Lecture 17

- State space models continued

→ How to support long-range dependancies?

$$\vec{h}_{t+1} = A\vec{h}_t + B\vec{x}_t$$
$$\vec{y}_t = C \cdot \vec{x}_t$$

All information about the past can only be captured in $\vec{h}_t$. It has to fit.

Efficiency versus Efficacy tradeoff.

Larger $\vec{h}_t$ ⇒ more memory/history tracking
but also, more compute.

So want to make $\vec{h}_t$ as large as possible, while still being able to run

---

S4

Efficiency via FFT – use of LTI systems.

What about long range dependancy?

Let $A = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_N \end{bmatrix}$ $\vec{h}_t$ is N dimensional.

How is λ related to system stability?

$|\lambda| > 1 \longrightarrow$ System explodes $\rightarrow$ bad.

$|\lambda| < 1 \longrightarrow$ history dies out fast.

$|\lambda| = 1 \longrightarrow$ critical damping.

For $\lambda \in \mathbb{R}$, $|\lambda| = 1 \Rightarrow \lambda = \pm 1$.

→ Only two choices!

→ Far apart from each other (So not GD friendly)

<u>Solution:</u> Complex eigenvalues!

$$\lambda = \exp\left(-\mathrm{ReLU}(\lambda_R) + j\lambda_I\right) \qquad j = \sqrt{-1}$$

- Exact initalization matters, have to be mostly on or very close to unit circle with some inside unit circle
- Hippo-based initialization - like analytic pre-training.
- Freezing A with good initialization and only training the rest also seemed to work reasonably!

Table 4: (**Long Range Arena**) (*Top*) Original Transformer variants in LRA. Full results in Appendix D.2. (*Bottom*) Other models reported in the literature. *Please read Appendix D.5 before citing this table.*

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
BigBird	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans.	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nyströmformer	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256	37.25	64.57	79.29	47.38	77.72	✗	59.37
S4	**59.60**	**86.82**	**90.90**	**88.65**	**94.20**	**96.35**	**86.09**

Outperformed Transformer models on long-range benchmarks!

But didn't quite match / beat transformers on other tasks.

Mamba / S6.

State-space models

- Recurrent view (state-based)

"inference time"

- Input/Output view (Convolution based)

"training time"

- Continuous-time view

"selectivity/gating"

Discrete-time system: $\vec{h}_{t+1} = A\vec{h}_t + B\vec{u}_t$

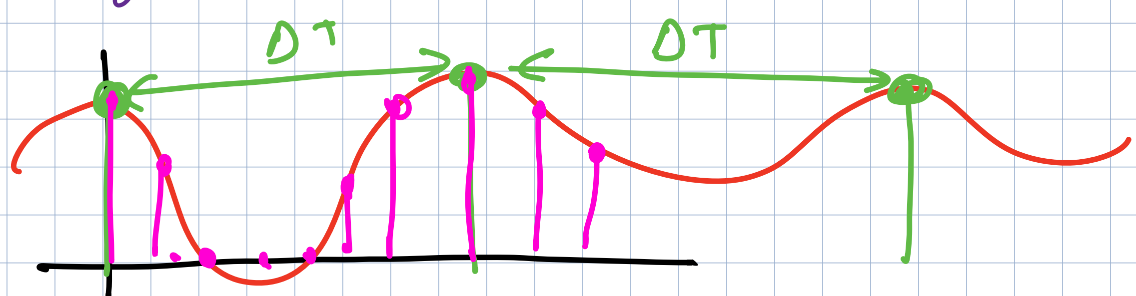
$\vec{y}_{t+1} = C\cdot\vec{h}_t$

Think of as a discretization of a continuous time system.
(sampling)

$\frac{d}{dt}\cdot\vec{h}(t) = A\vec{h}(t) + B\vec{u}(t)$ ← (Diffrent A, B)

$\vec{y}(t) = C\cdot\vec{h}(t)$

To convert from CT to DT, we need a sampling interval ΔT.



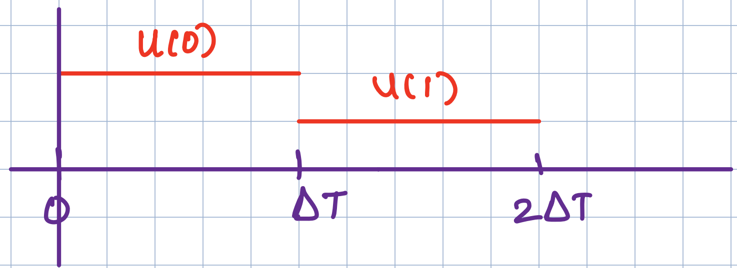
Scalar case:

$$\dot{x}(t) = a\,x(t) + b \cdot u(t).$$

Sampling time ΔT.

Assume $u(t)$ is piecewise constant for time ΔT.

For small ΔT this is reasonable.



Aside:

$a > 0 \Rightarrow$ explode

$a < 0 \Rightarrow$ decay to 0.

$$x(\Delta T) = \underbrace{e^{a\Delta T} \cdot x(0)}_{\text{exponential growth.}} + \underbrace{\frac{b}{a}\left(e^{a\Delta T} - 1\right) u(0)}_{\text{impact of input}}.$$

Assume $a < 0$

Then ΔT close to $0 \Rightarrow e^{a\Delta T} \approx 1 \Rightarrow$ Remember history.

Initial state?

ΔT large $\Rightarrow e^{a\Delta T}$ decaying exponential $\Rightarrow$ forget past

ΔT smallish $\Rightarrow$ $e^{a\Delta T} - 1 \approx \underbrace{1 + a\Delta T}_{\text{Taylor expansion.}} - 1 \approx a\Delta T.$

$$\Rightarrow \quad \frac{b}{a}(e^{a\Delta T} - 1) = b\Delta T.$$

Effect of input?

ΔT close to 0 $\Rightarrow$ $b\Delta T \approx 0.$ $\Rightarrow$ input doesn't matter

ΔT large $\Rightarrow$ $b\Delta T$ large $\Rightarrow$ input matters a lot.

So by changing ΔT, can change the relative importance of the past.

Idea: Choose ΔT based on input (i.e. learn it).

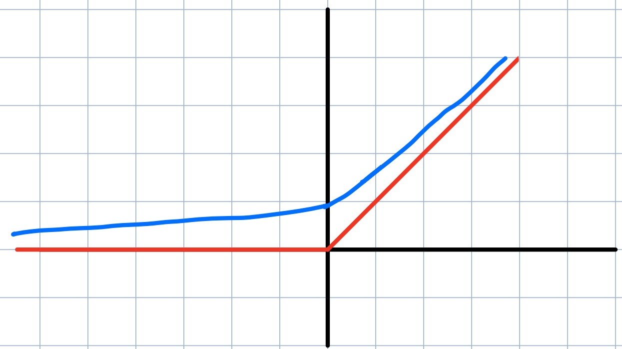
Try: Linear function of $\vec{u}$?

$$\langle \vec{w}, \vec{u} \rangle$$

Try: $\mathrm{ReLU}(\langle \vec{w}, \vec{u} \rangle + b)$

Softplus:

$$\text{Softplus}(x) = \ln(1+e^{x})$$



if $a=b=N=1$

$$g_t = \sigma(\text{Linear}(\vec{x}_t))$$

$$\vec{h}_t = g_t \vec{h}_{t-1} + (1-g_t)\vec{x}_t$$

So this gives us for each channel of input:

$$\vec{h}_{k+1} = A_k \vec{h}_k + B_k x_k$$

$$\vec{y}_{k+1} = C \vec{h}_k$$

$$A_k = \begin{bmatrix} e^{\lambda_1 \Delta T} & & \\ & e^{\lambda_2 \Delta T} & \\ & & \ddots \end{bmatrix}$$

$$B_k = \vec{b}_k = \vec{b}\,\Delta T.$$

But... have we lost anything?

$$A^{k} \rightarrow A_k A_{k-1} \cdots A_1 A_0$$

Initialization:

λI blocks for A.

$$\lambda_n = -\frac{1}{2} + ni$$

$$\lambda_n = \frac{-1}{2}$$

$$\lambda_n = -(n+1)$$

State-space models

- Recurrent view (state-based)

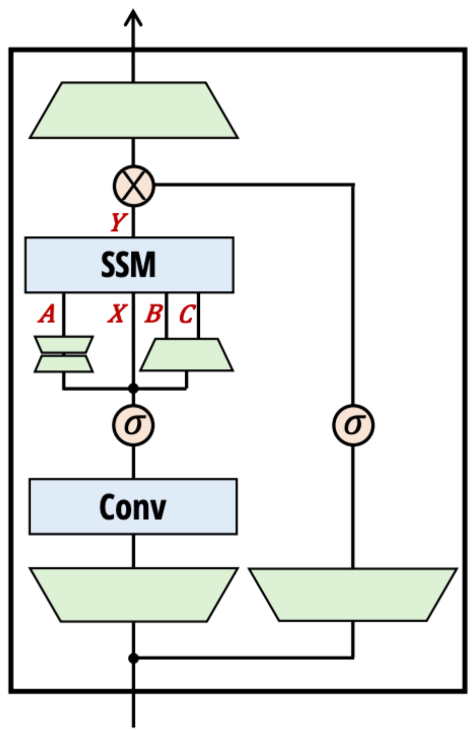
"inference time"

- Input / Output view (Convolution based

" training time"

- Continuous-time view

"selectivity) gating"



Sequential Mamba Block