Résumé

July 13, 2019

Il existe deux principales techniques pour traiter des données dépendantes du temps avec un réseau de neurones : les réseaux récurrents et les réseaux de temporels de convolutions qui appliquent une convolution 1D sur la dimension du temps.
J'ai implémenté les deux versions mais j'ai pour l'instant testé et tenté d'optimiser que le réseau récurrent. Je le trouve plus adapté dans le sens où on a juste besoin des valeurs au temps $t = t_0$, tandis qu'un réseau de convolution 1D nécessiterait que le médecin fasse lui même des réglages pendant un certain intervalle de temps $I = [t_0, t_n]$, puis fournisse au réseau les valeurs qu'il a réglé pendant cet intervalle afin de prédire les valeurs suivantes $t_{n+1}, ...$ Je ne sais pas ce que tu recherches précisement aussi comme utilisation et j'aimerais savoir si la deuxième option te parait aussi envisageable.

1 Réseau récurrent

Le réseau récurrent se présente comme suit. Pour faire court c'est une cellule contenant plusieurs fully-connected layers et qui sort à chaque itération sur le temps une prédiction et un état récurrent qui est actualisé à chaque pas de temps.
On considère ici un input $X \in \mathbb{R}^{N_{Pin} \times N_T}$ (plusieurs paramètres dépendant du temps) où N_{Pin} représente le nombre de paramètres d'entrée et N_T le nombre de time steps, i.e. on travaille sur un intervalle de temps $I_t = [t_0, \dots, t_{N_T}]$. De la même façon, on considère un output $Y \in \mathbb{R}^{N_{Pout} \times N_T}$ où N_{Pout} est le nombre de paramètres à la sortie du réseau. Alors, la structure vraiment basique pour ce type de réseau est la suivante :

$$\begin{aligned} h_n &= \text{ReLU}\left(W_{(x\ h)} x_n + W_{(h\ h)} h_{n-1} + b_{(h)}\right), \\ y_n &= W_{(h\ y)} h_n + b_{(y)}, \quad n \in \{1, \dots, N_T\} \end{aligned}$$

avec $h_0 = 0 \in \mathbb{R}^Q$ où Q est la dimension de l'état récurrent, $W_{(x\ h)} \in \mathbb{R}^{Q \times N_{Pin}}$, $W_{(h\ h)} \in \mathbb{R}^{Q \times Q}$ et $W_{(h\ y)} \in \mathbb{R}^{N_{Pout} \times Q}$ et $x_i \in \mathbb{R}^{N_{Pin}}$ pour $i = 1, \dots, N_T$ tels que $X = [x_1\ x_2\ \cdots\ x_{N_T}]$ (ce sont les mêmes notations que dans le cours de François Fleuret).

J'ai implémenté cette version et une version "gated" où en gros on fait une somme pondérée entre le nouvel état calculé et l'état précédent. La version "gated" donne de meilleurs résultats car elle permet d'éviter le problème du vanishing gradient. Pour rajouter un peu de complexité au réseau j'ai doublé chaque couche en rajoutant une fully-connected layer à chaque fois.

2 Test

Afin de tester le réseau, il a fallu créer un set de données qui se rapproche de la situation que l'on va traiter. Pour ça j'ai d'abord modéliser des paramètres $X \in \mathbb{R}^{N_{Pin} \times N_T}$ qui dans une application réelle représenterait par exemple les données physiologiques du patient, les paramètres hémodynamiques etc. Pour créer de telles paramètres qui dépendent du temps, j'ai résolu numériquement avec le schéma de Backward Euler, l'ODE de premier ordre suivante :

$$\dot{x} = Ax + b$$

où $A \in R^{N_{Pin} \times N_{Pin}}$ est une matrice que j'ai généré avec l'algorithme QR, de façon à ce qu'elle soit définie négative et très bien conditionnée (c'est pour faciliter la résolution de l'ODE avec Backward Euler), $b \in \mathbb{R}^{N_{Pin}}$. J'ai ensuite générer des outputs (doses à donner, RPM de l'ECMO etc.) qui sont fonctions des inputs tels que

$$Y = f(X) \in \mathbb{R}^{N_{Pin} \times N_T}$$

où $f : \mathbb{R}^{N_{Pin} \times N_T} \to \mathbb{R}^{N_{Pin} \times N_T}$ est une fonction qui applique paramètre par paramètre (donc ligne par ligne) une fonction aléatoirement choisie parmi les trois fonctions suivantes :

$$\begin{aligned} f_1 &= \alpha \cos(x) \\ f_2 &= \beta \sin(x) \\ f_3 &= \gamma \cos(x) + \delta \sin(x) \end{aligned}$$

où $f_i : \mathbb{R}^{N_T} \to \mathbb{R}^{N_T},\ i = 1, 2, 3$. À noter que dans ce cas particulier, $N_{Pin} = N_{Pout}$. Je génère donc ensuite un training set et un test set contenant plusieurs samples (plusieurs patients) avec des inputs X et les outputs Y fonctions de X. Je génère chaque sample avec une condition initiale différente pour ODE (et donc pour mes inputs). Ça correspond à des données physiologiques différentes par exemple donc ça permet de simuler plusieurs patients différents.

3 Résultats

En entraînant le réseau pour 100 epochs avec l'optimiseur Adam et une mse comme loss, j'obtiens les résultats suivants. Je Les prédictions sont les prédictions que je

fais à chaque pas de temps, ce n'est pas une prediction que je fais d'un seul coup à la fin. Je ne prédis donc pas une séquence à proprement parler mais vraiment des valeurs à chaque temps. Les figures que j'ai mises représentent un seul paramètre en fonction du temps (par exemple les rpm de l'ECMO). Elles sont assez bien mais certains paramètres sont moims bien representés.

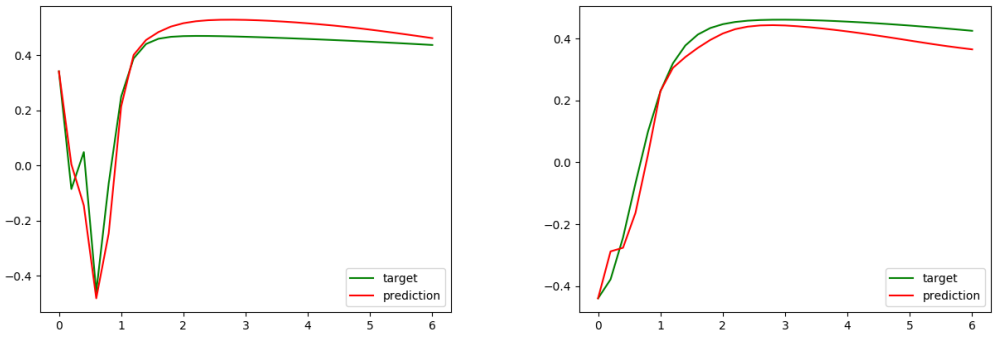

Figure 1: Quelques comparaisons prediction/target

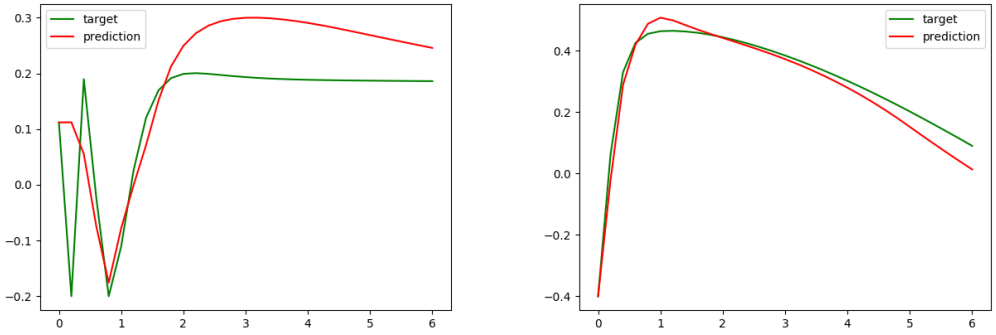

Figure 2: Quelques comparaisons prediction/target

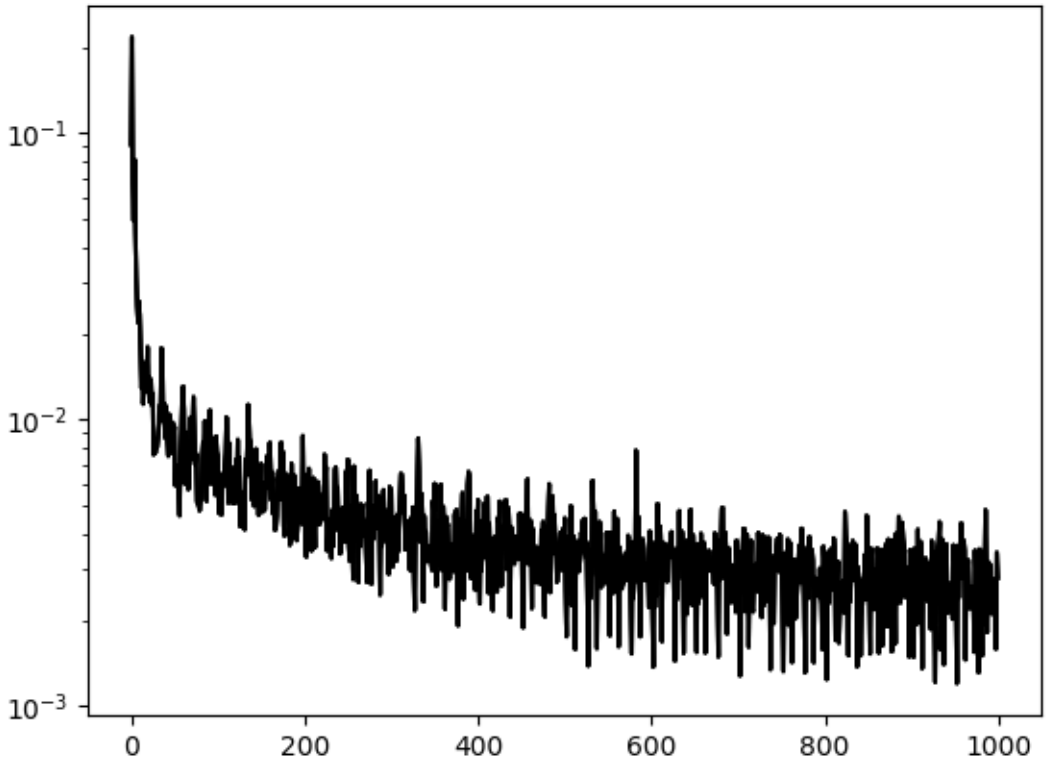

Figure 3: Exemple de la loss pour chaque mini batch (50 batches pour 500 samples) et pour plusieurs 100 epochs