NEDO

March 15, 2024

1 Résoudre une EDO Neurale

1.1 Rappels de Calculs Différentiels

Supposons que l'on veuille simuler une population d'individus. En d'autres termes, si $N(t)$ est la population t, alors N vérifie

$$\frac{\partial}{\partial t}N(t) = KN(t), \text{ avec } K \in \mathcal{R}$$
$$\implies \quad N(t) = Ce^{Kt}, \text{avec } C \in \mathcal{R}$$

Il est souvent plus simple de décrire une fonction par sa dynamique que par sa formule explicite. Cela justifie l'utilité des équations différentielles.

Toutefois, la résolution de ce type d'équations n'est pas toujours évidente et de nombreux problèmes sous jasent apparaissent. Entre autres, il n'y a pas toujours existence et unité de la solution, ni stabilité globale de celle ci. Ou encore, si la solution existe, elle n'est pas toujours calculable explicitement. Par exemple, si on suppose que N vérifie

$$\left(\frac{\partial}{\partial t}N(t)\right)^3 + y^2 = N(t)y.$$

Une méthode usuelle est d'approcher la solution par la méthode d'Euler, c'est à dire, discrétiser le problème de la manière suivante

$$N(t + \Delta t) = N(t) + \sqrt[3]{N(t)y - y^2}\Delta t.$$

Cette méthode reste la plus usuelle pour résoudre ce genre de problèmes.

1.2 Motivations

Parmis tous les réseaux de neuronnes existant, ce sont les RNN qui sont les plus adapté pour traiter des données avec une dépendance dans le temps.

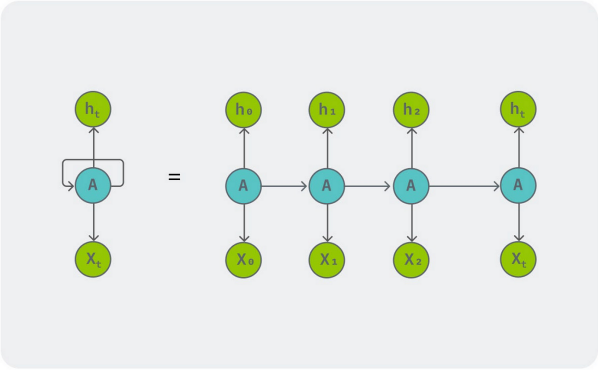

Figure 1: RNN

Alors, une idée naturelle pour réaliser de l'apprentissage profond (deep learning) est de rajouter un très grand nombre de couches au réseaux. Pour éviter les phénomènes de Gradient Vanishing/Exploding que nous ne traiterons pas ici, il est habituel d'utiliser des réseaux LSTM ou GRU.

Une idée plus novatrice est la suivante, supposons que l'on veuille paramétrer l'input state au temps $t+1$ en fonction de celui au temps t. En d'autres termes,

$$h_{t+1} = h_t + f(h_t, \theta_t) \tag{1}$$

On reconnait alors que (1) est une discrésation d'Euler de (2).

$$\frac{d}{dt}h(t) = f(h(t), t, \theta) = f_\theta(h(t), t) \tag{2}$$

Résoudre (2) a de nombreux avantages. Entre autres, on a unicité et existence de la solution et stabilité numérique.

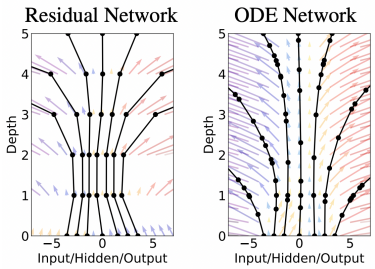

Figure 2: Caption

1.3 Cadre d'une NEDO

Dans le cadre d'un réseau de neurones, on veut évaluer la perte entre la prédiction estimée et le vrai résultat. Puis, réaliser une backpropagation pour mieux estimer les poids et les biais de chaque couche. La méthode la plus classique est de réaliser une descente de gradient de sorte que

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta L \tag{3}$$

Alors, tout l'enjeu ici est de calculer $\nabla_\theta L$. Une premiere approche serait d'uriliser des algorithmes de résolutions d'équations différentielles mais ils produisent une grande source d'erreurs. Cela s'explique par le fait qu'ils reposent sur des produits matriciels et des invrsion de matrices qui sont des méthodes très couteuse. C'est pourquoi, plutôt que le calculer directement, on applique la méthode de l'adjoint.

2 Résoudre une EDO "Neuronale"

Soit la fonction z qui décrit une transformation continue au-cours du temps t, pour $t \in [0, T]$. On chercher à étudier le comportement de z en fonction du temps, t. En d'autres termes, comment résoudre

$$\frac{\partial}{\partial t}z(t) = f_\theta(z(t), t) \tag{4}$$

Soit f_θ un réseau de neurone de paramètres θ, et L une fonction de coût. On interprète h_t comme une de couche au temps t de f_θ, on peut donc interpréter que il y une infinité de couches sur l'intervalle $[0, T]$ Alors, on a

$$z(T) = z(0) + \int_0^T f_\theta(z(t), t)dt \text{ où } z(0) = x,\ z(T) = \hat{y} \text{ notre prédiction.} \tag{5}$$

On se base sur des méthodes faisant appel à un calcul de gradient pour la backpropagation, en particulier on chercher à calculer

$$\frac{\partial}{\partial\theta}L(z(T),y,\theta).$$

Pour cela, on utilisera la méthode dite de l'adjointe.

$$\text{Montrons que si } a(t)=\frac{d}{dz(t)}L \implies \frac{d}{dt}a(t)=-a(t)\frac{\partial}{\partial z(t)}f_\theta(z(t),t). \tag{6}$$

Par la Chain Rule,

$$\frac{dL}{dh_t}=\frac{dL}{dh_{t+1}}\frac{dh_{t+1}}{dh_t}$$

.

Or ici le hidden state $z(t)$ est solution d'une EDO donc necessairement continue. Alors analogue continu de la Chain Rule,

$$\frac{dL}{dz(t)}=\frac{dL}{dz(t+\epsilon)}\frac{dz(t+\epsilon)}{dz(t)}=\frac{dL}{dT_\epsilon(z(t),t)}\frac{dT_\epsilon(z(t),t)}{dz(t)}. \tag{7}$$

Avec $z(t+\epsilon)=z(t)+\int_t^{t+\epsilon}f_\theta(z(t),t)dt=T_\epsilon(z(t),t)$.

$$\text{On obtient alors } a(t)=a(t+\epsilon)\frac{\partial}{\partial z(t)}T_\epsilon(z(t),t) \tag{8}$$

Par définiton de la dérivée :

$$\begin{aligned}
\frac{d}{dt}a(t)&=\lim_{\epsilon\to0^+}\frac{a(t+\epsilon)-a(t)}{\epsilon}\\
&=\lim_{\epsilon\to0^+}\frac{a(t+\epsilon)-a(t+\epsilon)\frac{\partial}{\partial z(t)}T_\epsilon(z(t),t)}{\epsilon}\\
&=\lim_{\epsilon\to0^+}\frac{a(t+\epsilon)-a(t+\epsilon)\frac{\partial}{\partial z(t)}\left[z(t)+\epsilon f_\theta(z(t),t)+\tau(\epsilon^2)\right]}{\epsilon}\\
&=\lim_{\epsilon\to0^+}\frac{a(t+\epsilon)-a(t+\epsilon)\left[Id+\epsilon\frac{\partial}{\partial z(t)}f_\theta(z(t),t)+\tau(\epsilon^2)\right]}{\epsilon}\\
&=\lim_{\epsilon\to0^+}\frac{-\epsilon a(t+\epsilon)\frac{\partial}{\partial z(t)}f_\theta(z(t),t)+\tau(\epsilon^2)}{\epsilon}\\
&=-a(t)\frac{\partial}{\partial z(t)}f_\theta(z(t),t)\quad(A).
\end{aligned}$$

Une idée naturelle est de généraliser (A) pour obtenir et gradient de θ et L, de t_0 à t_N.
On pose θ et t comme des états du réseau tels que

$$\frac{\partial}{\partial t}\theta(t)=0 \text{ et } \frac{d}{dt}t(t)=1.$$

On combine les 3 paramètres d'un coup pour former un augmented state.

$$\frac{d}{dt}\begin{bmatrix}z\\\theta\\t\end{bmatrix}(t):=f_{aug}([z,\theta,t])=\begin{bmatrix}f([z,\theta,t])\\0\\1\end{bmatrix}$$

Alors on définit

$$a_{aug}:=\begin{bmatrix}a\\a_\theta\\a_t\end{bmatrix} \text{ avec } a_\theta=\frac{d}{d\theta(t)}L,\ a_t=\frac{d}{dt(t)}L. \tag{9}$$

$$J(f_{aug}) \;=\; \frac{\partial f_{\text{aug}}}{\partial[z,\theta,t]} = \begin{bmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial t} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}(t) \tag{10}$$

On réutilise (A) appliqué à (6),

$$\frac{d}{dt}a_{aug}(t) = -\begin{bmatrix} a(t) & a_\theta(t) & a_t(t) \end{bmatrix} J(f_{aug}) = -\begin{bmatrix} a\frac{\partial}{\partial z}f & a\frac{\partial}{\partial \theta}f & a\frac{\partial}{\partial t}f \end{bmatrix}(t)$$

On retrouve alors (A) dans la première coordonnée de $\frac{d}{dt}a_{aug}(t)$ et on cherche le gradient de la deuxiement coordonnée.

$$\frac{d}{d\theta}L = a_\theta(t_0) = -\int_{t_N}^{t_0} \frac{\partial}{\partial \theta} f(z(t), t, \theta)dt. \tag{11}$$

Il ne reste alors plus qu'à optimiser le réseau.

3 Applications

3.1 Contexte

Dans cette partie, nous étudions l'article Neural-ODE for pharmacokinetics modeling de James Lu, Kaiwen Deng, Xinyuan Zhang, Gengbo Liu, et Yuanfang Guan.
Dans cet article, on utilise les méthodes de modélisation par Neural-EDO et on compare ces résultats avec les méthodes usuelles. Cette article est la premiere application des méthodes de NEDO.

3.2 Contexte et Enjeux

En recherche pharmaceutique, l'étude de la pharmacocinétique (PK) de chaque patient est fondamentale. On définit cela comme l'étude de la réaction du corps après l'administration d'une substance, pour toute la durée de l'exposition.

Naturellement, ce type de simulations, que l'on désigne par "PK", constitue un axe crucial dans l'industrie pharmaceutique. L'un des problèmes clés est que les modèles PK, construits selon les méthodes classiques des réseaux neuronaux récurrents (RNN), fonctionnent bien pour un dosage constant. Cependant, leur généralisation pour des dosages différents est très mauvaise. C'est pourquoi cet article utilise des Neural EDO qui se comportent bien mieux lors de la généralisation des modèles de prédictions.

Cette étude est la première à utiliser les Neural EDO pour simuler des PK.

3.3 Dataset et Méthodes

Déterminer le bon dosage et le moment approprié de l'administration pour chaque patient est un enjeu fondamental en recherche pharmaceutique, et cela a permis un grand nombre d'applications dans le développement des médicaments. De manière classique, on utilise les régressions linéaires (LR), la régularisation Lasso, et les forêts aléatoires (RF) pour générer ces modèles. Par exemple, on a réussi à déduire le bon dosage de remifentanil (un anesthésique) dans une population saine.

Dans cette étude, nous utilisons un ensemble de données de pharmacocinétique (PK) du trastuzumab emtansine (T-DM1) : un médicament contre le cancer du sein. Le PK de ce traitement est bien caractérisé et décrit par un modèle linéaire de PK. En pratique, le T-DM1 possède une fenêtre thérapeutique étroite et a été déterminée après des essais à petites doses sur les patients afin de déduire les meilleurs dosages. Deux modèles ont été établis : le premier avec un seul dosage toutes les 3 semaines (Q3W), et le deuxième avec un dosage hebdomadaire (Q1W).

En raison de la nature du traitement, l'ensemble de données de Q1W est beaucoup plus important que celui de Q3W. Ces différences nous permettront de déterminer dans quelle mesure l'extrapolation du réseau est efficace.

Les modèles suivants ont été utilisés.

1. Neural-ODE
2. LSTM Neural Network (outil classique)
3. LightGBM (version plus récente du XGBoost).

Finalement, la performance de chaque modèle est évaluée en testant sa capacité à prédire les concentrations PK chez les patients traités avec le régime opposé.

3.4 Résultats

Les réseaux neuronaux à équations différentielles ordinaires (Neural-ODE), LightGBM et LSTM présentent des performances similaires lorsque les données d'entraînement et de test proviennent des mêmes régimes de traitement. Cependant, les réseaux neuronaux à équations différentielles ordinaires surpassent les autres algorithmes lorsqu'il s'agit de généraliser à de nouveaux régimes.

L'avantage du modèle de réseau neuronal à équations différentielles ordinaires (Neural-ODE) peut résulter de sa capacité à traiter des points de données échantillonnés de manière inégale. Les modèles d'apprentissage profond alternatifs tels que les LSTM ont tendance à supposer un échantillonnage régulier. Neural-ODE intègre directement les données de dosage et de timing dans le modèle au stade du décodeur, ce qui contribue probablement également à la performance stable lors de l'adoption d'un modèle entraîné sur un régime de traitement différent à un autre régime de traitement.