Summary of Common Sequence Mapping Methods

This article aims to summarize and categorize common sequence mapping methods.

General Definition of Sequence Mapping

Consider a sequence mapping $f: \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, or $\mathbf{Y} = f(\mathbf{X})$:

$$\begin{bmatrix} \mathbf{y}_1^\top \\ \vdots \\ \mathbf{y}_n^\top \end{bmatrix} = \mathbf{Y} = f(\mathbf{X}) = f\left(\begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}\right)$$

$$\mathbf{y}_m = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)_m$$

In particular, we consider causal mapping:

$$\mathbf{y}_m = f(\mathbf{x}_1, \ldots, \mathbf{x}_m)_m \triangleq f_m(\mathbf{x}_1, \ldots, \mathbf{x}_m),$$

where

$$f_m : \mathbb{R}^{m\times d} \to \mathbb{R}^d$$

A common example of a causal mapping is a language model.

Since a general mapping can be obtained by two causal mappings, for example:

$$y_m = f_m(x_1, \ldots, x_m) + \bar{f}_{n-m}(x_{m+1}, \ldots, x_n)$$

We will only consider causal mapping in the following text.

Memory-Based Sequence Mapping

Inspired by RNN, we construct sequence mapping using memory:

(old version, for reference)

- memory $\mathbf{m}_t \in \mathbb{R}^{k\times d}$;
- forget gate $\mathbf{f}_t \in \mathbb{R}^{k\times ?}$;
- input gate $\mathbf{i}_t \in \mathbb{R}^k$;
- input state $\mathbf{u}_t \in \mathbb{R}^d$;
- output gate $\mathbf{o}_t \in \mathbb{R}^d$;

(new version)

- memory state $\mathbf{m}_t \in \mathbb{R}^{k\times d}$;
- forget state $\mathbf{f}_t \in \mathbb{R}^{k\times ?}$;
  - may be shock gate?
- expand state $\mathbf{e}_t \in \mathbb{R}^k$;
- input state $\mathbf{i}_t \in \mathbb{R}^d$;
- shrink state $\mathbf{s}_t \in \mathbb{R}^k$;

At each time t:

Input state and expand state are used to calculate the new memory $\bar{\mathbf{m}}_t = \mathbf{e}_t \mathbf{i}_t^\top$;

Then update using the following equation ($\mathbf{m}_0$ is initialized to $\mathbf{0} \in \mathbb{R}^{k\times d}$):

$$\mathbf{m}_t = f(\mathbf{f}_t, \mathbf{m}_{t-1}) + \mathbf{e}_t \mathbf{i}_t^\top$$

where $f = \odot$ (element-wise multiplication, in this case $? = d$) or $f = .$ (matrix multiplication, in this case $? = k$).

Finally, output state is obtained from memory by dot product to get the final output $\mathbf{y}_t$:

$$\mathbf{y}_t = \mathbf{m}_t^\top \mathbf{s}_t \in \mathbb{R}^d$$

Forget state, input state, expand state, shrink state are all calculated (or independent of $\mathbf{x}_t$) based on $\mathbf{x}_t$.

For convenience in later discussions, we temporarily refer to this method as MNet (Memory Network). We call this process: **expand, forget, then shrink**.

Example

The above definitions may seem a bit peculiar (but the idea is not much different from regular RNN), in this section, we will point out that the above definition encompasses many widely used sequence modeling methods. We will list the correspondence of each element in the table below:

Method	Shrink State	Forget State	Expand State	Input State	Memory Size	f
Linear Attention	$\mathbf{q}_t \in \mathbb{R}^k$	$\mathbf{I} \in \mathbb{R}^{k\times k}$	$\mathbf{k}_t \in \mathbb{R}^k$	$\mathbf{v}_t \in \mathbb{R}^d$	$k \times d$	Matrix Production
S4	$\mathbf{C} \in \mathbb{R}^k$	$\mathbf{A} \in \mathbb{R}^{k\times k}$	$\mathbf{B} \in \mathbb{R}^k$	$\mathbf{u}_t \in \mathbb{R}^1$	$k \times 1$	Matrix Production
S5	$\mathbf{C} \in \mathbb{R}^k$	$\mathbf{A} \in \mathbb{R}^{k\times k}$	$\mathbf{B} \in \mathbb{R}^k$	$\mathbf{u}_t \in \mathbb{R}^d$	$k \times d$	Matrix Production
TNL	$\mathbf{q}_t \in \mathbb{R}^k$	$\lambda\mathbf{I} \in \mathbb{R}^{k\times k}$	$\mathbf{k}_t \in \mathbb{R}^k$	$\mathbf{v}_t \in \mathbb{R}^d$	$k \times d$	Matrix Production
Mamba	$\mathbf{C}_t \in \mathbb{R}^k$	$\mathbf{A}_t \in \mathbb{R}^{k\times k}$	$\mathbf{B}_t \in \mathbb{R}^k$	$\mathbf{u}_t \in \mathbb{R}^d$	$k \times d$	Element-wise Production
RWKV	$\mathbf{R}_t \in \mathbb{R}^1$	$\exp(-w) \in \mathbb{R}^{1\times 1}$	$\exp(\mathbf{k}_t) \in \mathbb{R}^1$	$\mathbf{v}_t \in \mathbb{R}^1$	1×1	Element-wise Production / Matrix Production
Cosformer	$\mathbf{q}_t \in \mathbb{R}^k$	$\exp(i\theta)\mathbf{I} \in \mathbb{R}^{k\times k}$	$\mathbf{k}_t \in \mathbb{R}^k$	$\mathbf{v}_t \in \mathbb{R}^d$	$k \times d$	Matrix Production
Lrpe	$\mathbf{q}_t \in \mathbb{R}^k$	$\Lambda = \mathrm{diag}\{\exp(i\theta_1), \ldots, \exp(i\theta_k)\} \in \mathbb{R}^{k\times k}$	$\mathbf{k}_t \in \mathbb{R}^k$	$\mathbf{v}_t \in \mathbb{R}^d$	$k \times d$	Matrix Production

Linear Attention [1]

In Linear Attention, we obtain query $\mathbf{q}_t \in \mathbb{R}^k$, key $\mathbf{k}_t \in \mathbb{R}^k$, value $\mathbf{v}_t \in \mathbb{R}^d$ from the input $\mathbf{x}_t \in \mathbb{R}^d$, and recursively calculate as follows:

$$[\mathbf{kv}]_t = [\mathbf{kv}]_{t-1} + \mathbf{k}_t \mathbf{v}_t^\top,$$
$$\mathbf{y}_t = [\mathbf{kv}]_t^\top \mathbf{q}_t.$$

It can be seen that Linear Attention is a special case of MNet.

S4 [2]

In S4, the calculation is as follows:

$$\mathbf{m}_t = \mathbf{A}\mathbf{m}_{t-1} + \mathbf{B}\mathbf{u}_t^\top$$
$$\mathbf{y}_t = \mathbf{m}_t^\top \mathbf{C}.$$

It can be seen that S4 is also a special case of MNet.

Note: The original definition of S4 defined mappings $f_i, i=1,\ldots,d$ of $\mathbb{R}^{n\times 1}\to\mathbb{R}^{n\times 1}$. The mapping f of $\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$ is defined as follows:

$$f(\mathbf{X})_{mi} = f(\mathbf{X}_{:,i})_m.$$

That is, define an S4 for each channel.

S5 [3]

The recurrence equation of S5 is the same as S4, with the only difference being the direct definition of the mapping $\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$:

$$\mathbf{m}_t = \mathbf{A}\mathbf{m}_{t-1} + \mathbf{B}\mathbf{u}_t^\top$$
$$\mathbf{y}_t = \mathbf{m}_t^\top\mathbf{C}.$$

See table for dimensional differences.

TNL [4]

TNL (Transnormer LLM) adds exponential decay to Linear Attention:

$$[\mathbf{kv}]_t = \lambda[\mathbf{kv}]_{t-1} + \mathbf{k}_t\mathbf{v}_t^\top.$$
$$\mathbf{y}_t = [\mathbf{kv}]_t^\top\mathbf{q}_t.$$

Mamba [5]

The recurrence equation of Mamba is also simple:

$$\mathbf{m}_t = \mathbf{A}_t \odot \mathbf{m}_{t-1} + \mathbf{B}_t\mathbf{u}_t^\top$$
$$\mathbf{y}_t = \mathbf{m}_t^\top\mathbf{C_t}.$$

RWKV [6]

If we ignore the denominator of RWKV, the recurrence equation can be simplified to:

$$\mathbf{m}_t = \exp(-w)\mathbf{m}_{t-1} + \exp(\mathbf{k}_t)\mathbf{v}_t^\top.$$
$$\mathbf{y}_t = \mathbf{m}_t^\top\mathbf{r}_t.$$

Note: The original definition of RWKV defined mappings $f_i, i=1,\ldots,d$ of $\mathbb{R}^{n\times 1}\to\mathbb{R}^{n\times 1}$. The mapping f of $\mathbb{R}^{n\times d}\to\mathbb{R}^{n\times d}$ is defined as follows:

$$f(\mathbf{X})_{mi} = f(\mathbf{X}_{:,i})_m.$$

That is, define an RWKV for each channel.

Cosformer [7]

In Cosformer, we obtain query $\mathbf{q}_t\in\mathbb{R}^k$, key $\mathbf{k}_t\in\mathbb{R}^k$, value $\mathbf{v}_t\in\mathbb{R}^d$ from the input $\mathbf{x}_t\in\mathbb{R}^d$, and recursively calculate as follows:

$$
\begin{aligned}
[\mathbf{kv}]_t &= \exp(i\theta)[\mathbf{kv}]_{t-1} + \mathbf{k}_t\mathbf{v}_t^\top. \\
\mathbf{y}_t &= \mathrm{Rel}\{[\mathbf{kv}]_t\}^\top \mathbf{q}_t.
\end{aligned}
$$

Proof:

$$
\begin{aligned}
[\mathbf{kv}]_t &= \sum_{s=1}^{t} \exp(i(t-s)\theta)\mathbf{k}_s\mathbf{v}_s^\top \\
\mathbf{y}_t &= \mathrm{Rel}\{[\mathbf{kv}]_t\}^\top \mathbf{q}_t \\
&= \mathrm{Rel}\left\{\sum_{s=1}^{t} \exp(i(t-s)\theta)\mathbf{v}_s\mathbf{k}_s^\top \mathbf{q}_t\right\} \\
&= \sum_{s=1}^{t} \cos((t-s)\theta)\mathbf{v}_s\mathbf{k}_s^\top \mathbf{q}_t
\end{aligned}
$$

Lrpe [8]

In Lrpe, we obtain query $\mathbf{q}_t \in \mathbb{R}^k$, key $\mathbf{k}_t \in \mathbb{R}^k$, value $\mathbf{v}_t \in \mathbb{R}^d$ from the input $\mathbf{x}_t \in \mathbb{R}^d$, and recursively calculate as follows:

$$
\begin{aligned}
[\mathbf{kv}]_t &= \Lambda[\mathbf{kv}]_{t-1} + \mathbf{k}_t\mathbf{v}_t^\top. \\
\Lambda &= \mathrm{diag}\{\exp(i\theta_1), \ldots, \exp(i\theta_k)\}. \\
\mathbf{y}_t &= \mathrm{Rel}\{[\mathbf{kv}]_t\}^\top \mathbf{q}_t.
\end{aligned}
$$

Explanation:

$$
\begin{aligned}
[\mathbf{kv}]_t &= \sum_{s=1}^{t} \Lambda^{t-s}\mathbf{k}_s\mathbf{v}_s^\top \\
\mathbf{y}_t &= \mathrm{Rel}\{[\mathbf{kv}_t]\}^\top \mathbf{q}_t \\
&= \mathrm{Rel}\left\{\sum_{s=1}^{t} \Lambda^{t-s}\mathbf{v}_s\mathbf{k}_s^\top \mathbf{q}_t\right\} \\
&= \sum_{s=1}^{t} \mathbf{v}_s\mathbf{k}_s^\top \bar{\Lambda}_{t-s}\mathbf{q}_t \\
\bar{\Lambda}_{t-s} &= \mathrm{diag}\{\cos((t-s)\theta_1), \ldots, \cos((t-s)\theta_k)\}
\end{aligned}
$$

Backward

Now that we have defined the Forward form of Mnet, the next step is to define the Backward form. For convenience, we will refer to the case where ($f = \odot$) as Type1, and the case where ($f = .$) as Type2.

Type1:

$$
\begin{aligned}
\mathbf{m}_t &= \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{e}_t\mathbf{i}_t^\top, \\
\mathbf{y}_t &= \mathbf{m}_t^\top \mathbf{s}_t.
\end{aligned}
$$

Type2:

$$
\begin{aligned}
\mathbf{m}_t &= \mathbf{f}_t\mathbf{m}_{t-1} + \mathbf{e}_t\mathbf{i}_t^\top, \\
\mathbf{y}_t &= \mathbf{m}_t^\top \mathbf{s}_t
\end{aligned}
$$

Type1

$$
\begin{aligned}
\mathbf{ds}_t &= \mathbf{m}_t \mathbf{dy_t} \in \mathbb{R}^k, \\
\mathbf{dm}_{t-1} &= \mathbf{f}_t \odot \mathbf{dm}_t + \mathbf{s}_{t-1}\mathbf{dy}_{t-1}^\top \in \mathbb{R}^{k\times d}, \\
\mathbf{df}_t &= \mathbf{dm}_t \odot \mathbf{m}_t \in \mathbb{R}^{k\times d}, \\
\mathbf{de}_t &= \mathbf{dm}_t \mathbf{i}_t \in \mathbb{R}^k, \\
\mathbf{di}_t &= \mathbf{dm}_t^\top \mathbf{e}_t \in \mathbb{R}^d.
\end{aligned}
$$

Type2

$$
\begin{aligned}
\mathbf{ds}_t &= \mathbf{m}_t \mathbf{dy_t} \in \mathbb{R}^k, \\
\mathbf{dm}_{t-1} &= \mathbf{f}_t \mathbf{dm}_t + \mathbf{s}_{t-1}\mathbf{dy}_{t-1}^\top \in \mathbb{R}^{k\times d}, \\
\mathbf{df}_t &= \mathbf{dm}_t \mathbf{m}_t^\top \in \mathbb{R}^{k\times k}, \\
\mathbf{de}_t &= \mathbf{dm}_t \mathbf{i}_t \in \mathbb{R}^k, \\
\mathbf{di}_t &= \mathbf{dm}_t^\top \mathbf{e}_t \in \mathbb{R}^d.
\end{aligned}
$$

(need check)

Fast computation

Todo

Citations

1. Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 5156–5165. PMLR, 2020.
2. Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.
3. Jimmy T.H. Smith, Andrew Warrington, & Scott Linderman (2023). Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations* .
4. Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, & Yiran Zhong. (2023). TransNormerLLM: A Faster and Better Large Language Model with Improved TransNormer.
5. Albert Gu, & Tri Dao. (2023). Mamba: Linear-Time Sequence Modeling with Selective State Spaces.
6. Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, & Rui-Jie Zhu. (2023). RWKV: Reinventing RNNs for the Transformer Era.

7. Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, & Yiran Zhong. (2022). cosFormer: Rethinking Softmax in Attention.
8. Zhen Qin, Weixuan Sun, Kaiyue Lu, Hui Deng, Dongxu Li, Xiaodong Han, Yuchao Dai, Lingpeng Kong, & Yiran Zhong. (2023). Linearized Relative Positional Encoding.

Log

1. 20240217: Initial version.
2. 20240226: Fixed typos.
3. 20240305: Modified names, added more examples, and backward computations.
4. 20240306: Fixed typos.