Réseaux récurrents d'attention pour la régression de séries temporelles.

Victor PERRIER[1], Emmanuel LOCHIN[2], Jean-Yves TOURNERET[3], Patrick GÉLARD[4]

[1]ISAE-SUPAERO, TéSA, Toulouse
[2]ENAC, Toulouse
[3]Université de Toulouse, CNRS, IRIT, Toulouse
[4]CNES, Toulouse

victor.perrier@tesa.prd.fr, emmanuel.lochin@enac.fr, jean-yves.tourneret@enseeiht.fr, patrick.gelard@cnes.fr

Résumé – Cet article étudie une nouvelle architecture récurrente basée sur l'attention, plus légère et moins coûteuse en temps de calcul qu'un réseau d'attention global. Nous détaillons en quoi ce type d'architecture permet d'atteindre de meilleures performances que des réseaux récurrents plus classiques, dans le cas de la régression de séries temporelles. Nous montrons son intérêt pour la prédiction de l'état d'un réseau de communication, et plus particulièrement pour la détection de la congestion.

Abstract – This paper studies a new attention-based recurrent architecture, which is lighter and less computationally expensive than a global attention network. We detail how this type of architecture can achieve better performance than more classical recurrent networks for the regression of time series. We show its interest for the prediction of the state of a communication network, and more particularly for congestion detection.

1 Introduction

Contexte. Le contrôle de congestion TCP est un mécanisme essentiel pour le transport de l'information et le partage équitable des ressources de l'internet. Son principal objectif est d'éviter la congestion du chemin entre une source et une destination. En effet, la congestion est préjudiciable pour la performance du transfert, car elle se traduit par des pertes de paquets relatives au débordement de files au niveau des routeurs du chemin. Le contrôle de congestion est donc un algorithme qui, en fonction de l'état du réseau, prend une décision quant à la transmission d'un paquet. Cet état du réseau est déterminé grâce à la voie retour des acquittements des données transmises, qui permet de surveiller entre-autre : l'évolution du temps d'aller-retour, le nombre de paquets perdus, la gigue (variance de l'inter espacement des paquets), etc. Cependant, la modélisation et la prédiction du trafic internet est un problème complexe. Afin d'obtenir des algorithmes de contrôle de congestion de plus en plus performants, la communauté des réseaux s'est tournée vers l'apprentissage (machine learning) [1] [2]. Certaines méthodes de contrôle de congestion cherchent à estimer l'état interne du réseau afin d'envoyer les données de façon optimale [3]. Le but est alors d'estimer au mieux l'état de congestion du réseau à partir de séries temporelles décrivant l'évolution au cours du temps de paramètres clé du réseau (taille des files d'attente, évolution de la charge du goulot d'étranglement, ...).

État-de-l'art. Les modèles classiques de régression de séries temporelles peuvent être regroupées en plusieurs familles. La première, plus facilement explicable, regroupe les algorithmes de prédiction basés sur des méthodes paramétriques utilisant par exemple le filtre de Kalman ou des modèles du type ARIMA (AutoRegressive Integrated Moving Average) et ses améliorations. Des méthodes non paramétriques existent aussi comme celles basées sur la régression à base de vecteurs supports (support vector regression (SVR)) [4] ou sur les k plus proches voisins (k-Nearest Neighbors (kNN)). Une deuxième famille d'algorithmes de régression s'articule autour des réseaux de neurones, avec en particulier les méthodes d'apprentissage profond (deep-learning). Pour pouvoir traiter des séries temporelles, les réseaux de neurones récurrents (Récurrent Neural Networks (RNN)) ont tout d'abord été développés. Afin de pallier le problème de la disparition du gradient lors de l'entraînement [5], d'autres méthodes ont été introduites pour améliorer le principe des RNN qui sont basées sur des architecures plus originales comme les réseaux GRU [6] (Gated Recurrent Units), les réseaux LSTM (Long Short-Term Memory) [7] et les réseaux CNN (Convolution Neural Networks). Encore plus récemment, une nouvelle architecture appelée Attention a montré son intérêt pour plusieurs applications [8]. D'abord utilisée dans les tâches de traitement du langage naturel, cette architecture permet d'obtenir des modèles de prédiction plus précis, sous réserve d'avoir une puissance de calcul conséquente. On peut par exemple citer les travaux de [9] qui montrent que l'Attention se suffit à elle-même pour résoudre des tâches de traduction ou d'interprétation de textes.

Objectifs et contributions. L'objectif de ce travail est d'étudier une nouvelle architecture de réseaux de neurones basée sur l'Attention pour la régression de séries temporelles multivariées, avec une application à la prédiction de la congestion

des réseaux. Nous introduisons un nouveau modèle plus léger et aussi performant que l'Attention globale. Le travail ainsi réalisé permet de réduire le nombre de paramètres à estimer, ce qui résulte en un meilleur coût de calcul que celui associé à l'Attention. Plus précisément, au lieu d'avoir une complexité en $O(L^2)$ à chaque nouveau pas temporel, la nouvelle architecture permet d'obtenir une complexité de l'ordre de $O(L)$, L étant la taille de la série temporelle étudiée. Un autre apport de la méthode proposée est de montrer l'intérêt des réseaux d'Attention pour la régression de séries temporelles, puisqu'ils permettent d'estimer plus précisément certaines fonctions comme la fonction maximum et de mieux prendre en compte le passé de la série temporelle que l'on cherche à prédire.

2 Présentation du problème

Le problème de contrôle de congestion présenté dans [3] a été résolu à l'aide d'un estimateur défini par

$$\min_{i\in[t-L_1,t]} \boldsymbol{x}_i - \min_{i\in[t-L_2,t]} \boldsymbol{x}_i, \tag{1}$$

avec $L_1 << L_2$ et où $\boldsymbol{x}_i$ est la série temporelle des RTT (Round Trip Time), i.e., les temps d'aller-retour des paquets dans le réseau. Cet estimateur a montré de bonnes performances par rapport aux techniques utilisées actuellement telles que TCP cubic. L'objectif de cet article est de proposer une méthode d'apprentissage profond, fonctionnant en mode supervisé, permettant d'améliorer cet estimateur. L'idée est de construire une architecture neuronale permettant d'estimer des métriques utiles pour le contrôle de congestion comme (1). Afin de faire l'apprentissage en mode supervisé, nous proposons d'entraîner différents modèles (que nous allons expliciter) avec la fonction de coût

$$\frac{1}{L}\sum_{i=1}^{L} \|f(\boldsymbol{X}_{1:i}) - \boldsymbol{y}_i\|^2, \tag{2}$$

où $\boldsymbol{y}_i$ est un vecteur qui contient les métriques que l'on cherche à estimer à l'instant i, $\boldsymbol{X}_{1:i} \in \mathbb{R}^{i\times d}$ est une matrice contenant les observations jusqu'à l'instant i (dans (1) cette matrice est constituée des RTT, mais on peut aussi considérer d'autres observations complémentaires) et f est la fonction définie par le réseau de neurones que l'on cherche à entraîner.

3 Architectures neuronales existantes

3.1 Réseaux LSTM

Cette section considère l'exemple des réseaux LSTMs. Cependant, le même raisonnement pourrait être appliqué à d'autres structures (RNNs en général et CNNs) avec des résultats similaires. Nous avons remarqué que certaines tâches de régression, comme l'estimateur présenté dans (1), ne pouvaient être prédites avec précision avec des réseaux de neurones récurrents. En effet nous remarquons dans la figure 3b, qu'une prédiction effectuée avec un réseau LSTM détecte bien les maxima de la série temporelle, mais n'arrive pas à utiliser cette information pour les prochains pas temporels. La régression par LSTM semble approcher la série temporelle par une fonction linéaire par morceaux afin de minimiser l'erreur moyenne de prédiction. Ce type de réseau peine à utiliser une information spécifique contenue dans le passé de l'instant d'intérêt. Même si cette information était disponible, le vecteur caché utilisé par le réseau LSTM aurait du mal à accueillir toute l'information contenue dans la fenêtre temporelle d'intérêt.

Cette observation est donc une motivation pour créer un modèle capable d'estimer des fonctions comme le minimum et le maximum d'une fenêtre glissante issue d'une ou de plusieurs séries temporelles. Dans la suite de cet article, nous montrons que des réseaux utilisant l'Attention peuvent estimer avec une meilleure précision ces maxima et minima.

3.2 Attention

Afin d'expliquer l'architecture de l'Attention, nous introduisons les notations suivantes :

Notations	
Symbole	Signification
d	dimension des vecteurs de la série temporelle
L	taille de la série temporelle
M	nombre de couches dans l'architecture d'attention
$\mathbf{X} \in \mathbb{R}^{L\times d}$	série temporelle multivariée
$\mathbf{W}_k \in \mathbb{R}^{d\times d}$ $\mathbf{W}_q \in \mathbb{R}^{d\times d}$ $\mathbf{W}_v \in \mathbb{R}^{d\times d}$	matrices de paramètres de l'attention
$\mathbf{P} \in \mathbb{R}^{L\times d}$	matrice d'encodage de position ajoutée à $\mathbf{X}$

L'Attention est un mécanisme de traitement de séries temporelles construit à l'origine pour effectuer des tâches de traduction du langage. Cependant, il peut facilement s'étendre à d'autres domaines comme la régression de séries temporelles. L'Attention permet de définir une matrice de poids (dont la somme de chaque ligne vaut 1) qui nous indique à quel point les données de notre série temporelle sont connectées comme illustré dans la figure 1.

FIGURE 1 – Exemple d'attention pour l'interprétation d'une phrase. Une ligne pointillée représente un lien faible entre les mots tandis qu'une ligne continue correspond à un lien fort.

Le mécanisme d'Attention, en lui-même, n'est pas sensible à la position relative des éléments de la série temporelle $\boldsymbol{X}_i, i =$

$1, ..., L$. Il utilise une matrice de position $\boldsymbol{P} = (p_{i,j})$, avec $i, j \in [1, L] \times [1, d]$, définie de la manière suivante [9] :

$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{100^{2j/d}}\right) & \text{si } j = 2n, n \in \mathbb{N} \\ \cos\left(\frac{i}{100^{2j/d}}\right) & \text{si } j = 2n+1, n \in \mathbb{N} \end{cases}$$

Cette matrice permet de définir la position d'un vecteur dans une série temporelle, comme une horloge peut définir le moment de la journée avec trois aiguilles pour les secondes, minutes et heures. Le fait d'utiliser des sinus et cosinus pour $p_{i,j}$ permet d'obtenir une information comprise entre -1 et 1. Une couche d'Attention peut ensuite se définir comme suit :

$$\text{ATTENTION}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V},$$

avec

$$\text{softmax}(\boldsymbol{X})_{i,j} = \frac{e^{\boldsymbol{X}_{i,j}}}{\sum_{k=0}^{d} e^{\boldsymbol{X}_{i,k}}},$$

qui dépend de variables que nous allons définir. Dans l'exemple de la figure 1, où l'on cherche à avoir de l'information sur les liens entre les éléments de la série temporelle, on a $\boldsymbol{K} = \mathbf{W}_k\boldsymbol{X}$, $\boldsymbol{Q} = \mathbf{W}_q\boldsymbol{X}$ et $\boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{X}$, où $\mathbf{W}_k$, $\mathbf{W}_q$ et $\mathbf{W}_v$ sont des matrices de paramètres qui sont déterminées pendant la phase d'entraînement. Le résultat de cette auto-Attention est une combinaison linéaire des éléments de la matrice $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t)^T$.

Les rôles des matrices $\mathbf{W}_q$, $\mathbf{W}_k$ et $\mathbf{W}_v$ s'inspirent du language SQL, et représentent respectivement des requêtes (queries), des clés (keys) et des valeurs (values). Le duo de matrices $\mathbf{W}_q$, $\mathbf{W}_k$ va ainsi permettre au modèle de voir quels éléments du passé sont utiles pour prédire notre résultat. Par exemple, si on applique l'auto-attention à une matrice $\boldsymbol{X}$ de RTT avec $d = 1$, et $\mathbf{W}_q = \mathbf{W}_k = 1$, le résultat de de l'opération softmax nous donne une matrice de $\mathbb{R}^{L\times L}$ avec des lignes proches de 0, sauf la ligne correspondant à l'indice du maximum de $\boldsymbol{X}$ dont les éléments sont proches de 1. Ainsi si on choisit aussi $\mathbf{W}_v = 1$, le résultat de l'attention est une matrice dont les éléments sont des approximations du maximum de $\boldsymbol{X}$. Si on s'intéresse au minimum de $\boldsymbol{X}$ (au lieu du maximum), il suffit de choisir $\mathbf{W}_k = -1$. Il est également possible de définir des requêtes plus compliquées, comme par exemple demander le moment où s'est produit ce maximum des RTTs (si $\boldsymbol{X}$ contient aussi l'information du temps (avec $d = 2$)) : Il suffit de choisir $\mathbf{W}_q = \mathbf{W}_k$ comme la projection de $\boldsymbol{X}$ sur l'axe des RTTs, et $\mathbf{W}_v$ comme la projection de $\boldsymbol{X}$ sur l'axe sur l'axe des temps.

Malgré leurs performances remarquables, les réseaux d'Attention sont difficiles à utiliser à cause de leur taille massive (le modèle GPT-3 crée par OpenAI pour l'interprétation de textes possède 175 milliards de paramètres), et des entraînements longs. En effet, pour appliquer un modèle d'Attention à une série temporelle, il faut l'appliquer à chaque pas temporel. Ainsi, à l'étape $t \in \{1, ..., L\}$, la complexité du calcul est $\mathcal{O}(t^2)$ à cause du produit matriciel. Pour traiter une série temporelle de longueur L, la complexité est donc de l'ordre de $\mathcal{O}(t^3)$ (nous ne pouvons pas nous servir du calcul effectué à l'étape $t-1$ pour faciliter la tâche à cause des couches non-linéaires), alors que des méthodes comme les réseaux LSTMs ont une complexité de l'ordre de $\mathcal{O}(t)$. Ce temps de calcul est une motivation pour trouver une nouvelle architecture aussi performante et plus rapide, ce qui est l'objectif de cet article.

4 Nouvelle architecture

Afin de pallier les défauts des deux architectures des réseaux LSTM et de l'attention, nous proposons une nouvelle architecture hybride et étudions sa performance. Cette architecture est définie comme suit :

- 1) nous concaténons la matrice d'observation $\boldsymbol{X}$ avec la matrice de position $\boldsymbol{P}$.

$$\mathbf{X}_p = [\mathbf{X}, \mathbf{P}]$$

- 2) nous appliquons une couche de réseaux LSTM.

$$\mathbf{H_i} = \mathbf{LSTM}(\mathbf{x}_1, ...\mathbf{x}_i) \in \mathbb{R}^{J\times d}$$

ou J est la taille du vecteur produit par le réseau LSTM, à choisir par l'utilisateur.

- 3) nous appliquons le réseau d'Attention qui à partir de la matrice $\boldsymbol{H}_i$ va déterminer quels sont les éléments passés les plus importants.

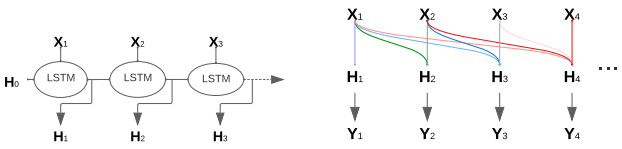
$$\mathbf{Y}_i = \mathbf{ATTENTION}(\mathbf{W}_q\mathbf{H}_i, \mathbf{W}_k\mathbf{X}_{1;i}, \mathbf{W}_v\mathbf{X}_{1;i})$$

L'idée de cette architecture est de ne pas utiliser l'auto-attention qui est trop coûteuse en temps de calcul, mais de générer grâce à un réseau LSTM le vecteur qui va nous donner les requêtes (J est alors le nombre de requêtes).

- 4) nous appliquons une couche non linéaire (fonction d'activation RELU), comme souvent pour l'apprentissage profond.

$$\mathbf{Z} = \mathbf{FeedForward}(\mathbf{Y})$$

Nous pouvons itérer ce processus en appliquant les étapes 2), 3) et 4) pour chacune des M couches du réseau.



(a) La première étape consiste à créer les vecteurs $\mathbf{H}_i$

(b) La deuxième étape utilise l'attention pour savoir quels $\mathbf{X}_i$ utiliser pour produire $\mathbf{Y}_i$

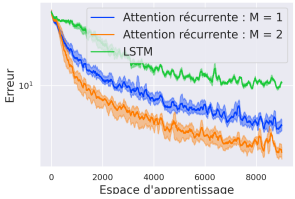
FIGURE 2 – Deux étapes de l'architecture proposée.

5 Résultats

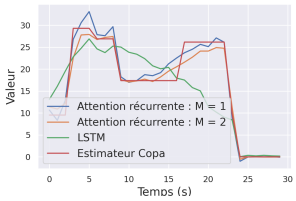
Pour tester la nouvelle architecture, nous allons tout d'abord mesurer sa capacité à prédire un maximum et à pouvoir chercher l'information dans le passé. Ensuite, nous aborderons un cas d'application concret.

5.1 Trouver un Maximum

Comme expliqué dans 3.1, un estimateur simple de la charge des files d'attente est de la forme (1). Nous essayons donc d'approcher cet estimateur avec la structure hybride proposée pour $L_1 = 5$ et $L_2 = 30$. Pour entraîner les réseaux de neurones, nous avons choisi un pas d'apprentissage de 0.001 et une descente de gradient de type ADAM. La série temporelle est générée aléatoirement à chaque étape d'entraînement selon une distribution normale, pour éviter le sur-apprentissage.



(a) Erreurs d'apprentissage en fonction du temps. Les aires représentent les erreurs maximales et minimales pour 10 modèles.



(b) Prédiction d'un réseau LSTM et du nouveau mécanisme d'attention pour l'estimateur (1).

FIGURE 3 – Estimation de (1) avec l'architecture proposée.

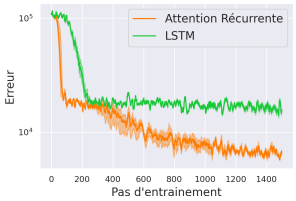
Nous observons sur la figure 3a que la nouvelle architecture, en plus d'apprendre plus vite à estimer la fonction f, peut apprendre à estimer la différence de deux minima avec plus de précision qu'un réseau LSTM qui atteint un plateau lors de l'apprentissage. Bien évidemment, cela reste une tache artificielle et le réseau créé est optimisé pour cette dernière. Nous remarquons également que plus le réseau est profond (plus M est grand), plus il apprend vite à utiliser des éléments du passé.

5.2 Application aux files d'attente

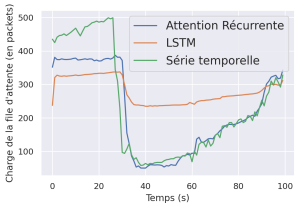
Cette partie considère un cas d'application concret : la prédiction du niveau de remplissage d'une file d'attente au niveau d'un goulot d'étranglement du chemin. Pour cela, nous avons à disposition la série temporelle des RTT ainsi que le moment d'émission de ces paquets. Nous avons recueilli ces données grâce à des outils d'émulation. L'entraînement se fait avec un pas d'apprentissage de 0.0005 et à nouveau l'optimiseur ADAM.

Nous remarquons dans la figure 4a qu'un plateau est atteint pour un apprentissage avec un réseau LSTM, alors que l'Attention permet d'apprendre assez rapidement la relation entre les données et la charge du réseau.

Nous constatons aussi que la bonne performance du nouvel algorithme est similaire à celle obtenue avec un réseau d'Attention global. Cependant, l'architecture proposée permet un entrainement plus rapide. Nous constatons enfin que pour cet exemple, il n'est pas utile d'augmenter le nombre de couches du réseau M.



(a) Erreur d'apprentissage en fonction du temps. Les aires représentent les erreurs maximale et minimale pour 10 modèles entraînés.



(b) Prédiction d'un réseau LSTM et du nouveau mécanisme d'attention pour la tâche de l'estimation du niveau de charge d'un goulot d'étranglement.

FIGURE 4 – Résultats pour le cas concret d'estimation de la charge d'une file d'attente avec l'architecture proposée.

6 Conclusion

Nous avons présenté dans cet article une nouvelle architecture hybride d'apprentissage profond utilisant un réseau d'Attention et un réseau de neurones récurrents, pour la régression de séries temporelles. Cette architecture offre un compromis intéressant entre précision et temps de calcul. L'application à la détection de congestion de files d'attente semble prometteuse.

Références

[1] K. Winstein and H. Balakrishnan, "TCP ex machina : Computer-generated congestion control," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 123–134, 2013.

[2] M. Dong et al., "PCC : Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX NSDI Conf.*, 2015, pp. 395–408.

[3] A. Venkat and H. Balakrishnan, "Copa : Practical Delay-Based congestion control for the internet," in *Proc. 15th USENIX NSDI Conf.*, Renton, WA, Apr. 2018, pp. 329–342.

[4] M. Awad and R. Khanna, "Support vector regression," in *Efficient learning machines*, pp. 67–80. Springer, 2015.

[5] S. Hochreiter et al., *Gradient flow in recurrent nets : the difficulty of learning long-term dependencies*, 2001.

[6] J. Chung et al., "Gated feedback recurrent neural networks," in *Proc. Int. Conf. on Machine Learning*, 2015, pp. 2067–2075.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] A. de Santana Correia and E. L. Colombini, "Attention, please ! a survey of neural attention models in deep learning," *Artificial Intelligence Review*, pp. 1–88, 2022.

[9] A. Vaswani et al., "Attention is all you need," in *Proc. Conf. Advances in neural information processing systems*, 2017, vol. 30.