Essai d'utilisation des réseaux de neurones

Martin Desruisseaux, Michel Petit, Guillaume Constantin De Magny, Frédéric Huynh

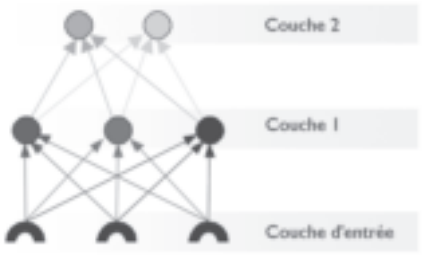
© D. Guyomard

Dans ce chapitre, nous passerons sous silence les délicates questions théoriques, voire philosophiques, qui accompagnent ces recherches pour n'aborder que les aspects pratiques nécessaires à la programmation d'un réseau de neurones de type *feed-forward*.

Il existe une multitude de façons de connecter des neurones entre eux. On pourrait par exemple imaginer un réseau où tous les neurones sont connectés avec tous les autres[1]. Le réseau que nous décrirons dans ce chapitre sera fait de couches (fig. 112). Ce type de réseaux (appelé en anglais *feed-forward*) à l'avantage d'être bien étudié par la littérature. La discussion qui suit résume les chapitre V et VI du livre de Hertz *et al.* (1991).

Dans les réseaux à couches, chaque neurone est connecté à tous les neurones de la couche précédente. On soumet un problème au réseau en donnant une valeur numérique à chaque neurone de la couche d'entrée. Ces neurones d'entrée ne faisant aucun calcul avec les valeurs qu'on leur donne, on ne considère pas vraiment qu'ils forment une couche du réseau. Dans la première « véritable » couche, chaque neurone utilisera les valeurs de tous les neurones de la couche d'entrée pour calculer sa propre valeur. Une fois cette étape complétée, chaque neurone de la seconde couche utilisera les valeurs de la première couche pour calculer leurs propres valeurs. Le processus continue ainsi pour toutes les autres couches.

1. Ce cas est effectivement étudié. Il représente une étape importante dans la compréhension des réseaux de neurones, un peu comme l'atome de Bohr en physique atomique.



▽ *Fig. 112*
Exemple de réseau de neurones à deux couches.

Chez les organismes biologiques, les cerveaux forment des réseaux de neurones bien plus complexes que ce réseau *feed-forward*. Les divisions en couches ne sont pas aussi nettes et les neurones se connectent entre eux d'une façon infiniment plus compliquée. Les équations exposées ici s'appliquent au cas relativement simple d'un réseau idéalisé, en aucun cas pour un réseau de neurones biologiques. Pour cette raison, nous n'utilisons le terme de « neurones » que parce qu'il est largement répandu. Pour ne pas pousser plus loin la confusion, nous bannirons de notre discussion le terme de « synapses » ; nous préférons parler de connections entre neurones.

Calcul d'une sortie à partir de valeurs d'entrée

Prenons un premier neurone du réseau, le neurone noir de la figure 113 par exemple. Ce neurone est connecté à tous les neurones de la couche précédente. À chaque connexion entre deux neurones est associé un poids w_{np} où n désigne le neurone étudié et p un neurone de la couche précédente. La sortie O_n du neurone se calcule alors par :

$$O_n = g\left(\sum_p w_{np} V_p\right)$$

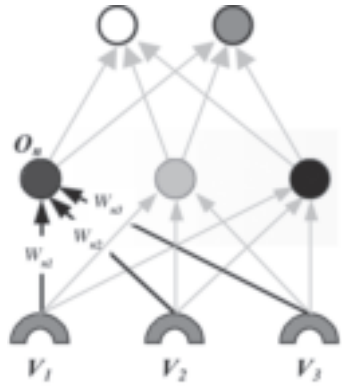
▽ *Équation 1*
Sortie d'un neurone en fonction de ses entrées.

Dans cette équation, $g(x)$ représente une *fonction d'activation* qui sera discutée par la suite. En notation vectorielle, on peut représenter la somme par une multiplication scalaire entre le vecteur d'entré V et les poids w_n d'un neurone n, que l'on peut voir en programmation comme un tableau à une dimension.

$$O_n = g\,(w_n V)$$

▽ *Équation 2*
Sortie d'un neurone (notation vectorielle).

Une fois calculées les sorties de tous les neurones de la première couche, ces sorties deviennent les entrées de la seconde couche et ainsi de suite jusqu'à la dernière couche. La figure 113 donne un exemple de calcul pour les neurones de la première couche d'un réseau qui comporte trois entrées.



▽ *Fig. 113*
Exemple de calcul de la sortie d'un neurone d'un réseau.
$O_n = g\,(w_{n1}V_1 + w_{n2}V_2 + w_{n3}V_3)$

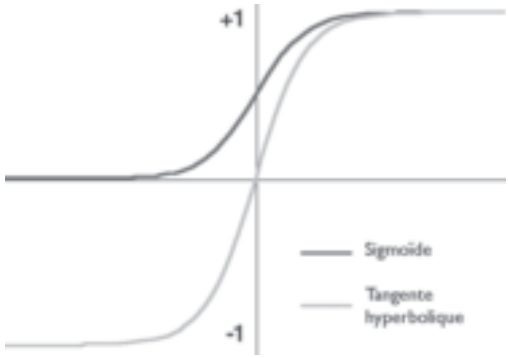
On appelle *couches cachées* toutes les couches qui se trouvent entre la couche d'entrée et la couche de sortie. Elles tiennent leur qualificatif de *cachées* du fait qu'elles n'ont aucun contact avec l'extérieur du réseau. Si on utilisait le réseau comme une boîte noire, on ignorerait leur présence. La figure 113 montre un exemple de calcul sur l'unique couche cachée du réseau illustré.

Sigmoïde

$$\mathrm{sig}(x) = \frac{1}{1+e^{-2x}}$$

Tangente hyperbolique

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



▽ *Fig. 114*
Illustration de deux sigmoïdes couramment utilisées comme fonctions d'activations.

Toutes les équations exposées dans ce chapitre font intervenir une *fonction d'activation*, dénotée *g(x)*. Cette fonction d'activation est généralement de forme sigmoïdale. De telles fonctions peuvent recevoir une valeur quelconque en entrée, mais retournent toujours une valeur comprise entre deux bornes en sortie. La figure 114 illustre deux sigmoïdes couramment utilisées. La première retourne toujours des valeurs comprises entre 0 et 1, tandis que la seconde (la tangente hyperbolique) retourne des valeurs comprises entre - 1 et + 1.

Pourquoi s'encombrer d'une fonction d'activation ? Ne pourrait-on pas calculer simplement $O_n = w_n \cdot V$ à la place de l'équation 1 ? En fait, de tels réseaux existent. On les appelle des réseaux *linéaires*, car ils reviennent à utiliser une fonction d'activation $g(x) = x$. MINSKY et PAPERT (1969) ont démontré que les réseaux linéaires ne peuvent résoudre qu'un nombre assez restreint de problèmes. Ils ont donc peu d'utilité dans la pratique. Ajouter des couches n'y change rien, car additionner des relations linéaires donne toujours une relation linéaire. Quel que soit le nombre de couches, un réseau sans fonction d'activation équivaut toujours à un réseau avec une seule couche.

En introduisant un caractère non linéaire, les fonctions d'activations augmentent les possibilités du réseau. Elles rendent utiles les couches cachées qui autrement auraient été superflues. Les fonctions de forme sigmoïdale ont aussi la propriété de présenter une pente plus abrupte pour les valeurs près de zéro. Lors de la phase d'apprentissage du réseau (discutée dans la prochaine section), cette propriété aura pour effet d'appliquer des corrections plus radicales sur les neurones qui « hésitent » encore sur les valeurs à produire.

Apprentissage supervisé du réseau

Pour obtenir du réseau qu'il produise les valeurs que l'on désire, il faut lui donner les bons poids w_{np}. On y parvient en soumettant le réseau à une phase d'apprentissage, pendant laquelle on lui spécifie à la fois les valeurs d'entrée et les valeurs de sortie qu'il devrait obtenir. Les poids appropriés sont alors calculés pour l'ensemble du réseau, y compris pour les couches cachées. L'algorithme effectuant cette tâche est connu en anglais sous le nom de *back-propagation*. Il en existe de multiples variantes, mais cette section ne présente que la plus classique.

On peut initialiser le réseau en donnant à ses poids des valeurs aléatoires. À partir d'un vecteur d'entré *V* qu'on lui spécifiera, le réseau calculera évidemment un vecteur de sortie *O* qui sera très différent de celui qu'on attendait. La différence entre la valeur calculée et la valeur attendue constitue l'*erreur* du réseau. L'idée de la méthode d'apprentissage par *back-propagation* consiste à modifier les poids de chaque connection par un facteur proportionnel à l'erreur. Plus spécifiquement, la correction à apporter au poids de la connexion entre

un neurone *n* de la couche de sortie et un neurone *p* de la couche précédente est la suivante :

$$C_{n,p} = \alpha \times O'_n \times (T_n - O_n) \times V_p$$

où t_n est la valeur attendue pour le neurone *n*.
α est le facteur d'apprentissage.

▽ *Équation 3*
Correction à apporter aux poids d'une connexion entre deux neurones.

Dans l'équation 3, O'_n est une expression similaire à O_n (équation 1), mais faisant intervenir la dérivée *g'* de la fonction de transfert plutôt que la fonction elle-même. Le tableau 14 rappelle les expressions de O_n et O'_n, ainsi que les dérivées premières des deux fonctions d'activations illustrées dans la figure 114.

▽ *Tableau 14*
Sortie d'un neurone et fonctions d'activations courantes.

	Fonction d'activation $g(x)$ et leurs dérivées premières
$O_n = g\left(\sum_p w_{np} V_p\right)$	$\mathrm{sig}(\beta x) = \frac{1}{1+e^{-2\beta x}}$ $\quad \mathrm{sig}'(\beta x) = 2\beta\left(\mathrm{sig} - \mathrm{sig}^2\right)$
$O'_n = g'\left(\sum_p w_{np} V_p\right)$	$\tanh(\beta x) = \frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}}$ $\quad \tanh'(\beta x) = b\left(1 - \tanh^2\right)$

WIDROW et STEARNS (1985) ont montré que lorsque α est suffisamment petit, l'application de ces corrections ajuste les poids des connections vers des valeurs qui minimisent l'erreur.

Application d'un réseau de neurones artificiels aux données de pêche

Nous avons construit un réseau de neurones avec suffisamment d'entrées pour recevoir des paramètres environnementaux sélectionnés et une sortie représentant le « succès » de la pêche (pas nécessairement représenté par le nombre d'individus capturés). Une multitude de configurations est possible pour le réseau. La figure 115 montre l'une de celles que nous avons essayée. Les entrées apparaissent en haut et la sortie en bas, avec une seule couche intermédiaire.

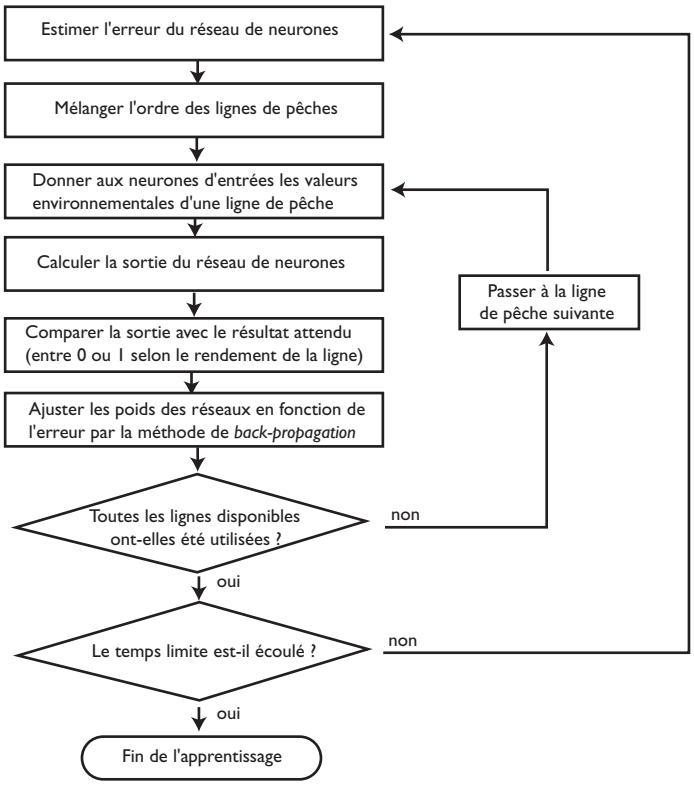
▽ *Fig. 115*
Exemple de réseau de neurones expérimenté sur les données de pêches.

Pour tous les neurones du réseau (incluant le neurone de sortie), nous avons utilisé une fonction d'activation sigmoïde. Cette fonction produit des valeurs proches de 0 ou de 1, avec parfois des valeurs intermédiaires quand le réseau hésite entre deux états. Cette fonction donne un caractère binaire à la sortie, qu'on pourrait décrire par « bonne pêche » ou « mauvaise pêche ». Or, les données de pêches disponibles expriment un rendement en nombre d'individus capturés par 1 000 hameçons. Il nous fallait donc un critère traduisant ce rendement en bonne ou mauvaise pêche. Le choix d'un tel critère est forcément arbitraire. Pour une première approche, nous avons fixé un seuil de rendement à environ 8 ‰. Ce seuil correspond au rendement moyen des lignes retenu pour notre simulation. Il est peut-être peu réaliste sur le plan halieutique, mais dans un premier temps il a l'avantage de séparer notre jeu de données en deux groupes de bonnes et mauvaises pêches de taille à peu près égale, ce qui facilitera l'interprétation des résultats.

Tous les réseaux ont d'abord été construits avec des poids aléatoires pour chacune de leurs connections. Nous avons ensuite entraîné les réseaux comme indiqué sur la figure 116.

L'erreur du réseau (la première étape du schéma précédent) est estimée en lui soumettant les données environnementales de chaque ligne de pêche, en calculant les sorties et en comparant ces dernières aux résultats attendus. On exprime l'erreur par la valeur RMS des différences entre les sorties du réseau et les résultats attendus (une différence par ligne de pêche).

L'erreur RMS du réseau initialisé avec des poids aléatoires (donc avant la phase d'apprentissage) est typiquement de l'ordre de 0,7. Rappelons que la sortie du réseau est comprise entre 0 et 1 et que l'erreur ne peut donc pas être supérieure à 1. Après la phase d'apprentissage, l'erreur descend à une valeur de l'ordre de 0,4, ce qui est encore élevé. Pour être plus exact, le niveau auquel descend l'erreur dépend de la définition donnée à une bonne et mauvaise pêche.



▽ *Fig. 116*
Soumission des données environnementales et de pêches pour apprentissage.

En effet, chaque définition produit sa propre distribution de 0 et de 1 et chaque distribution a sa moyenne et son écart-type. Dans tous nos essais avec les données environnementales et de pêches, l'erreur du réseau de neurones après la phase d'apprentissage était toujours très proche de l'écart-type de la sortie attendue.

Un réseau de neurones qui effectue son apprentissage sur des paramètres d'entrée et de sortie aléatoires, sans lien entre l'entrée et la sortie, obtient une erreur RMS à peu près égale à l'écart-type des données des sorties attendues. Nos résultats avec les données environnementales et de pêches donnaient une erreur RMS légèrement inférieure à l'écart-type, mais de très peu. Le faible écart entre les erreurs RMS et les écarts-types suggère qu'aux yeux du réseau de neurones, la relation entre les données environnementales et les captures d'espadons n'est pas très différente d'une relation aléatoire.

Ce résultat ne consacre par pour autant l'échec des réseaux de neurones. Plusieurs facteurs peuvent expliquer cette absence de résultat concluant :

– Le choix de la position à laquelle ont été évalués les paramètres environnementaux (la moyenne des positions de début et de fin) n'est peut-être pas approprié. Seules les positions de début et de fin sont à peu près fiables.

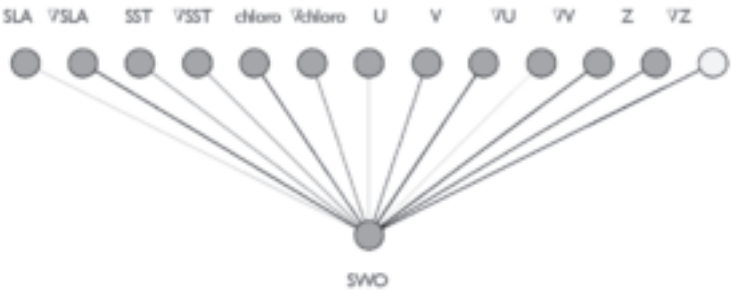
– Les conditions environnementales avant et après les pêches sont probablement des paramètres essentiels pour le réseau de neurones. Les analyses statistiques ont montré qu'ils ont souvent plus de poids que les conditions environnementales le jour de la pêche.

– Les données de pêches sont fortement biaisées par les contraintes du pêcheur (autonomie limitée du bateau, choix « intuitif » ou « guidé » par les cartes de SST, endroit où il met sa ligne à l'eau, etc.). Ce biais est peut-être suffisamment fort pour cacher le signal dû aux paramètres environnementaux.

– Il reste difficile de calibrer le temps d'apprentissage du réseau. Des essais sur des données artificielles (sur lesquelles nous avons volontairement introduit une relation systématique entre les données environnementales et les captures) ont montré que l'erreur du réseau de neurones pouvait plafonner longtemps à la valeur RMS de la sortie avant de chuter lorsque le réseau a trouvé une solution.

– De meilleurs réseaux de neurones et méthodes d'apprentissage peuvent probablement êtres expérimentés. La méthode d'apprentissage par *back-propagation* est reconnue comme étant lente et peu robuste. Une méthode plus complexe telle que *Scaled Conjugate Gradient* donnerait peut-être de meilleurs résultats. Cependant, nous pensons que le jeu de données de pêche couplées aux données environnement reste, de façon rédhibitoire, trop biaisé et surtout trop limité (deux années réelles).

Malgré le fait que les erreurs des réseaux de neurones soient proches de l'écart-type, les poids qu'ils établissent sont quand même assez cohérents avec les concepts océanographiques classiques et les résultats des autres méthodes testées. La figure 117 représente un exemple de réseau de neurones sans couche cachée. Un tel réseau n'apporte rien de nouveau par rapport aux analyses statistiques, mais a l'avantage d'être plus facile à interpréter qu'un réseau plus complexe. L'erreur RMS du réseau illustré est de 0,478 alors que l'écart-type est de 0,481. Les connections foncées représentent des poids plus forts que les connections pâles[2].



▽ *Fig. 117*
Exemple de réseau de neurones simple après apprentissage.

2. Afin d'éviter les biais dus au fait que l'ordre de grandeur de la profondeur est beaucoup plus grand que celui de la chlorophylle (par exemple), les paramètres environnementaux ont été normalisés avant de les soumettre au réseau de neurones. Cette normalisation ne change pas la performance du réseau de neurones, mais permet de relativiser les tons de gris dans des schémas comme celui de la figure 117 (ou 115 ?).

Les intensités des connections varient d'une exécution à l'autre du réseau de neurones. Mais pour le réseau simple illustré dans la figure 117, on retrouve le plus souvent un poids fort pour le gradient de SLA et le gradient de la composante U du courant géostrophique. Les poids de la SST et de la chlorophylle sont assez variables.

Ces observations ne sont peut-être pas significatives. Mais la piste des réseaux de neurones aurait besoin d'être poursuivie un peu plus loin avec un jeu de couples de données plus conséquents et moins biaisés.