CHIMIE ET ÉQUATION DIFFÉRENTIELLES ORDINAIRES NEURONALES

PARTIE 1

Thierry Lepoint

30 mai 2020

Table des matières

1 Introduction **4**

2 Modélisation de données **5**

2.1 Ajustement de courbes – Régression 5

2.2 Fonction de perte 6

3 Modélisation du taux de changement **7**

3.1 Efficacité paramétrique 7

3.2 Méthode d'Euler – Principe général pour la résolution d'EDOs 8

4 EDOs neuronales **9**

4.1 Aspects généraux 9

4.2 Lien entre EDOs et réseaux de neurones 10

4.3 Résolution de l'EDO neuronale 11

4.4 Rétro-différentiation 12

5 Conclusion **13**

Table des figures

1 Comment établir le lien entre t (le temps) et la concentration en réactif R au cours de la réaction chimique 1 ? 5

2 Schéma simplifié de la modélisation de données soit par régression *via* un algorithme itératif (figure 2a), soit avec des fonctions découlant de l'usage d'EDOs (figure 2b) 6

3 Illustration de la méthode d'Euler pour l'équation chimique 1 d'ordre 1. 9

4 Représentation générale d'un réseau de neurones artificiels. 15

5 Sortie pondérée ($\mathbf{h}(N)$) associée à un réseau avec une couche cachée composée de deux neurones. Les paramètres a_1 et a_2 permettent de régler la position et la largeur de l'impulsion tandis que w_1 règle la hauteur (l'impulsion carrée peut être négative). 16

6 Approximation d'une fonction continue par un réseau de neurones. . 17

1 Introduction

Nous consacrons deux articles rédigés à l'attention des chimistes et des biochimistes et dévolus aux équations différentielles ordinaires (EDOs) neuronales telles que traitées par Chen *et al.* [1] [1]. Récemment abordé dans la littérature [2], ce sujet est directement lié à l'intelligence artificielle et à l'apprentissage profond [3].

Dans le présent volet, nous traitons de la modélisation des données expérimentales, c'est-à-dire de l'établissement d'une relation entre les données du domaine d'entrée (par exemple, le temps) et de sortie (par exemple, la concentration en une espèce chimique) :

- par régression (section 2) avec minimalisation de la fonction de perte (sous-section 2.2),
- par approximation *via* une équation différentielle ordinaire (EDO), forme fonctionnelle du taux de changement reflétant la relation entre les données du domaine d'entrée et du domaine de sortie [4]. Cette approche permet de gagner en efficacité paramétrique par rapport aux méthodes de régression (sous-section 3.1). Dans la sous-section 3.2, nous rappelons la méthode de base d'Euler parce qu'elle permet de faire le lien avec les EDOs neuronales.
- par réseaux de neurones artificiels (voir section 4). Après une brève introduction (sous-section 4.1) renvoyant à l'annexe B qui brosse le principe général et les paramètres de fonctionnement d'un réseau de neurones artificiels, nous introduisons l'approche de Chen *et al.* [1] qui permet de connecter les EDOs à la séquence que certains réseaux de neurones établissent entre les domaines d'entrée et de sortie des données (sous-section 4.2).

Dans le cadre de la comparaison entre régression et EDO, notre fil conducteur prend en compte une réaction chimique :

$$R \xrightarrow{k_1} P \tag{1}$$

dont on étudie la cinétique de consommation en réactif R. P désigne un (ou des) produit(s) et k_1, le coefficient cinétique. Pour fixer les idées, il peut s'agir de la thermolyse de l'azométhane, réaction d'ordre 1 par rapport à ce composé [5] :

$$CH_3N_2CH_3\,(g) \underset{600\,K}{\longrightarrow} CH_3CH_3\,(g) + N_2\,(g)$$

Dans le second article [6], tout en utilisant le langage de programmation Julia, nous

1. La relation entre réseaux de neurones artificiels et équations différentielles a été étudiée depuis 2017 par Weinan, Lu *et al.*, Haber et Ruthotto, Ruthotto et Haber (Réf. [2]).

mettons en application le principe des EDOs neuronales pour traiter la mise en forme du modèle théorique du Brusselator [6] et son injection dans un réseau de neurones artificiels. Parmi d'autres modèles, le Brusselator illustre le comportement non-linéaire d'un système chimique auto-catalytique [2].

2 Modélisation de données

Imaginons que soit étudiée la réaction chimique 1 en suivant par dosimétrie la disparition du réactif R au cours du temps.

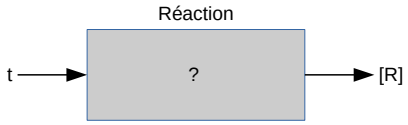


FIGURE 1 – Comment établir le lien entre t (le temps) et la concentration en réactif R au cours de la réaction chimique 1 ?

À la fin de l'expérience, la dosimétrie permet d'obtenir un domaine constitué de N paires de données expérimentales, $\mathscr{D} = \{(t_1, [R]_1), (t_2, [R]_2), ..., (t_N, [R]_N)\}$. Comment établir le lien $t \rightarrow R$ (figure 1) ? Pour être prédictible, c'est-à-dire pour attribuer à tout instant t_i une concentration R_i, il convient de corréler par modélisation le domaine d'entrée $\mathscr{T}$ (la collection des temps mesurés) au domaine de sortie $\mathscr{R}$ (la collection des concentrations en réactif R déterminées par dosimétrie). De manière classique, deux approches peuvent être utilisées (voir la figure 2) :

— la modélisation par une méthode de régression qui établit un lien direct entre les domaines d'entrée et de sortie, en utilisant toutes les données [8].

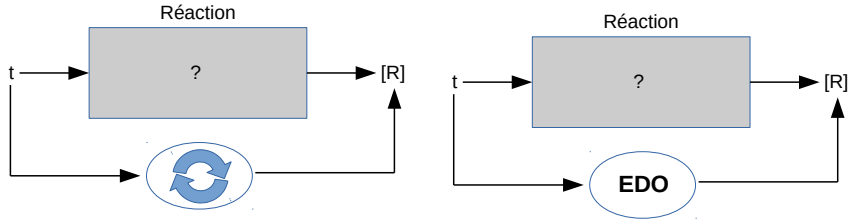
— la modélisation par une EDO qui permet de décrire le taux de changement entre $\mathscr{R}$ et $\mathscr{T}$ [4]. Selon les cas, la résolution de l'EDO se fait de manière analytique ou à l'aide d'un solveur numérique.

Dans la section 4, nous verrons que peuvent être utilisés les réseaux de neurones artificiels (expression ci-dessous simplifiée en « réseaux de neurones »). Ils correspondent à la catégorie schématiquement décrite à la figure 2a.

2.1 Ajustement de courbes – Régression

L'ensemble des données brutes $\mathscr{D}$ issues de l'expérience ci-dessus peut être disposé dans un graphique $\mathbb{R}^2$.

2. L'auto-catalyse est invoquée, par exemple, en chimie prébiotique.



(a) La régression (mais aussi les réseaux de neurones) utilise un algorithme itératif traitant toutes les données pour établir une corrélation entre domaines d'entrée et domaine de sortie.

(b) Modélisation des données par une EDO.

FIGURE 2 – Schéma simplifié de la modélisation de données soit par régression *via* un algorithme itératif (figure 2a), soit avec des fonctions découlant de l'usage d'EDOs (figure 2b) .

Une méthode de régression directement appliquée à l'ensemble $\mathscr{D}$ consiste à établir une fonction paramétrique telle que (le symbole $\hat{}$ indique une moyenne) :

$$[\hat{R}] = [R]_0 \exp(-k_1 t) \tag{2}$$

Le lien entre les concentrations et les données de temps est calculé en ajustant les paramètres libres k_1 et $[R]_0$. L'algorithme exécuté est toutefois contraint.

2.2 Fonction de perte

En effet, il convient de minimiser l'erreur ε associée à chaque valeur de concentration, définie comme $\varepsilon = [R]_t - [\hat{R}]_t$. Le terme ε représente le bruit aléatoire inhérent aux données expérimentales. Sur l'ensemble de $\mathscr{D}$, l'écart aux valeurs moyennes s'appelle la *fonction de perte*.

Utilisant la technique de la *descente de gradient* [3, 11], l'algorithme machine calcule les valeurs optimales du coefficient cinétique et de la concentration en R. Appelons-les respectivement, k_1^* et $[R]_0^*$. La fonction s'approchant au mieux des résultats expérimentaux s'exprime comme :

$$f^*(t) = [R]_0^* \exp(-k_1^* t)$$

Ici, ressort bien le fait que les méthodes de régression établissent une relation directe entre domaine d'entrée ($\mathscr{T}$, ici le temps) et domaine de sortie ($\mathscr{R}$, ici les concentrations en R) d'une part et que de l'autre, elles supposent la connaissance de la forme de la fonction reliant domaines d'entrée et de sortie (voir l'annexe A).

3 Modélisation du taux de changement

Avec l'équation (2), nous n'avons rien écrit d'autre qu'une relation continue et différentiable [3] entre t et $[R]$:

$$[R] = f(t)$$

Par conséquent, il est permis d'écrire :

$$\frac{d[R]}{dt} = f^{'}(t) \tag{3}$$

Cette formulation décrit le *taux de changement*, c'est-à-dire la manière dont $[R]$ varie lorsque t varie. L'équation (3) est la forme de base d'une EDO qui peut être résolue par intégration analytique ou, lorsque ce n'est plus possible, par voie numérique à l'aide d'un solveur. De cette manière, connaître $f^{'}(t)$, selon l'approximation qu'on peut lui donner, c'est connaître indirectement $f(t)$.

Pourquoi est-il judicieux de s'intéresser à une mise en forme telle que l'équation (3) ?

Outre le fait que les EDOs sont maîtrisées depuis longtemps, il existe deux raisons majeures :

1. l'usage des équations différentielles réduit le nombre de paramètres libres par rapport aux méthodes de calcul reliant directement les données d'entrée aux données de sortie. C'est l'*efficacité paramétrique* (voir la section 3.1) qu'exploitent les techniques de résolution d'EDOs. Cette efficacité se double d'une efficacité concernant la correction des divergences associées à la méthode numérique de base (Euler, voir la section 3.2).
2. la réduction du nombre de paramètres ajustables et de corrections de divergence contribue à l'*efficacité numérique*, ce qui constitue un point essentiel du couplage « EDOs – réseaux de neurones » (voir la section 4).

3.1 Efficacité paramétrique

Dans la méthode de régression (section 2.1), il est notable que la concentration à tout instant dépend de deux paramètres, k_1 et R_0, librement ajustables pour minimiser la fonction de perte. Dans une approche EDO, nous connaissons la forme paramétrique de la différentielle de l'équation (2), c'est-à-dire :

3. Étant donné que nous utilisons des fonctions réelles sur un intervalle ouvert de $\mathbb{R}$, les termes « dérivables » et « différentiables » sont équivalents.

$$\frac{d[\hat{R}]}{dt} = -k_1[R]$$

Dès lors, nous approximons la relation entre des points du domaine d'entrée $\mathscr{T}$ et de sortie $\mathscr{R}$, en optimisant la forme fonctionnelle du taux de changement :

$$\frac{d[R]}{dt} \approx f(t, [R]) = -k_1[R] \tag{4}$$

de sorte que nous observons qu'il n'y a plus qu'un paramètre ajustable, k_1. Dans l'équation 4, $f(t, [R])$ est le gradient de l'intégrale recherchée.

Une résolution analytique exigerait d'ajuster dynamiquement le paramètre $[R]_0$. Numériquement, la résolution ne nécessite que la connaissance d'une condition initiale *fixe*, en l'occurrence $(t_0, [R]_0)$. De ce fait l'optimisation numérique d'une EDO est plus efficace que celle d'une régression.

3.2 Méthode d'Euler – Principe général pour la résolution d'EDOs

La méthode d'Euler [4] de résolution numérique des EDOs est la base des deux grandes catégories d'algorithmes : ceux de Runge-Kutta [9] et d'Adams-Bashfort [10].

La méthode d'Euler découle de la définition de l'approximation tangentielle du gradient en un point :

$$\frac{d[R]}{dt} \approx \frac{[R](t+\delta) - [R](t)}{\delta}$$

où δ est un pas de temps fixe. Par réarrangement, et tenant compte de l'équation (3), la valeur de la concentration en R à $t+\delta$ peut être déterminée :

$$[R](t+\delta) = [R](t) + \delta\frac{d[R]}{dt} = [R](t) + \delta f(t, [R])$$

Afin de calculer des approximations de $[R]$ par la méthode d'Euler, il convient de discrétiser le domaine de la fonction. Le calcul commence au point initial fixe $(t_0, [R]_0)$ et se poursuit selon une trajectoire récursive où, avec $n = 0, 1, 2, \ldots$:

$$\begin{cases} t_{n+1} = t_n + \delta(n+1) \\ [R]_{n+1} = [R]_n + \delta f(t_n, [R]_n) \end{cases} \tag{5}$$

À chaque pas, la valeur de $[R]_n$ est utilisée pour calculer la valeur suivante $[R]_{n+1}$ et le gradient indique à la fois la direction du déplacement et sa magnitude. L'allure générale d'une résolution selon Euler est donnée à la figure 3.

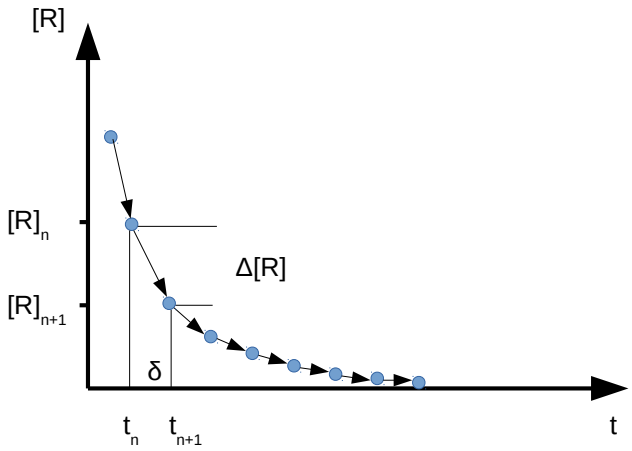


FIGURE 3 – Illustration de la méthode d'Euler pour l'équation chimique 1 d'ordre 1.

La méthode numérique d'Euler n'est plus utilisée comme telle à cause de divergences récurrentes des solutions. Restant sensibles à une divergence progressive des solutions à chaque pas de calcul, les solveurs modernes d'équations différentielles effectuent un contrôle de la fonction de perte. À nouveau, le principe consiste à recourir à la *descente de gradient* [3, 11]. En cas de divergence supérieure à la tolérance accordée par le numéricien, un rétro-justement est opéré mais pas nécessairement à chaque pas de calcul, ce qui constitue un avantage numérique. Enfin, l'usage de pas adaptatifs et l'optimisation du nombre de pas de calcul renforcent l'efficacité numérique des solveurs modernes d'équations différentielles.

4 EDOs neuronales

4.1 Aspects généraux

Lorsque les méthodes de régression et les EDOs classiques ne sont plus utilisables (nombre de données à traiter très grands et/ou complexité de la relation entre domaines d'entrée et de sortie), les réseaux de neurones entrent en scène. Le principe de ces réseaux consiste :

— à établir une corrélation directe entre domaine d'entrée et domaine de sortie au cours d'une phase d'apprentissage,

— à utiliser la fonction d'apprentissage obtenue pour produire de manière fiable toute valeur de sortie en fonction de toute nouvelle valeur d'entrée.

En accord avec la description et la définition des paramètres reprises à l'annexe B et, dans le cas d'un réseau discret, le passage d'une couche cachée $\mathbf{h}_n$ à la suivante s'exprime comme :

$$f(\mathbf{h}_n, \theta) = \sigma \left[\sum_{i=1}^{N} \theta h_n \right]$$

Qu'ont en commun les réseaux de neurones et les EDOs ?

4.2 Lien entre EDOs et réseaux de neurones

Les réseaux résiduels, récurrents ou à normalisation de flux [4] –tous discrets– utilisent des transformations d'une couche cachée $\mathbf{h}_n$ à la suivante $\mathbf{h}_{n+1}$ selon l'expression (voir les réf. [1, 12]) :

$$\mathbf{h}_{n+1} = \mathbf{h}_n + f(\mathbf{h}_n, \theta) \tag{6}$$

Dans cette description, la profondeur du réseau est parcourue de manière spatiale, à travers les n couches (*cf.* la figure 4).

Dans un premier temps, imaginons que la profondeur du réseau soit parcourue de manière temporelle, à travers t, cette échelle « se superposant » à celle de n. L'équation 6 se réécrit :

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \Delta t \cdot f(\mathbf{h}(t), t, \theta) \tag{7}$$

La fonction de transformation $f(\mathbf{h}_n, \theta)$ devient dès lors une fonction fréquentielle, $f(\mathbf{h}(t), t, \theta)$, correspondant au nombre de pas par unité de temps. Il est remarquable que cette reformulation soit analogue à celle de l'équation 5 associée à la méthode d'Euler.

L'équation 7 peut être remaniée :

$$\frac{\mathbf{h}_{t+1} - \mathbf{h}_t}{\Delta t} = \frac{\Delta \mathbf{h}(t)}{\Delta t} = f(\mathbf{h}(t), t, \theta)$$

Dans un second temps, imaginons que soient ajoutées de plus en plus de couches cachées avec des pas de plus en plus fins. À la limite, il est possible d'écrire [1] :

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta) \tag{8}$$

4. Par exemple, les réseaux résiduels sont utilisés dans la reconnaissance d'images. Les réseaux récurrents (notamment à large mémoire à court terme) sont efficaces dans la reconnaissance vocale, la reconnaissance et la compréhension de textes.

Nous retrouvons ici la forme fonctionnelle d'une EDO qu'un solveur numérique moderne peut résoudre. À présent, le réseau de neurones n'est plus perçu comme discrétisé ; au contraire, il est décrit par une fonction continue.

En partant de la couche d'entrée $\mathbf{h}(0)$, nous pouvons définir la couche de sortie $\mathbf{h}(T)$ comme étant la solution à ce problème d'EDO à valeur initiale pour un temps T.[5] Ceci permet de profiter des avantages décrits à la section 3 concernant les équations différentielles.

4.3 Résolution de l'EDO neuronale

Avant tout, illustrons comment pratiquer pour résoudre une EDO simple, en utilisant le langage Julia[6]. L'équation est

$$\frac{du}{dt} = f(u, t, p)$$

où l'interval de temps $t \in [0.0, 1.0]$ et $f(u,\ t,\ p) = -\alpha u$. C'est la reformulation de l'équation chimique 1 où u représente $[R]$ et $\alpha = k_1$ (cinétique d'ordre 1).

Listing 1 – Résolution numérique d'une EDO élémentaire avec u0 normalisé et k, une valeur numérique.

```
1 using DifferentialEquations
2 f(u,p,t) = -k*u
3 u0 = 1.0
4 tspan = (0.0,1.0)
5 prob = ODEProblem(f,u0,tspan)
6 sol = solve(prob)
```

Pour résoudre numériquement l'équation 8, il nous faut une condition initiale d'entrée ; il s'agit d'$\mathbf{h}(t_0)$. La sortie sera évaluée à la fin du segment `tspan` (t_1). Nous obtiendrons ainsi $\mathbf{h}(t_1)$. Sur l'axe du temps intrinsèque au réseau désormais continu, t_0 et t_1 ne sont pas connus et la matrice θ non plus. Il s'agit de paramètres libres ; la phase d'apprentissage a pour but de les calculer.

L'équivalent de la ligne 5 du listing 1, s'écrit :

5. Il faut être attentif au fait que le t de l'équation 7, n'est pas le t que nous avons utilisé précédemment dans le domaine d'entrées $\mathscr{T}$, pour l'équation chimique 1. Ici, il s'agit d'un paramètre intrinsèque au réseau de neurones, se superposant à n.

6. Julia est syntaxiquement proche de MATLAB, mais utilisant une compilation à la volée (JIT, *Just in Time*), ses performances en terme de vitesse d'exécution sont comparables à celles du langage C.

$$prob = ODEProblem(f, \mathbf{h}(t_0), t_0, t_1, \theta)$$

La solution `sol=solve(prob)` génère $\mathbf{h}(t_1)$.

4.4 Rétro-différentiation

Cependant, la principale difficulté associée à l'apprentissage selon un mode continu consiste à effectuer une *rétro-différenciation* à travers le solveur d'EDO afin de minimiser la fonction de perte. La rétro-différentiation est l'équivalent de la rétro-propagation en mode discret (voir l'annexe B). Cette fonction de perte s'exprimant comme :

$$L = L(\mathbf{h}(t_1)) = L[solve(ODEProblem(f, \mathbf{h}(t_0), t_0, t_1, \theta))]$$

il est difficile de la calculer directement pour des raisons de consommation de mémoire et du fait qu'elle-même engendre des erreurs [1]. Pour pallier à ces problèmes, Chen *et al.* [1] font appel à la *méthode adjoint*e pour calculer les gradients de L . Ils sont deux : l'un dépend de l'état du système ($\mathbf{h}(t)$) au temps t, l'autre des paramètres d'apprentissage, θ. Ceci constitue la seconde contribution décisive de Chen *et al.*. Ces auteurs définissant un état adjoint $a(t)$, ils résolvent les deux gradients en remontant de t_1 à t_0 de cette manière (avec $^\intercal$ indiquant la transposée) :

— pour le gradient par rapport à $\mathbf{h}(t)$:

$$\frac{\partial L}{\partial \mathbf{h}(t)} = \int_{t_1}^{t_0} a(t)^\intercal \frac{\partial f(h(t), t, \theta)}{\partial h(t)} dt \tag{9}$$

— pour le gradient par rapport aux paramètres d'apprentissage θ :

$$\frac{\partial L}{\partial \theta} = \int_{t_1}^{t_0} a(t)^\intercal \frac{\partial f(h(t), t, \theta)}{\partial \theta} dt \tag{10}$$

La troisième contribution décisive de Chen *et al.* est de montrer (annexe B de la réf. [1]) que les trois équations 8, 9 et 10 peuvent être traitées par un seul appel au même solveur en vectorisant leurs paramètres (voir complémentairement la réf. [15]).

Dans le deuxième volet, nous voyons comment utiliser cette technique dans le cas du Brusselator [6].

5 Conclusion

Les réseaux de neurones constituent une évolution importante dans le traitement des données lorsque celles-ci sont très nombreuses et/ou que la relation non-linéaire entre données d'entrée et de sorties est complexe à établir. En principe, l'augmentation du nombre de couches cachées et du nombre de neurones internes à ces couches doit rendre l'apprentissage profond plus performant. Toutefois, cette pratique conduit à une imprécision de l'apprentissage (que divers procédés tentent de résorber) qui retentit sur la capacité à prédire un résultat fiable. Cependant, une discrétisation de plus en plus fine s'apparente à la technique de traitement des pas infinitésimaux sur laquelle sont fondées les équations différentielles. L'idée centrale de Chen *et al.* [1] est de substituer les couches discrètes des réseaux par des équations différentielles ordinaires. Outre le fait que cette méthode permet de contrôler explicitement le compromis entre vitesse de calcul et précision, un autre de ses avantages est de pouvoir traiter des données d'entrée distribuées irrégulièrement en termes d'acquisition. Encore jeune, la méthode des EDO neuronales est prometteuse. Si elle peut jouer un grand rôle dans le traitements des données médicales [18], son potentiel devrait s'étendre à la (bio)chimie et la physique.

Cependant, il a été remarqué que les EDOs neuronales ne peuvent pas traiter toutes les fonctions. Une extension est proposée par Dupont *et al.* [19] pour résoudre cette limitation grâce aux EDOs neuronales augmentées (ANODEs pour *Augmented Neural ODEs*) donnant des solutions empiriquement plus stables avec une meilleure capacité de généralisation et un moindre coût de calcul. De nouvelles perpectives s'ouvrent encore avec les équations différentielles neuronales stochastiques à saut [20], utilisables par exemple en géophysique dans l'étude des tremblements de terre.

Annexe A

Nous référant à l'étude de la réaction chimique 1 par régression, un (bio)chimiste aurait pu utiliser une mise en forme semi-logarithmique en représentant $\ln \frac{[R]}{[R]_0} = f(t)$ dans l'espace $\mathbb{R}^2$. Ceci aurait permis d'exploiter une régression linéaire caractérisée par une fonction de perte minimisée par l'erreur quadratique moyenne $L(t, [R])$ telle que :

$$L(t, [R]) = \frac{1}{n} \sum_{i=1}^{n} ([R]_i - [\hat{R}]_i)^2$$

où n est le nombre total de mesures et où i indice toute valeur individuelle de concentration en R sur l'axe des temps.

Cependant, quel que soit le type de régression, nous observons qu'il est nécessaire de connaître la forme fonctionnelle exacte qui relie l'entrée à la sortie.

Annexe B

Fonctionnement d'un réseau de neurones

D'une manière générale, les réseaux de neurones sont représentés avec une couche d'entrée $\mathbf{h}(0)$, un état latent constitué de n couches cachées ($\mathbf{h}_n$) et une couche de sortie $\mathbf{h}(N)$ (figure 4)[7] [2]. La notation $\mathbf{h}$ désigne un vecteur. Par exemple, le vecteur $\mathbf{h}_2$ contient un ensemble de nombres réels $[h_{2,1},\ h_{2,2},\ \ldots,\ h_{2,m}]$. Pour des raisons de simplicité en relation avec l'interprétation de la section 4.1, nous considérons que le nombre de neurones dans chaque couche est le même, en l'occurrence m, comme indiqué en figure 4.

Lorsqu'un neurone du vecteur $\mathbf{h}_1$ (par exemple, $h_{1,2}$) transmet son information numérique à un neurone de $\mathbf{h}_2$ (par exemple, $h_{2,5}$), ce passage s'effectue à l'intervention d'une fonction non-linéaire σ telle que :

$$\sigma(z) = \sigma(w_{1,5} \cdot h_{1,2} + b_{1,5}) \rightarrow h_{2,5}$$

Le neurone $h_{2,5}$ prend la valeur de $\sigma(z)$. Ceci est schématisé par la flèche rouge de la figure 4. La fonction σ peut être une sigmoïde ($\frac{1}{1+e^{-z}}$). Les paramètres numériques $w_{1,5}$ et $b_{1,5} \in \mathbb{R}$ sont ajustés durant la phase d'apprentissage du réseau[8]. Supposons

7. L'expérience montre que lorsque le nombre de couches > 2, nous avons à faire à de l'apprentissage profond.

8. Pour des raisons pratiques, nous utilisons la notation en w et b qui provient des termes anglais *weight* et *bias*.

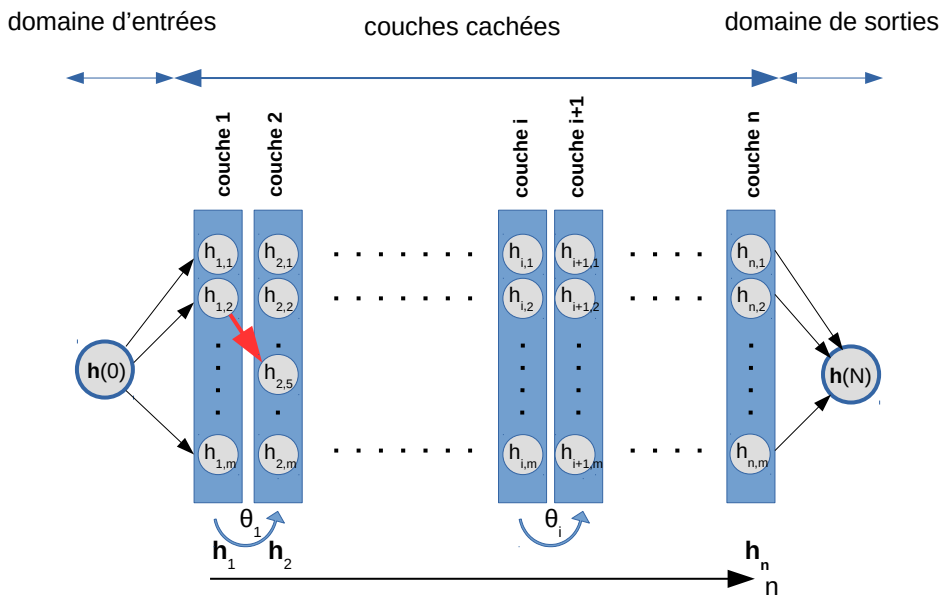


FIGURE 4 – Représentation générale d'un réseau de neurones artificiels.

donc que σ soit une sigmoïde suffisamment raide pour imiter une marche d'escalier. Le saut de cette marche apparaît à $-\frac{b_{1,5}}{w_{1,5}}$, rapport que nous définissons comme $a_{1,5}$.

Concernant la transformation de la couche 1 à 2, nous trouvons un vecteur tel que :

$$\theta_1 = \begin{bmatrix} a_{1,1} \\ a_{1,2} \\ \vdots \\ \\ a_{1,m} \end{bmatrix}$$

D'une manière générale, à chaque passage de couches, existe un vecteur (figure 4). Ces vecteur peuvent être regroupés sous forme d'une matrice :

$$\theta = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{n,1} \\ a_{1,2} & a_{2,2} & & a_{n,2} \\ \vdots & \vdots & & \vdots \\ \\ a_{1,m} & a_{2,m} & & a_{n,m} \end{bmatrix}$$

Apprentissage du lien $\mathbf{h}(0)$-$\mathbf{h}(N)$

Le théorème d'approximation universel établit que toute fonction continue est d'autant plus approximable par un réseau de neurones qu'il y a de paramètres au sein d'une couche cachée [13] et qu'il y a de couches cachées au sein du réseau [14]. Autrement dit : que m et i sont grands [9].

Limitons-nous à un réseau à une seule couche cachée contenant 2 neurones, caractérisé par a_1 et a_2 ainsi que par $w_1 = -w_2$ (une description intéractive est donnée à la réf. [15]). Si à l'entrée de la couche cachée, c'est-à-dire de $\mathbf{h}(0)$ au neurone 1, nous utilisons une valeur élevé de w_0, nous obtenons une sigmoïde raide approchant une marche d'escalier. En répétant la même opération pour le neurone 2, alors en sortie (c'est-à-dire en $\mathbf{h}(N)$), nous obtenons deux sigmoïdes raides symétriques approchant une fonction « impulsion carrée » (figure 5).

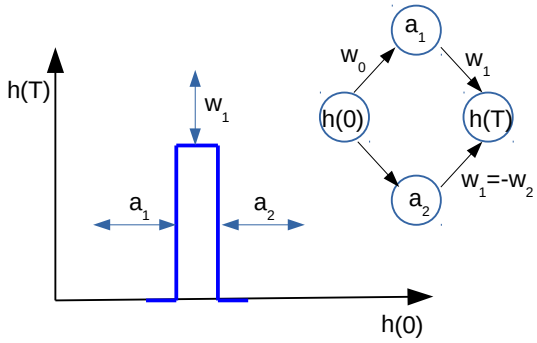


FIGURE 5 – Sortie pondérée ($\mathbf{h}(N)$) associée à un réseau avec une couche cachée composée de deux neurones. Les paramètres a_1 et a_2 permettent de régler la position et la largeur de l'impulsion tandis que w_1 règle la hauteur (l'impulsion carrée peut être négative).

La double opération que nous venons d'effectuer peut être répétée avec deux neurones supplémentaires de sorte à ce que les impulsion carrées fusionnent, leur hauteur étant éventuellement différentes. Dès lors, si les paires de neurones sont multipliées, le nombre d'impulsions le sera aussi. En réglant les paramètres a et w, il devient possible d'approximer toute fonction continue (figure 6). Le principe peut être étendu dans la profondeur du réseau, c'est-à-dire en augmentant le nombre de couches cachées.

Durant la phase d'apprentissage, un réseau de neurones reçoit un vecteur $\mathbf{h}(0)$ et modifie les paramètres θ_i pour approximer les valeurs constitutives d'un vecteur $\mathbf{h}(N)$

9. Il existe deux réserves : (i) il s'agit d'une approximation, de telle sorte qu'à chaque réseau de neurones est associée une fonction de perte à minimiser, (ii) en principe, ne sont approximables que des fonctions continues.

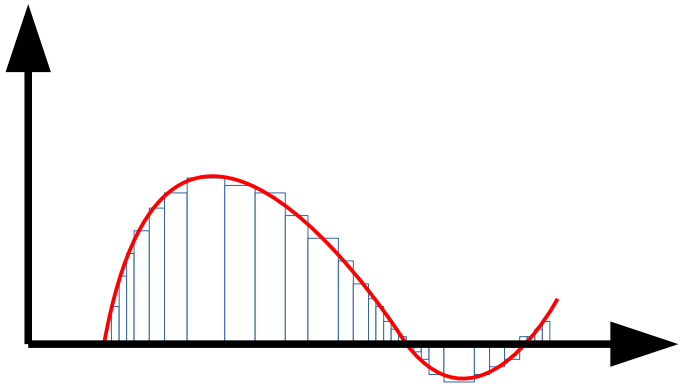


FIGURE 6 – Approximation d'une fonction continue par un réseau de neurones.

connu. L'optimisation des valeurs contituant les vecteurs θ_i se fait en minimisant la fonction de perte. Une des techniques est la rétro-propagation. Un aller le long de n (figure 4) et un retour à rebours de n forment une *époque*. Une fois ces valeurs obtenues de manière satisfaisante, un nouveau vecteur $\mathbf{h}(0)$ peut être communiquer au réseau pour découvrir de manière originale, cette fois, les valeurs du vecteur de sortie $\mathbf{h}(N)$. De cette manière, l'apprentissage se fait sans supervision.

Références

[1] R. T. Q Chen, Y. Rubanova, J. Bettencourt et D. Duvenaud, Neural Ordinary Differential Equations, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, 2018.

[2] (a) E. Weinan, A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics, **5**(1) :1–11, 2017 ; (b) Y. Lu, A. Zhong, Q. Li et B. Dong, Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations, 2017 ; (c) E. Haber et L. Ruthotto, Stable Architectures for Deep Neural Networks. Inverse Problems, **34**(1) :014004, 2017 ; (d) L. Ruthotto et E. Haber, Deep Neural Networks Motivated by Partial Differential Equations, 2018.

[3] A. Burkov, The Hundred-Page Machine Learning Book, 2019.

[4] N. Piskounov, *Calcul différentiel et intégral*, Ellipses, Paris, 1993.

[5] P. Atkins et J. De Paula, *Chimie physique*, De Boeck Université, Bruxelles, 2008.

[6] T. Lepoint, chimieetjulia.org

[7] (a) I. Prigogine, *Non-Equilibrium Statistical Mechanics*, Monographs in Statistical Physics and Thermodynamics, Vol.1, Intersciences Publishers, New-York, 1962 ; (b) R. Lefever, G. Nicolis et P. Borckmans, The Brusselator : It Does Oscillate all the same, J. Chem. Soc. Faraday Trans. 1, 1988, 84(4), 1013-1023 ; (c) M. Ruth et B. Hannon, *The Brusselator*. In : Modeling Dynamic Biological Systems. Modeling Dynamic Systems. Springer, New York, NY, 1997.

[8] Sylvie Huet, Emmanuel Jolivet et Antoine Messéan, *La régression non-linéaire : méthodes et applications en biologie*, INRA Édition, Paris, 1992.

[9] J. C. Butcher, *Numerical Methods for Ordinary Numerical Equations*, John Wiley, Chichester, UK, 2016.

[10] R. Plato, *Concise Numerical Mathematics*, American Mathematical Society, 2003.

[11] (a) Agorithme du gradient ; (b) Une illustration est donnée sur le blog de C. Bordet

[12] K. Gibson, Neural Networks as Ordinary Differential Equations, 2019.

[13] A. M. Bradley, PDE-Constrained Optimization and the Adjoint Method, 2019.

[14] A. Lörke, F. Schneider, J. Heck et P. Nitter, Cybenko's Theorem and the capability of a neural network as function approximator, 2019

[15] K. Hornik, M. Stinchcombre et H. White, Multilayer Feedforward Networks Are Universal Approximators, Neural Networks, **2**(5), 359-366, 1989.

[16] M. Surtsukov, Neural Ordinary Differential Equations, 2019

[17] M. Nielsen, Neural Network and Deep Learning, Chap. 4, 2019.

[18] K. Hao, A Radical New Neural Network Design Could Overcome Big Challenges in AI, 2018.

[19] E. Dupont, A. Doucet et Y. W. Teh, Augmented Neural ODEs, 33rd Conference on Neural Processing System (NeurIPS 2019), Vancouver, Canada.

[20] J. Jia et A. R. Benson, Neural Jump Stochastic Differential Equations, 2019.