage profond (*deep learning*).
- **Le choix de la fonction d'activation**. Parmi les fonctions d'activation les plus utilisées, on peut citer la fonction sigmoïde et la fonction ReLu.
- Enfin, certains réseaux ont des **architectures plus complexes**. Cela signifie chaque neurone n'est pas nécessairement connecté à tous les neurones de la couche précédente, mais que les neurones sont connectés entre eux selon un schéma plus complexe. C'est le cas des réseaux convolutifs utilisés pour la reconnaissance d'images, par exemple.

4 Apprentissage automatique

4.1 Introduction et paramétrage

Maintenant que vous comprenez le fonctionnement interne d'un réseau de neurones et l'effet de ses paramètres, il doit vous sembler clair qu'il serait très fastidieux de choisir manuellement le poids de chaque connexion et la valeur de chaque biais, surtout dans le cas de réseaux de plus grande taille. Il est donc nécessaire d'utiliser un algorithme pour décider des valeurs de ces paramètres. C'est ce qu'on appelle un **algorithme d'apprentissage**.

Le rôle d'un algorithme d'apprentissage est de déterminer des valeurs pour les connexions et les biais d'un réseau, qui permettent d'obtenir le comportement souhaité. La plupart des algorithmes d'apprentissage ont besoin de beaucoup de données (et de beaucoup de temps) pour accomplir leur tâche, c'est pour cette raison que l'apprentissage automatique est fortement lié au développement du Big Data.

AlphAI permet de réaliser des apprentissages automatiques avec un petit nombre de données à l'aide d'un algorithme de descente de gradient. Pour cela, il faut ac-

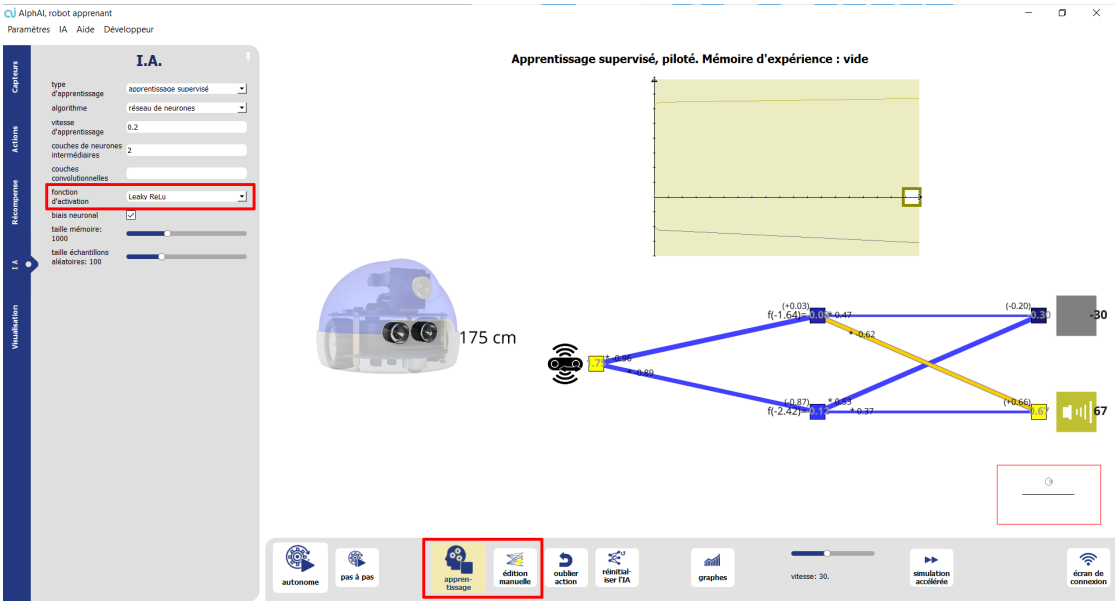

FIGURE 5 – Paramétrage pour apprentissage automatique

tiver le mode *apprentissage* et choisir comme fonction d'activation *Leaky ReLu*, comme dans la figure 5.

4.2 RÉALISATION DE L'APPRENTISSAGE

1. Nous allons réaliser un apprentissage en utilisant seulement 4 **points de données**. Deux points de données correspondant à l'action *ne rien faire* lorsque la distance mesurée par le robot est "normale", un point de données correpondant à l'action *bipper* lorsque la distance est trop grande et un autre point correspondant à l'action *bipper* lorsque la distance est trop faible. *Entrez ces 4 points de données dans la mémoire d'expérience du robot en cliquant sur l'action choisie en fonction de la distance mesurée.* Vous pouvez bien sûr utiliser un objet plat afin de contrôler la distance mesurée par le robot. La taille de la mémoire d'expérience est affichée en haut de l'écran. Vous pouvez effacer un point de données à l'aide du bouton *oublier action* et vider entièrement la mémoire à l'aide du bouton *réinitialiser l'IA*.
2. Observez l'évolution de l'apprentissage : valeurs des connexions et des biais, résultats en termes de courbes et de zones de décision. Pour rendre cette visualisation plus facile, vous pouvez désactiver l'affichage de l'*activité synaptique* dans l'onglet *visualisation*, et ouvrir la fenêtre des graphes à l'aide du bouton correspondant. L'apprentissage se déroule-t-il bien comme prévu ? Un apprentissage efficace devrait prendre moins de 5 minutes.
3. L'apprentissage automatique n'est pas toujours aussi précis et efficace que l'apprentissage humain. Pour rendre l'apprentissage plus aisé, vous pouvez augmenter le nombre de neurones de la couche intermédiaire. Essayez de trouver le nombre de neurones minimal qui permet d'obtenir de bons résul-

tats. Attention : il faut de nouveau remplir la mémoire d'expérience après chaque modification de la couche intermédiaire, vous pouvez donc choisir de la sauvegarder pour gagner du temps (dans les menus : *IA* > *Sauver la mémoire d'expérience*).

4. Vous pouvez également tenter de modifier le paramètre *vitesse d'apprentissage* dans l'onglet *IA*. Cela ne nécessite pas de remplir de nouveau la mémoire d'expérience. Attention : une vitesse d'apprentissage plus grande signifie aussi que le résultat obtenu sera moins bon ! Il faut donc trouver un juste mileu pour cette valeur.

4.3 Conclusion

À retenir :

En apprentissage supervisé, on dispose toujours de **données d'entraînement** pour l'apprentissage du réseau de neurones. il faut toujours faire suivre la phase d'apprentissage d'une phase de tests (ou validation) pour évaluer la qualité de notre modèle.

Les réseaux de neurones se basent sur les principes d'apprentissage et de communication des neurones du cerveau humain.

Le **biais neuronal** aide à contrôler la valeur d'activation des neurones permet ainsi d'augmenter le nombre de fonctions réalisables par un neurone.

Ajouter des **couches de neurones intermédiaires** sert à traiter les informations de façon plus approfondie, et permet notamment de les traiter avec des **fonctions non linéaires**.