Filter Bank-Embedded MRNN and MLSTM

Uladzimir Charniauski
University of Connecticut
uladzimir.charniauski@uconn.edu

Abstract

The MRNN/MLSTM networks have significantly improved the ability of recurrent networks to handle the long-range dependence data with their novel long-memory filter component, whose weights decay at a polynomial rate. Signal processing techniques, such as filter banks, have been effectively used in time series analysis to extract relevant features on varying frequency scales. We extend the definition of memory-augmented by requiring the model to process the data through multiple long-memory filters. To support our motivation, we demonstrate that this approach uplevels the original ability of original MRNN/MLSTM to process the long-memory sequences by allowing these networks to learn the best combination of filters to use for data processing.

1 Introduction

Memory-Augmented Recurrent Neural Network (MRNN) and Long-Short-Term-Memory (MLSTM) [4] proved their effectiveness in forecasting time series with long-term dependency. These networks utilize the fractionally integrated processes to handle the long-range dependence data via special singular filter components embedded in their architecture. However, both MRNN and MLSTM only learn one memory filter that only enables only one way for a network process to extract long-term dependence from the input. This limitation potentially requires a methodology that would allow an expansive search of the best filter combination among an initially given collection of memory filters. Filter banks technique enables a broader and more flexible approach by allowing the network to learn from multiple memory filters simultaneously. The existing research mostly focuses on implementing the filter banks into the CNN for improving various tasks traditionally involving the application of this neural network, such as texture classification [1] or speech recognition [3]. The scarcity of existing literature studying the behavior of recurrent networks has spurred the initiation of this study.

This paper explores the ability of MRNN/MLSTM to extract long-memory via filter banks by processing the data through multiple memory filters. We incorporate the filter banks composed of multiple originally developed long-memory filter components into MRNN and MLSTM and introduce the Filter Bank-Embedded Memory-Augmented RNN (F-MRNN) and Filter Bank-Embedded Memory-Augmented LSTM (F-MLSTM), whose filters are controlled by unique learnable memory parameters d. Modifying MRNN and MLSTM with the collection of memory filters endorses them with an ability to approximate the fractional differencing effect with multiple filters that capture different levels of memory and frequency components simultaneously. We then perform the numerical studies on several long-range time series datasets to illustrate the advantages of our proposed models.

2 Methods

This section will describe the proposed modifications for MRNN and MLSTM networks. Here the presented mechanism is a combination of learnable filters that process the same input through uniquely generated selections of memory parameters guaranteeing the memory elongation effect for

Preprint. Under review.

each component. In Sections 2.1 and 2.2, we modify the MRNN and MLSTM by the proposed filter bank structure, respectively.

2.1 F-MRNN

Long memory pattern is introduced via memory filter:

$$F(x^{(t)}; d) = ((I - \mathcal{B})^d - I)x^{(t)},$$

for the p_x-dimensional input $\{x^{(t)}\} \in \mathbb{R}^{p_x}$. For the ith element, the memory filter is the following:

$$F(x^{(t)}; d)_i = ((1 - B)^{d_i} - 1)x_i^{(t)} = \sum_{i=1}^{\infty} w_j(d_i)x_i^{(t-j+1)},$$

where $w_j(d) = \Gamma(d + j)/[j!\Gamma(d)] = \prod_{i=0}^{j-1} \frac{i-d}{i+1}$.

In F-MRNN, we introduce filter banks through a weighted sum of long memory filters within the memory hidden unit:

$$m^{(t)} = \tanh(W_{mm}m^{(t-1)} + \sum_{k=1}^{n_f} W_{mf}^{(k)} F^{(k)}(x^{(t)}; d) + b^{(m)})$$

,

where n_f denotes the number of filters. This way, the unit $m^{(t)}$ models long memory patterns in time series with multiple learnable memory filters.

For $i \in \{1, \ldots, p_x\}$, $t \in \{1, \ldots, T\}$, $h^{(t)}, m^{(t)} \in \mathbb{R}^q$, $d^{(t)}, x^{(t)} \in \mathbb{R}^{p_x}$, and σ is a sigmoid activization function, the procedure of MRNN is given below:

$$\begin{cases} l^{(t)} = \|y^{(t)} - z^{(t)}\|^2 \\ z^{(t)} = g(W_{zh}h^{(t)} + W_{zm}m^{(t)} + b_z) \\ h^{(t)} = \tanh(W_{hh}h^{(t-1)} + W_{hx}x^{(t)} + b_h) \\ F(x^{(t)}; d)_i = \sum_{i=1}^{K} w_j(d_i^{(t)})x_i^{(t-j+1)} \\ d^{(t)} = \frac{1}{2}\sigma(W_d[d^{(t-1)}h^{(t-1)}m^{(t-1)}x^{(t)}] + b_d) \\ m^{(t)} = \tanh(W_{mm}m^{(t-1)} + \sum_{k=1}^{n_f} W_{mf}^{(k)} F^{(k)}(x^{(t)}; d) + b^{(m)}) + b^{(m)}) \end{cases}$$

The elements of the memory parameter $d = (d_1, \ldots, d_q)^T$ are restricted to the range $(0, 0.5)$ to model the long-memory effect. This parameter also depends on other variables at each round t, giving it a notation $d^{(t)}$.

2.2 F-MLSTM

We introduce the MLSTM cell state update by adding the long memory filter to it:

$$(I - \mathcal{B})^d c_t = i \odot \tilde{c},$$

where the memory parameter d depends on other variables as in MRNN.

We propose a filter bank-embedded cell update in our F-MLSTM modification as:

$$\sum_{k=1}^{n_f} W_{cf}^{(k)} (I - \mathcal{B})^{d_k} c^{(t)} = i \odot \tilde{c}^{(t)},$$

with d_k are the memory parameters corresponding to kth filter bank for $k \in [1, n_f]$.

The designed underlying network remains

$$z^{(t)} = g(W_{zh}h^{(t)} + b_z) + \varepsilon^{(t)}, \text{for} t \in \mathbb{Z},$$

where $h^{(t)}$ is a hidden unit produced by the MLSTM cell.

For $t \in \{1, \ldots, T\}$, we introduce the F-MLSTM procedure as the following:

$$\begin{cases} d^{(t)} = \frac{1}{2}\sigma(W_d[d^{(t-1)}, h^{(t-1)}, x^{(t)}] + b_d) \\ i^{(t)} = \sigma(W_{ih}h^{(t-1)} + W_{ix}x^{(t)} + b_i) \\ o^{(t)} = \sigma(W_{oh}h^{(t-1)} + W_{ox}x^{(t)} + b_o) \\ \tilde{c}^{(t)} = \tanh(W_{ch}h^{(t-1)} + W_{cx}x^{(t)} + b_c) \\ \sum_{k=1}^{n_f} W_{cf}^{(k)}(I-\mathcal{B})^{d_k}c^{(t)} = i \odot \tilde{c}^{(t)} \text{ with each } d_k = d_k^{(t)} \\ h^{(t)} = o^{(t)} \odot c^{(t)} \end{cases}$$

To implement the model, the infinite summation is truncated to $(I-\mathcal{B})^d c_t$ at a lag K:

$$c_i^{(t)} = \sum_{k=1}^{n_f} W_{cf}^{(k)} \sum_{i=1}^{K} w_j(d_i^k) c_i^{(t-j)} + i^t \tilde{c}^{(t)},$$

where $c_i^{(t)}$ is the i-th element of $c^{(t)}$.

3 Experiments

This section our numerical experiments on several time series datasets. We select four originally introduced long-memory time series datasets and compare the models using appropriate statistical metrics.

We use PyTorch to implement all the neural networks. We use the Adam algorithm with learning rate 0.01 for optimization. Due to resource constraint, we are unable to conduct the evaluation of considered networks on the same scale as in [4]. Therefore, the optimization stops when the loss function drops by less than 10^{-4}, or has been increasing for 100 steps or has reached 500 steps in total. We also consider $n_f = \{2, 4\}$ for the filter bank-embedded models.

Considering the non-convexity of the optimization and the limitness of our computational powers, we only initialize with 10 different random seeds and arrive at 10 different trained models for each model. We refer to the distribution of these 10 results as the *overall performance*, which reflects the expectations for a locally optimal model, and best of them as the *best performance*, which illustrates the approximate outcomes of the globally optimal model. MSE is the target loss function for training. We perform one-step rolling forecasts and calculate RMSE on a test dataset.

3.1 Datasets

We compare our models with the baselines on one synthetic dataset and three real datasets. We split the datasets into training, validation and test sets, and report their lengths below using notation $(n_{train} + n_{val} + n_{test})$.

3.1.1 ARFIMA series

. We use a series length 4001 (2000+1200+800) using the model generated [4] with the model $(1 - 0.7B + 0.4B^2)(1 - B)^{0.4}Y_t = (1 - 0.2B)\varepsilon_t$.

3.1.2 Dow Jones Industrial Average (DJI)

The raw dataset containing DJI daily closing prices from 2000 to 2019 obtained from Yahoo Finance.

3.1.3 Metro interstate traffic volume

The raw data set contains the hourly traffic volume on Interstate 94 westbound for MN DoT ATR station 301, roughly midway between Minneapolis and St Paul, MN, obtained from the MN Department of Transportation.

3.1.4 Tree ring

Tree ring datasets that contains 4351 (2500 + 1000 + 850) tree ring measures of a pine from Indian Garden, Nevada Gt Basin obtained from R package *tsdl* [2].

3.2 Results

	ARFIMA	DJI(x100)	Traffic	Tree
RNN	1.2733	0.00269	356.39	0.2934
	(0.2396)	(0.00013)	(6.071)	(0.0096)
LSTM	1.3391	0.00260	350.65	0.2918
	(0.2075)	(0.00016)	(9.103)	(0.0106)
MRNN	1.1939	0.00258	353.50	**0.2843**
	(0.1719)	(0.00015)	(11.281)	(0.0070)
MRNNF	1.2102	**0.00254**	349.60	0.2854
	(0.1624)	(0.00015)	(6.925)	(0.0061)
MLSTM	1.2733	0.00257	351.26	0.2877
	(0.1684)	(0.00015)	(5.298)	(0.0054)
MLSTMF	1.2950	0.00257	349.80	0.2879
	(0.1510)	(0.00014)	(5.031)	(0.0055)
F-MRNN2	**1.1788**	0.00264	348.17	0.2861
	(0.1426)	(0.00012)	(8.561)	(0.0006)
F-MRNNF2	1.3097	0.00262	351.65	0.2902
	(0.2364)	(0.00011)	(10.665)	(0.0093)
F-MLSTM2	1.2036	0.00264	351.26	0.2861
	(0.1820)	(0.00011)	(8.430)	(0.0064)
F-MLSTMF2	1.3492	0.00268	351.16	0.2907
	(0.1754)	(0.00011)	(5.256)	(0.0063)
F-MRNN4	1.1749	0.00270	347.77	0.2853
	(0.1784)	(0.0002)	(8.788)	(0.0068)
F-MRNNF4	1.2427	0.00257	**346.01**	0.2867
	(0.1942)	(0.00017)	(6.707)	(0.0075)
F-MLSTM4	1.2756	0.00267	351.98	0.2889
	(0.2280)	(0.00011)	(7.811)	(0.0802)
F-MLSTMF4	1.289	0.00262	353.66	0.2877
	(0.1553)	(0.00012)	(6.073)	(0.0046)

Table 1: Overall Performance in terms of RMSE. Average RMSE and the Standard Deviation (in brackets) are reported. The best result is highlighted in **bold**

	ARFIMA	DJI(x100)	Traffic	Tree
RNN	1.0469	0.00244	345.41	0.2797
LSTM	1.0371	0.00243	338.31	0.2781
MRNN	1.0385	0.00241	338.23	**0.2772**
MRNNF	1.0554	0.00241	339.37	0.2781
MLSTM	1.0330	0.00241	341.13	0.2780
MLSTMF	1.0326	0.00243	341.50	0.2784
F-MRNN2	**1.0326**	0.00245	337.27	0.2782
F-MRNNF2	1.0514	0.00242	**333.97**	0.2797
F-MLSTM2	1.0467	0.00242	342.99	0.2793
F-MLSTMF2	1.0489	0.00250	341.82	0.2792
F-MRNN4	1.0363	0.00243	337.33	0.2786
F-MRNNF4	1.0489	**0.00241**	338.10	0.2796
F-MLSTM4	1.0476	0.00245	340.76	0.2798
F-MLSTMF4	1.0517	0.00243	346.34	0.2795

Table 2: Best performance in terms of RMSE.

Table 1 reports the resulting RMSE metrics for the overall performance of one-step forecasting. We can see that fMRNN2 and fMRNNF4 give the smallest average RMSE among all the models for ARFIMA and Traffic datasets, respectively. On the other hand, the traditional MRNN and MRNNF still achieve the best RMSE results for DJI and Tree datasets, slightly outpacing their filter-embedded variants. This occurrence is possible because the structures of F-MRNN(F)/F-MLSTM(F) models are more complex than of their standard counterparts, which may not always align well with the structure of the dataset, thus causing higher loss rates. The considered F-MLSTM(F) models does not always show an obvious advantage over MLSTM across the datasets. This could be due to their misalignment with the dataset structure or increased difficulties in training of these models from the presence of multiple filters.

Table 2 displays the best model performance in terms of RMSE for all considered datasets. The performances of F-MRNN networks are better than others on ARFIMA, DJI and Traffic data, whereas MRNN and MRNNF outperform all of the other models on a Tree dataset. These results show a pattern similar to that present in Table 1. We also observe, for most of the cases, that the standard memory-augmented networks are competent in achieving the global minimum as their filter bank-embedded counterparts.

4 Conclusion

This paper introduces Filter Bank-Embedded Memory-Augmented RNN (F-MRNN) and Filter Bank-Embedded Memory-Augmented LSTM (F-MLSTM) that extract the long-memoryness via multiple filters. Having different numbers of filters, these methods proved themselves to be advantageous in modeling long-memory sequences only on some of presented datasets, while slightly underperforming in the other ones. We should note that due to the limited computational resources constraint, we are unable to conduct the experiments on a large scale and present more meticulous results in assessing the performance of our models. However, we believe that the limiting computational power may be a major factor for the underperformance of our filter-embedded networks. Therefore, we leave testing our models on larger number of epochs and random seeds for the future numerical work. Furthermore, F-MRNN and F-MLSTM with dynamic d are increasingly time consuming compared to the original models, with the computational speed slowing down as the number of filters increase. This issue was also raised in [4] for MRNN and MLSTM, and finding solutions for faster optimization approaches is a living task for future studies of these networks. Lastly, we construct the filter banks representation in our models as a sum of multiple long-memory filters multiplied by unique learnable weights, and it is interestring to explore other architectures for filter bank-embedded models that may substantially improve the model sensitivity to the exhibited long-memory dependencies in the data.

References

[1] Vincent Andrearczyk and Paul F. Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters*, 84:63–69, 2016.

[2] Rob J Hyndman and Yangzhuoran Yang. *tsdl: Time Series Data Library*, 2018. R package version 0.1.0.

[3] N. Zeghidour, D. Grangier, N. Usunier, and E. Dupoux. Locally normalized filter banks applied to deep neural-network-based robust speech recognition. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 669–673, 2016.

[4] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do rnn and lstm have long memory? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11365–11375. PMLR, 2020.