Réseau Feed-forward

Le document suivant décrit un réseau de neurones feed-forward. Si cette description est déjà connue, il est possible de simplement lire la fin du document, qui indique dans quel ordre les paramètres du réseau doivent être fournis à la plateforme de défis.

Neurone formel

Un **neurone formel** (on dira simplement *neurone* dans la suite) est un élément de calcul caractérisé par des poids, un biais, et une fonction d'activation. Il effectue un calcul à partir de ses entrées numériques (il peut y en avoir un nombre arbitraire) et fournit une sortie numérique unique.

Les poids sont donnés par un vecteur contenant autant de valeurs que le neurone a d'entrées. Le biais est un nombre. Poids et biais sont des **paramètres** du neurones. La fonction d'activation (qui est plutôt un **métaparamètre** qu'un paramètre) peut être par exemple :

- la fonction identité : $Id(x) = x$
- la fonction *ReLU* (*Rectified Linear Unit*) : $ReLU(x) = x$ si $x \geq 0$ et $ReLU(x) = 0$ si $x < 0$
- ...

Par exemple, pour un neurone avec 3 entrées (x_1, x_2 et x_3), on pourrait choisir :

- des poids égaux à $[2, 6, -1]$
- un biais égal à 4
- la fonction d'activation *ReLU*

Ce choix est schématisé sur la figure suivante :

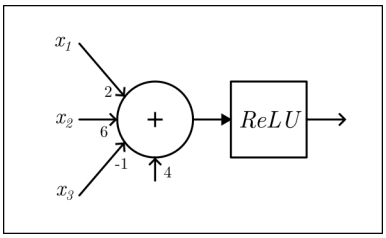

Figure 1: Un neurone formel

La sortie d'un tel neurone, fonction des entrées est :

$$ReLU([x_1, x_2, x_3].[2, 6, -1] + 4)$$

L'opération $[x, y, z].[a, b, c]$ désigne le produit scalaire entre deux vecteurs : $x \times a + y \times b + z \times c$.

Si l'entrée vaut $[1, 0, 2]$, alors la sortie vaut 4. En effet :

$$ReLU([1, 0, 2].[2, 6, -1] + 4) = ReLU((2 + 0 - 2) + 4) = 4$$

À partir de maintenant, on désignera par *entrée du neurone* (au singulier) le *n-uplet* de nombres qui constituent l'entrée, autrement appelé **vecteur d'entrée**, qui a pour taille 3 dans l'exemple qui précède.

Couche de neurones

Une **couche de neurones** est un ensemble de neurones partageant le même vecteur d'entrée. Ils ont généralement tous la même fonction d'activation (même si on pourrait envisager des fonctions d'activation différentes), mais ont des poids et des biais différents les uns des autres.

La figure 2 montre une couche de 3 neurones, partageant un vecteur d'entrée de taille 2. La fonction d'activation de chaque neurone est la fonction $ReLU$. Les deux poids et le biais de chaque neurone sont indiqués sur la figure, avec par exemple une entrée qui vaut $[2, -1]$.

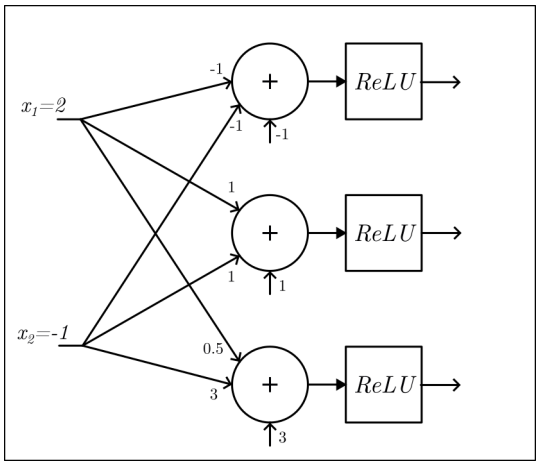

Figure 2: Une couche de neurones

La sortie de chaque neurone peut être calculée indépendamment, comme dans la section précédente, mais le formalisme matriciel permet une écriture plus compacte. Soit W la matrice de tous les poids, ici de dimension 3×2 (car il y a 3 neurones ayant 2 entrées), les poids de chaque neurone étant disposés en ligne :

$$W = \begin{pmatrix} -1 & -1 \\ 1 & 1 \\ 0.5 & 3 \end{pmatrix}$$

Soit B le vecteur de taille 3 contenant les biais des 3 neurones :

$$B = \begin{pmatrix} -1 \\ 1 \\ 3 \end{pmatrix}$$

Et soit E le vecteur d'entrée de la couche de neurones :

$$E = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

Alors le vecteur de sortie S de la couche est un vecteur de taille 3, donné par :

$S = ReLU(W \times E + B)$

Dans la formule qui précède, $\times$ désigne le produit matriciel, $+$ est l'addition de deux vecteurs, et $ReLU$ s'applique à un vecteur (on calcule $ReLU$ pour chaque composante du vecteur).

$$S = ReLU\left(\begin{pmatrix}-1 & -1\\ 1 & 1\\ 0.5 & 3\end{pmatrix} \times \begin{pmatrix}2\\ -1\end{pmatrix} + \begin{pmatrix}-1\\ 1\\ 3\end{pmatrix}\right) = ReLU\left(\begin{pmatrix}-1\\ 1\\ -2\end{pmatrix} + \begin{pmatrix}-1\\ 1\\ 3\end{pmatrix}\right) = ReLU\begin{pmatrix}-2\\ 2\\ 1\end{pmatrix} = \begin{pmatrix}0\\ 2\\ 1\end{pmatrix}$$

Ainsi, si l'entrée de la couche de neurones est $[2, -1]$, alors, avec les poids et biais choisis, la sortie sera $[0, 2, 1]$.

Réseau feed-forward

Un **réseau de neurones feed-forward** est un ensemble de couches de neurones, connectées de telle manière que la sortie d'une couche soit l'entrée de la suivante. L'entrée du réseau est l'entrée de la première couche, et la sortie du réseau est la sortie de la dernière couche.

Voici un exemple de réseau contenant 2 couches, la première couche contient 3 neurones, et la seconde couche en contient 2. La fonction d'activation de la couche d'entrée est $ReLU$, celle de la couche de sortie est l'identité. Les paramètres sont fixés à des valeurs aléatoires (symbolisées par `?` sur la figure).

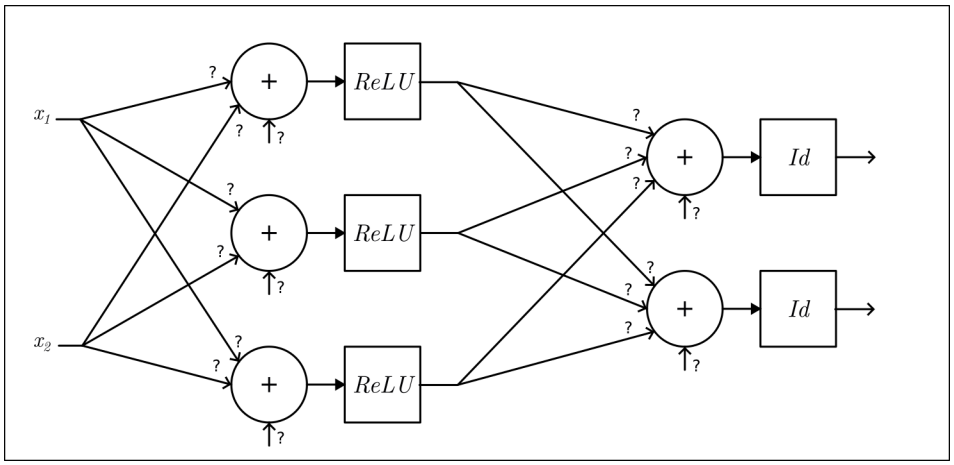

Figure 3: Un réseau de neurones

Apprentissage

L'objectif est de trouver les paramètres pour qu'à des vecteurs d'entrée choisis, correspondent des sorties souhaitées. Faire **apprendre** des couples entrée/sortie à un réseau de neurones (on parle d'**apprentissage supervisé**) consiste donc à trouver un jeu de paramètres (poids et biais) qui à chaque élément d'un ensemble de vecteurs d'entrée donnés fait correspondre la sortie souhaitée, avec une erreur *tolérable*.

On peut imaginer que chaque paramètre est ajustable avec un bouton (on peut augmenter ou diminuer sa valeur en tournant le bouton dans un sens ou dans l'autre). Le réseau contient 17 paramètres (poids et biais), et il y a donc 17 boutons à ajuster, sachant que pour un vecteur d'entrée fixé, chaque bouton modifie différemment les élément du vecteur de sortie du réseau.

La phase d'apprentissage consiste à ajuster tous les paramètres pour qu'à chaque vecteur d'entrée proposé, la sortie obtenue soit proche de la sortie souhaitée. Une fois l'apprentissage effectué, les paramètres restent fixes, et on peut évaluer le réseau en comparant la sortie calculée avec la sortie souhaitée pour chaque entrée. Cette phase d'apprentissage, pendant laquelle les paramètres sont ajustés, est réalisée à l'aide d'un algorithme dit de *rétropropagation du gradient*, duquel il existe de très nombreuses variantes, et qui consiste à ajuster petit à petit chacun des paramètres, de manière à se rapprocher de la sortie souhaitée pour chaque entrée. Le détail de cet

algorithme et de ses variantes dépasse le cadre de ce document, mais une recherche sur Internet (ou dans le Guide galactique) « *rétropropagation du gradient dans un réseau feed-forward* » permettra de se documenter sur le sujet si nécessaire.

Par exemple, supposons qu'on dispose de 5 entrée/sortie, chaque entrée et sortie étant un vecteur de taille 2 :

Entrée	Sortie
[2 , -1]	[1 , 3.5]
[0 , 0]	[1.5 , 2.5]
[0 , 1]	[4.5 , 1]
[1 , -1]	[0 , 3.25]
[1 , 0.5]	[4 , 2]

Le réseau suivant (obtenu après ajustement des paramètres, c'est à dire après la phase d'apprentissage) donne des résultats qu'on pourrait considérer convenables :

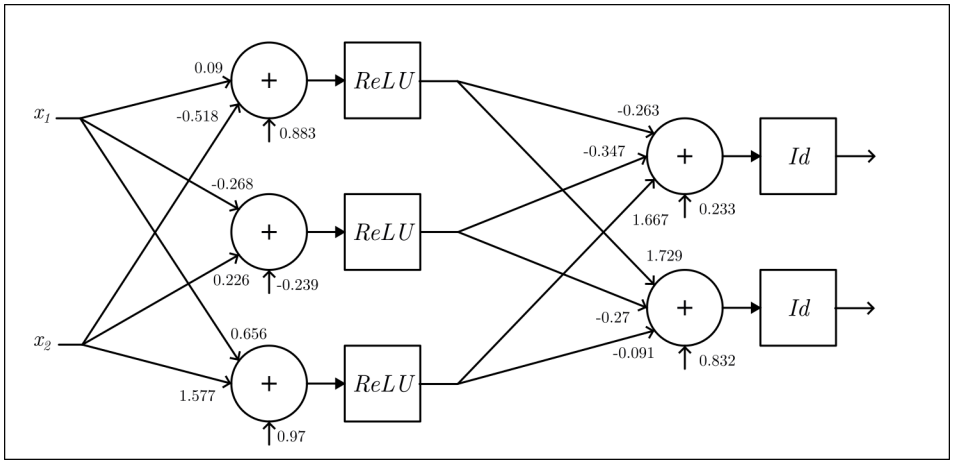

Figure 4: Un réseau de neurones après ajustement des paramètres

En effet, pour chaque entrée appliquée, on obtient les sorties suivantes, qui sont proches des sorties souhaitées :

Entrée	Sortie
[2 , -1]	[0.992 , 3.501]
[0 , 0]	[1.618 , 2.270]
[0 , 1]	[4.383 , 1.231]
[1 , -1]	[-0.077 , 3.405]
[1 , 0.5]	[4.070 , 1.847]

Il n'est pas exclu qu'on puisse s'approcher encore plus des valeurs souhaitées avec d'autres paramètres.

L'ensemble des paramètres de ce réseau (il y en a 17) est donné ci-dessous :

$$W_1 = \begin{pmatrix} 0.09 & -0.518 \\ -0.268 & 0.226 \\ 0.656 & 1.577 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} 0.883 \\ -0.239 \\ 0.97 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} -0.263 & -0.347 & 1.667 \\ 1.729 & -0.27 & -0.091 \end{pmatrix}$$

$$B_2 = \begin{pmatrix} 0.233 \\ 0.832 \end{pmatrix}$$

Remarque : L'objectif d'un tel réseau de neurones est la plupart de temps de fournir des valeurs de sorties *a priori* correctes pour des entrées inconnues non utilisées lors de la phase d'apprentissage. Autrement dit, une fois que le réseau a *appris* les données d'entraînement, on le teste sur des *nouvelles données*, ce qui fait parfois apparaître des problèmes supplémentaires de sous ou de sur-apprentissage. Ces problèmes ne sont pas abordés ici, et on se contente dans ce qui précède de décrire comment un réseau de neurones peut fournir des sorties souhaitées à des entrées choisies.

Description d'un réseau dans les défis

Important : Dans le cadre des défis, les paramètres sont donnés sous forme d'une unique liste obtenue en lisant ligne à ligne les matrices de poids puis le vecteur de biais, depuis la couche d'entrée jusqu'à la couche de sortie :

```
params = [ 0.09, -0.518, -0.268, 0.226,  0.656,  1.577, 0.883, -0.239, 0.97,
         -0.263, -0.347,  1.667, 1.729, -0.27,  -0.091, 0.233,  0.832]
```