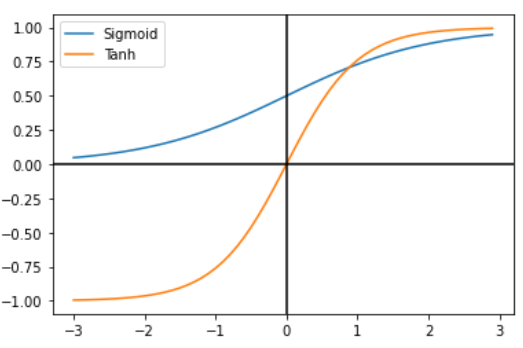
Le playground du site Tensorflow permet de simuler et de construire le comportement de perceptrons multicouches. De nombreux paramètres sont disponibles pour affiner le comportement de ces derniers.

* **Les fonctions d’activations**

Nous n’allons utiliser que les fonctions Sigmoïde et TanH. La fonction Sigmoïde n’est jamais négative, tandis que TanH peut l’être. TanH est centrée sur 0, tandis que Sigmoïde sur 0.5.



**Les poids et biais**

Nous avons donc les deux premiers neurones x1 et x2, qui ne bougeront pas, qui sortent respectivement une séparation verticale et horizontale et bien distincte sur les poids 0 et 1.

Plus le poids de chacun est important, plus l’output ressemblera à sa forme. Si les deux poids sont égaux, alors l’output est en forme de diagonale, ce qui est la moyenne des deux output de chacun des deux neurones présents. Une image contenant texte, capture d’écran, diagramme, logiciel

Description générée automatiquement

Ici sur l’image, les poids sont tous deux à 0.5. On remarque également que si on inverse les poids, ici on a mis -0,5 de chaque côté, la sortie s’inverse :

Une image contenant capture d’écran, texte, conception

Description générée automatiquement

Ici, pour bien visualiser, j’ai coché la case « discretize output » qui colore en bleu les valeurs négatives et en orange les valeurs positives. On peut décocher cette case pour avoir une visualisation plus précise, qui affiche du blanc à 0, et plus la sortie est élevée, plus la couleur est proche du bleu/orange.

Ajoutons maintenant un premier neurone dans une première couche :

Une image contenant texte, capture d’écran, diagramme, carte

Description générée automatiquement

Si les deux premiers n’envoient rien, donc des poids de 0, ce neurone ne renvoie rien non plus.

On remarque que contrairement aux deux premiers, ce neurone dispose d’un biais.

Essayons de comprendre comment la modification de ce dernier affecte la sortie.

Si un seul neurone envoie un poids, et que ce dernier renvoie un poids du même signe, avec un biais de 0, il est simplement l’image du neurone Xn qui lui envoie le poids.

Une image contenant texte, capture d’écran

Description générée automatiquement

Si on change le biais pour le passer à +2, on constate que le neurone renvoie toujours la même image, mais en accentuant les valeurs positives à hauteur de 2 dans les coordonnées :

Une image contenant texte, capture d’écran, diagramme, bleu vert

Description générée automatiquement

Naturellement, si le biais est négatif, c’est l’inverse qui se produira.

**Taux d’apprentissage**

D’autres paramètres sont présents comme le taux d’apprentissage (learning rate)

Lors de l’apprentissage, le réseau ajuste ses poids en suivant une descente de gradient.

Cet ajustement peut être défini par le learning rate qui est un paramètre que l’on peut modifier. Ce paramètre n’a pas de bonne réponse : plus il est petit, plus les ajustements seront précis et longs, mais on peut se retrouver enfermé dans un optimum local et ne jamais atteindre la meilleure solution.

Plus il est grand, plus on oscille rapidement, de manière moins précise, mais on peut sortir des optimums locaux. Il faut donc trouver un juste milieu, voire adapter ce taux d’apprentissage en cours d’apprentissage.

**Exemple 1 – Circle**

On comprend rapidement, en lançant la résolution du problème, qu’avec cette structure on ne peut qu’obtenir une séparation linéaire.

Une image contenant capture d’écran, texte, diagramme, ligne

Description générée automatiquement

En rajoutant un neurone dans la seconde couche, on arrive à obtenir une séparation parabolique. Cependant, cette séparation ne peut pas s’isoler totalement au centre afin de correspondre à notre structure de cercle imbriqué :

Une image contenant capture d’écran, texte, diagramme, ligne

Description générée automatiquement

On dirait que chaque neurone de la seconde couche forme une ligne, et elles viennent s’annuler à leur rencontre, ce qui forme la parabole. Pour essayer de faire un cercle, on pourrait donc tenter d’ajouter une troisième ligne pour venir couper le débordement que l’on voit en haut à gauche :

Une image contenant texte, capture d’écran, diagramme, Tracé

Description générée automatiquement

Les 3 lignes forment donc une sorte de triangle délimitant les valeurs positives des négatives, et remplissant donc complètement le pattern désiré.

Pour toutes ces manipulations, j’ai utilisé la fonction d’activation TanH. Avec Sigmoïd, on a aucun mal à obtenir le même résultat avec le même nombre de neurones :

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

**Exemple 2 – Exclusive OR**

Pour ce modèle, la configuration minimale avec TanH qui m’a permis d’obtenir un résultat satisfaisant est la suivante : Une image contenant capture d’écran, texte, diagramme, ligne

Description générée automatiquement

Avec Sigmoïd, le résultat est un poil moins précis mais tout de même satisfaisant.

Pour le rôle de chaque neurone de la couche cachée, on remarque que chacun d’entre eux s’occupe de couvrir un « coin », en mettant un poids positif ou négatif pour correspondre à la couleur à laquelle ce coin doit correspondre.

Cette structure satisfait les données de test, mais on remarque que les formes arrondies ne correspondent pas exactement à ce que l’on voudrait s’attendre pour un ou exclusif, j’entends par là qu’un OU exclusif est binaire, donc on voudrait avoir des séparations claires entre 0 et 1.

Voici la configuration minimale qui correspond à une structure davantage linéaire :

Une image contenant texte, capture d’écran, diagramme, carte

Description générée automatiquement

**Exemple 3 – Gaussian**

Ce modèle est de loin le moins complexe à résoudre, il ne nécessite pas de couches cachées. Il correspond à un mélange de x1 et x2 avec des poids à peu près similaires, et aucun besoin de biais ici.

**Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement**

On peut même compléter le modèle en enlevant l’un des deux neurones d’entrée :

Une image contenant texte, capture d’écran

Description générée automatiquement

**Exemple 4 – Spiral**

Ce modèle semble être de loin le plus complexe.

La configuration suivante montre le résultat le plus proche que j’ai réussi à obtenir (test loss le plus bas) :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

En essayant de réduire le nombre de neurones, j’ai obtenus d’autres résultats, moins bons mais tout de même satisfaisants comme celui-ci.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Afin d’y arriver, j’ai augmenté la variable « Ratio of training to test data”, qui augmente la concentration des points afin de reproduire la spirale plus fidèlement. Cela empêche au réseau de prendre des raccourcis qui ne devraient pas être pris, lorsqu’il y a des trous dans la spirale, comme c’est le cas en haut à droite :

