State-Space Models

Generative AI and Foundation Models

Spring 2024

Department of Mathematical Sciences

Ernest K. Ryu

Seoul National University

Background: Matrix exponential

For $A \in \mathbb{C}^{n\times n}$, define

$$e^A = \exp(A) := \sum_{j=0}^{\infty} \frac{1}{j!} A^j = I + A + \frac{1}{2}A^2 + \frac{1}{6}A^3 + \cdots$$

(It can be shown that this power series is globally convergent.)

If $A \in \mathbb{C}^{n\times n}$ is diagonalizable, i.e., there is an invertible $V \in \mathbb{C}^{n\times n}$ such that

$$A = V \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} V^{-1} \quad \text{then,} \quad e^A = V \begin{bmatrix} e^{\lambda_1} & & & \\ & e^{\lambda_2} & & \\ & & \ddots & \\ & & & e^{\lambda_n} \end{bmatrix} V^{-1}.$$

Background: Matrix exponential

The matrix exponential arises as the solution to the ODE:

$$\dot{x} = Ax(t), \qquad x(0) = x_0 \in \mathbb{C}^n$$

Then, the solution is

$$x(t) = e^{tA}x_0.$$

More generally, let $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times k}$. Let $u(t) \in \mathbb{C}^k$ for $t \geq 0$ be given. Then, the ODE

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(0) = x_0 \in \mathbb{C}^n$$

is solved by the *variation of parameters formula*

$$x(t) = e^{tA}x_0 + \int_0^t e^{(t-s)A}Bu(s)\,ds$$

Background: ODE discretization

Let $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times k}$, and consider the linear time-invariant ODE

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(0) = x_0 \in \mathbb{C}^n$$

For any $t \geq 0$ and $\Delta t \geq 0$, the solution satisfies

$$x(t+\Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\, ds$$

An ODE discretization (integrator) is a scheme for approximating this formula.

Background: ODE discretization

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\ ds$$

Forward Euler discretization:

$$x(t + \Delta t) \approx (I + \Delta t A)x(t) + \Delta t B u(t)$$

derived from

$$x(t + \Delta t) \approx x(t) + \Delta t(Ax(t) + Bu(t))$$

Background: ODE discretization

$$x(t+\Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\, ds$$

Backward Euler discretization:

$$x(t+\Delta t) \approx (I - \Delta t A)^{-1} x(t) + \Delta t (I - \Delta t A)^{-1} B u(t+\Delta t)$$

derived from

$$x(t+\Delta t) \approx x(t) + \Delta t \big(A x(t+\Delta t) + B u(t+\Delta t) \big)$$

However, the HiPPO paper uses

$$x(t+\Delta t) \approx (I - \Delta t A)^{-1} x(t) + \Delta t (I - \Delta t A)^{-1} B u(t)$$

A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, HiPPO: Recurrent memory with optimal polynomial projections, *NeurIPS*, 2020.

Background: ODE discretization

$$x(t+\Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\, ds$$

Bilinear discretization:

$$x(t+\Delta t) \approx \left(I - \frac{\Delta t}{2}A\right)^{-1}\left(I + \frac{\Delta t}{2}A\right)x(t) + \Delta t\left(I - \frac{\Delta t}{2}A\right)^{-1}B\frac{1}{2}\left(u(t)+u(t+\Delta t)\right)$$

derived from

$$x(t+\Delta t) \approx x(t) + \Delta t\left(A\frac{1}{2}\left(x(t)+x(t+\Delta t)\right) + B\frac{1}{2}\left(u(t)+u(t+\Delta t)\right)\right)$$

However, the HiPPO paper uses

$$x(t+\Delta t) \approx \left(I - \frac{\Delta t}{2}A\right)^{-1}\left(I + \frac{\Delta t}{2}A\right)x(t) + \Delta t\left(I - \frac{\Delta t}{2}A\right)^{-1}Bu(t)$$

A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, HiPPO: Recurrent memory with optimal polynomial projections, *NeurIPS*, 2020.

Background: ODE discretization

$$x(t+\Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\, ds$$

Generalized Bilinear Transformation (GBT) with $\alpha \in (0,1)$

$$x(t+\Delta t) \approx (I - \Delta t \alpha A)^{-1}(I + \Delta t(1-\alpha)A)x(t) + \Delta t(I - \Delta t \alpha A)^{-1} Bu(t)$$

derived from

$$x(t+\Delta t) \approx x(t) + \Delta t\Big(A\big((1-\alpha)x(t) + \alpha x(t+\Delta t)\big) + Bu(t)\Big)$$

Background: ODE discretization

$$x(t+\Delta t) = x(t) + \int_t^{t+\Delta t} Ax(s) + Bu(s)\, ds$$

Zero-order hold (ZOH)

$$x(t+\Delta t) \approx e^{\Delta t A} x(t) + \Big(\int_0^{\Delta t} e^{sA}\, ds \Big) Bu(t)$$

derived from

$$x(t+\Delta t) = e^{\Delta t A} x(t) + \int_0^{\Delta t} e^{(\Delta t - s)A} Bu(t+s)\, ds$$

and the approximation that $u(s) = u(t)$ for $s \in [t, t+\Delta t]$.

If A is invertible,

$$x(t+\Delta t) \approx e^{\Delta t A} x(t) + \left(e^{\Delta t A} - I\right) A^{-1} Bu(t)$$

Background: Inner product spaces

Inner product of vectors $x, y \in \mathbb{R}^d$

$$\langle x, y \rangle := \sum_{i=1}^{d} x_i y_i$$

For $f : \mathbb{R} \to \mathbb{R}$ and $g: \mathbb{R} \to \mathbb{R}$, define the inner product with respect to a measure μ as

$$\langle f, g \rangle_\mu := \int_{\mathbb{R}} f(x)g(x)\, d\mu(x) = \int_{-\infty}^{\infty} f(x)g(x)\, d\mu(x)$$

In this class, the measure μ will always have density ω.
In other words, $d\mu(x) = \omega(x)dx$ (with $\omega(x) \geq 0$ everywhere) and

$$\langle f, g \rangle_\mu := \int_{\mathbb{R}} f(x)g(x)\omega(x)\, dx = \int_{-\infty}^{\infty} f(x)g(x)\omega(x)\, dx$$

Background: Inner product spaces

We say f, g are *orthogonal* (with respect to μ) if

$$\langle f, g \rangle_\mu = 0$$

We define a function norm $\|\cdot\|_\mu$ as

$$\|f\|_\mu := \sqrt{\langle f, f \rangle_\mu}$$

We say f is *normalized* (with respect to μ) if

$$\|f\|_\mu = 1$$

Background: Orthogonal polynomials

Orthogonal polynomials with respect to μ are a sequence of polynomials $\{P_n\}_{n=0}^{\infty}$ such that

$$\deg P_n = n, \qquad \langle P_i, P_j \rangle_\mu = 0 \text{ for } i \neq j.$$

We say $\{P_n\}_{n=0}^{\infty}$ are orthonormal polynomials if every P_n is normalized, i.e., if $\|P_n\|_\mu = 1$ for $n = 0,1,2, \ldots$.

The measures of primary interest are

$$w(x) = \mathbf{1}_{[-1,1]}(x) \qquad \text{(Legendre)}$$

$$w(x) = \mathbf{1}_{[0,\infty)}(x)e^{-x} \qquad \text{(Laguerre)}$$

$$w(x) = \mathbf{1}_{(-1,1)}(x)\frac{1}{\sqrt{1-x^2}} \qquad \text{(Chebyshev)}$$

Background: Orthogonal polynomials

In fact, every measure induces a unique sequence of orthonormal polynomials. This can be found by orthogonalizing the monomial basis with Gram–Schmidt with respect to $\langle \cdot, \cdot \rangle_\mu$.

Example) Laguerre orthonormal polynomials:

$$
\begin{aligned}
L_0(x) &= 1 \\
L_1(x) &= 1 - x \\
L_2(x) &= \frac{x^2}{2} - 2x + 1 \\
L_3(x) &= -\frac{x^3}{6} + \frac{3x^2}{2} - 3x + 1 \\
L_4(x) &= \frac{x^4}{24} - \frac{2x^3}{3} + 3x^2 - 4x + 1 \\
L_5(x) &= -\frac{x^5}{120} + \frac{5x^4}{24} - \frac{5x^3}{3} + 5x^2 - 5x + 1
\end{aligned}
$$

Background: Orthogonal polynomials

Let $\{P_n\}_{n=0}^{\infty}$, orthogonal polynomials with respect to μ, and f be given.

The best degree-$(N-1)$ polynomial approximation of f, precisely defined as the solution to

$$\underset{\deg(g)\leq N-1}{\text{minimize}} \quad \|f-g\|_\mu$$

can be obtained by

$$g(x)=\sum_{i=0}^{N-1} c_i P_i(x), \qquad c_i=\frac{\langle f,P_i\rangle_\mu}{\|P_i\|_\mu^2}$$

Background: Orthogonal polynomials

Pseudo-proof) Assume $$f(x) = \sum_{i=0}^{\infty} c_i P_i(x)$$

Then,

$$\langle f, P_j \rangle = \sum_{i=0}^{\infty} c_i \langle P_i, P_j \rangle = c_j \|P_j\|_\mu^2$$

gives us the formula to compute $c_0, c_1, \ldots$. If $g(x) = \sum_{i=0}^{N-1} c_i' P_i(x)$, then

$$\|f - g\|_\mu^2 = \left\langle \sum_{i=N}^{\infty} c_i P_i - \sum_{i=0}^{N-1} (c_i - c_i') P_i, \sum_{i=N}^{\infty} c_i P_i - \sum_{i=0}^{N-1} (c_i - c_i') P_i \right\rangle_\mu$$

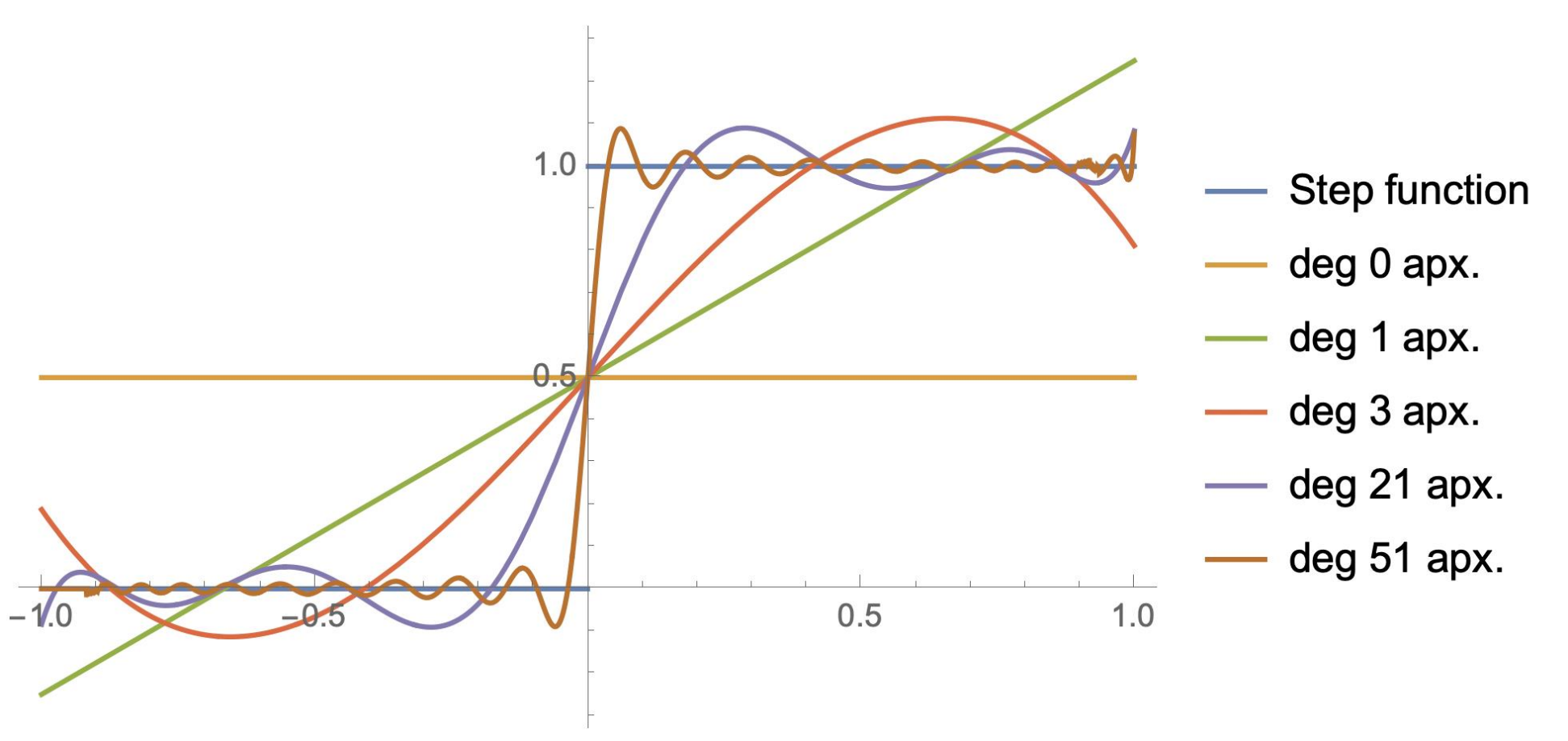
$$= \sum_{i=N}^{\infty} c_i^2 \|P_i\|_\mu^2 + \sum_{i=0}^{N-1} (c_i - c_i')^2 \|P_i\|_\mu^2$$

is minimized with $c_i' = c_i$ for $i = 0, \ldots, N-1$.

■

Example) Polynomial approximation

The following is an example of approximating a step function with Legendre polynomials.



RNNs vs transformers language models

RNNs have an advantage over transformers in one regard.

- More efficient inference (sequence generation).
    - Generation cost depends linearly on sequence length.

Transformers outperform classical (LSTM- or GRU-based) RNNs for two main reasons.

- TF allows efficient parallel computation during training.
    - Classical (non-linear) RNNs are inherently sequential. We return to this point later.
- RNN hidden state is unable to faithfully retain sequence information.
    - Long shot-term memory is not long enough.

Memory units

What if we add memory units to RNNs so that they can maintain longer memory of the input sequence, longer than the hidden state of LSTMs?

Assume $\{f_\ell\}_{\ell=1}^L$ or $\{f(t)\}_{t\in[0,T]}$ is an input sequence. A transformer architecture has no compression or memory mechanism; the $\{q_\ell\}_{\ell=1}^L, \{k_\ell\}_{\ell=1}^L, \{v_\ell\}_{\ell=1}^L$ vectors derived from $\{f_\ell\}_{\ell=1}^L$ are all accessible from all time-steps.

Use the notation $f_{\leq \ell} = \{f_1, \dots, f_\ell\}$ and $f_{\leq t} = \{f(\tau)\}_{\tau\in[0,t]}$ to denote the inial part of the signal.

Assume $c_\ell \in \mathbb{C}^N$ or $c(t) \in \mathbb{C}^N$ is a "summary" or "memory" of $f_{\leq \ell}$ or $f_{\leq t}$.
How do we know the summary is good?

Memory units with online function approximation

Conceptually, we say $c(t) \in \mathbb{C}^N$ is a good summary of $f_{\leq t}$ if there is a reconstruction mechanism

$$c(t) \mapsto \hat{f}_{\leq t}$$

such that $\hat{f}_{\leq t} \approx f_{\leq t}$.

Online function approximation has two goals:

1. Find $c(t) \in \mathbb{R}^N$ that is a good summary of $f_{\leq t}$.
2. Update $c(t)$ online.

A. Voelker, I. Kajić, and C. Eliasmith, Legendre memory units: Continuous-time representation in recurrent neural networks, *NeurIPS*, 2019.

High-order polynomial projection operators (HiPPO) framework

For every $t \geq 0$, let $\mu^{(t)}$ be a time-varying measure supported on $[0, t]$. (So $\omega(t, x) > 0$ only on $x \in [0, t]$ and $\omega(t, x) = 0$ for $x \notin [0, t]$.)

We seek to find some polynomial $g^{(t)}$ of degree at most $N - 1$ that minimizes the approximation error : $\| f_{\leq t} - g^{(t)} \|_{\mu^{(t)}}$

We let $c(t) \in \mathbb{R}^N$ be the coefficients of orthonormal polynomials $\{P_i^{(t)}\}$ defined with respect to the measure $\mu^{(t)}$:

$$g^{(t)}(x) = \sum_{i=0}^{N-1} c_i(t) P_i^{(t)}(x), \qquad c_i(t) = \langle f, P_i^{(t)} \rangle_{\mu^{(t)}} = \langle f_{\leq t}, P_i^{(t)} \rangle_{\mu^{(t)}} = \int_0^t f(x) P_i^{(t)}(x) \omega(t, x)\, dx$$

A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, HiPPO: Recurrent memory with optimal polynomial projections, *NeurIPS*, 2020.

On-line calculation of $c(t)$

Re-computing $c(t)$ at every time t would be infeasible.

For the measures $\mu^{(t)}$ of interest, incredibly, the dynamics of $c(t)$ can be described by the ODE

$$\frac{d}{dt}c(t) = A(t)c(t) + B(t)f(t), \qquad A(t) \in \mathbb{R}^{N\times N}, B(t) \in \mathbb{R}^{N\times 1}$$

In the following, we derive the above ODE with uniform measure (HiPPO-LegS):

$$\omega^{(t)}(x) = \omega(t,x) = \tfrac{1}{t}\mathbf{1}_{[0,t]}(x)$$

Legendre Polynomials

The Legnedre Polynomials $(P_n)_{n=0,1,\dots}$ are defined by the following conditions:

$$P_0(x) = 1$$

$$P_n(x) \text{ is a polynomial of degree } n \text{ such that } \begin{cases} P_n(1) = 1 \\ \int_{-1}^{1} P_n(x)P_m(x)dx = 0 \quad (m < n) \end{cases}$$

These conditions give n+1 equations, determining a unique polynomial of deg n.

There are other ways to define Legendre Polynomials, e.g. solutions to Legendre's differential equation: $(1-x^2)P_n''(x) - 2xP_n'(x) + n(n+1)P_n(x) = 0.$

Examples)

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{1}{2}(3x^2 - 1)$$
$$P_3(x) = \frac{1}{2}(5x^3 - 3x)$$
$$P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$$
$$\vdots$$

$$\int_{-1}^{1} P_2(x)P_3(x)dx = \int_{-1}^{1} \frac{1}{4}(15x^5 - 14x^3 + 3x)dx = 0$$

Properties of Legendre Polynomials

The Legnedre polynomial P_n satisfies:

$$\frac{2n+1}{2}\int_{-1}^{1} P_n(x)P_m(x)dx = \delta_{nm}$$

$$P_n(1) = 1, \qquad P_n(-1) = (-1)^n$$

$$(2n+1)P_n(x) = P'_{n+1}(x) - P'_{n-1}(x) \qquad \text{(Leg1)}$$

for $n \geq 0$ (with $P'_{-1} = 0$), which implies

$$P'_{n+1}(x) = (2n+1)P_n(x) + (2n-3)P_{n-2}(x) + \cdots \qquad \text{(Leg2)}$$

where the sum stops at P_0 or P_1.

Properties of Legendre Polynomials

$$P'_{n+1}(x) = (2n+1)P_n(x) + (2n-3)P_{n-2}(x) + \cdots \qquad \text{(Leg2)}$$

Also,

$$P'_{n+1}(x) = (n+1)P_n(x) + xP'_n(x) \qquad \text{(Leg3)}$$

for $n \geq 0$, which (by adding $P'_n(x)$ to both sides) implies

$$(x+1)P'_n(x) = P'_{n+1}(x) + P'_n(x) - (n+1)P_n(x)$$

Now we apply (Leg2) twice on RHS to get:

$$(x+1)P'_n(x) = nP_n + (2n-1)P_{n-1} + (2n-3)P_{n-2} + \cdots \qquad \text{(Leg4)}$$

where the sum stops at P_0.

Derivation for HiPPO-LegS

Recall, $d\mu^t(x) = \omega(t,x)dx$ by

$$\omega(t,x) = \frac{1}{t}\mathbf{1}_{[0,t]}(x)$$

and let

$$g_n(t,x) = \sqrt{2n+1}P_n\left(\frac{2x}{t} - 1\right)$$

where P_n are basic Legendre polynomials. Then $\{g_n(t,\cdot)\}_{n\in\mathbb{N}}$ form an orthonormal basis with respect to the measure μ^t.

Derivation for HiPPO-LegS

First, $\frac{\partial}{\partial t}\omega(t,\cdot) = -t^{-2}\mathbb{1}_{[0,t]} + t^{-1}\delta_t = t^{-1}(-\omega(t,\cdot) + \delta_t)$

results in $\int_0^\infty f(x)g_n(t,x)\frac{\partial}{\partial t}\omega(t,x)dx = -t^{-1}\int_0^\infty f(x)g_n(t,x)\omega(t,x)dx + t^{-1}\int_0^\infty f(x)g_n(t,x)\delta_t(x)dx$

$$= -t^{-1}c_n(t) + t^{-1}f(t)g_n(t,t)$$

Moreover,

$$\begin{aligned}
\frac{\partial}{\partial t}g_n(t,x) &= -(2n+1)^{\frac{1}{2}}2xt^{-2}P_n'\left(\frac{2x}{t}-1\right) \\
&= -(2n+1)^{\frac{1}{2}}t^{-1}\left((\frac{2x}{t}-1)+1\right)P_n'\left(\frac{2x}{t}-1\right) \\
&= -(2n+1)^{\frac{1}{2}}t^{-1}(z+1)P_n'(z) \qquad \left(z := \frac{2x}{t}-1\right) \\
&= -(2n+1)^{\frac{1}{2}}t^{-1}\left[nP_n(z) + (2n-1)P_{n-1}(z) + (2n-3)P_{n-2}(z) + \cdots\right] \qquad \text{(by Leg4)} \\
&= -t^{-1}(2n+1)^{\frac{1}{2}}\left[n(2n+1)^{-\frac{1}{2}}g_n(t,x) + (2n-1)^{\frac{1}{2}}g_{n-1}(t,x) + (2n-3)^{\frac{1}{2}}g_{n-2}(t,x) + \cdots\right]
\end{aligned}$$

Derivation for HiPPO-LegS

Then for fixed n, we have

$$\begin{aligned}
\frac{d}{dt}c_n(t) &= \frac{d}{dt}\int_0^\infty f(x)g_n(t,x)d\mu^{(t)}(x) = \int_0^\infty f(x)\left(\partial_t g_n(t,x)\right)\omega(t,x)dx + \int_0^\infty f(x)g_n(t,x)\left(\partial_t \omega(t,x)\right)dx \\
&= \langle f, \partial_t g_n(t,\cdot)\rangle_{\mu^{(t)}} - t^{-1}c_n(t) + t^{-1}f(t)g_n(t,t) \\
&= -t^{-1}(2n+1)^{\frac{1}{2}}\left[n(2n+1)^{-\frac{1}{2}}\langle f, g_n(t,\cdot)\rangle + (2n-1)^{\frac{1}{2}}\langle f, g_{n-1}(t,\cdot)\rangle(2n-3)^{\frac{1}{2}}\langle f, g_{n-2}(t,\cdot)\rangle + \cdots\right] \\
&\qquad - t^{-1}c_n(t) + t^{-1}f(t)g_n(t,t) \\
&= -t^{-1}(2n+1)^{\frac{1}{2}}\left[n(2n+1)^{-\frac{1}{2}}c_n(t) + (2n-1)^{\frac{1}{2}}c_{n-1}(t) + (2n-3)^{\frac{1}{2}}c_{n-2}(t) + \cdots\right] \\
&\qquad - t^{-1}c_n(t) + t^{-1}f(t)g_n(t,t) \\
&= -t^{-1}(2n+1)^{\frac{1}{2}}\left[(n+1)(2n+1)^{-\frac{1}{2}}c_n(t) + (2n-1)^{\frac{1}{2}}c_{n-1}(t) + (2n-3)^{\frac{1}{2}}c_{n-2}(t) + \cdots\right] \\
&\qquad + t^{-1}(2n+1)^{\frac{1}{2}}f(t),
\end{aligned}$$

as $g_n(t,t) = (2n+1)^{\frac{1}{2}}P_n(1) = (2n+1)^{\frac{1}{2}}$.

Derivation for HiPPO-LegS

Gathering the results for all n, we have

$$\frac{d}{dt}c(t) = -\frac{1}{t}Ac(t) + \frac{1}{t}Bf(t), \qquad c(0) = 0 \in \mathbb{R}^N$$

where

$$A_{nk} = \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases} \qquad B_n = (2n+1)^{\frac{1}{2}}$$

Summary: LegS

Scaled Legendre (LegS) measure assigns uniform weight to entire history:

$$\omega(t,x) = \tfrac{1}{t}\mathbf{1}_{[0,t]}(x) \qquad \langle f,g\rangle_{\mu^{(t)}} = \frac{1}{t}\int_0^t f(x)g(x)\,dx$$

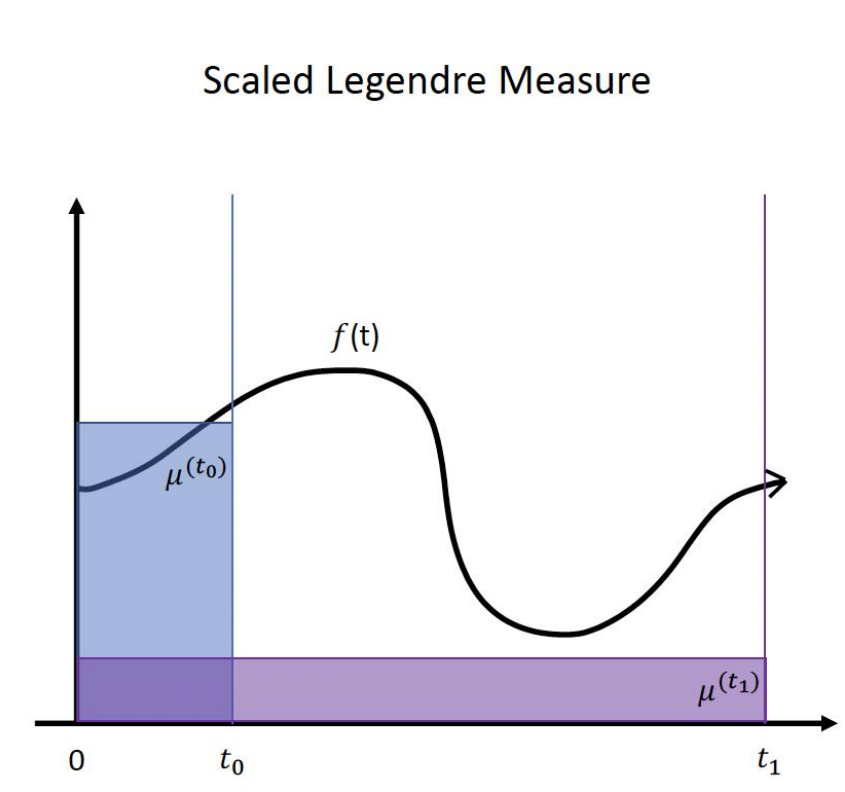
Summary of LegS HiPPO update:

$$\frac{d}{dt}c(t) = -\frac{1}{t}Ac(t) + \frac{1}{t}Bf(t), \qquad c(0) = 0 \in \mathbb{R}^N$$

$$A_{nk} = \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases} \qquad B_n = (2n+1)^{\frac{1}{2}}$$

$$f(x) \approx \sum_{n=0}^{N-1} c_n(t)p_n(t,x), \qquad x \in [0,t] \qquad p_n(t,x) = (2n+1)^{\frac{1}{2}}P_n\left(\frac{2x}{t}-1\right)$$

where $\{P_n\}_{n\in\mathbb{N}}$ is the Legendre polynomials



Summary: LegT

Translated Legendre (LegT) measures assigns uniform weight to $[t-\tau, t]$, most recent history:

$$\omega(t,x) = \frac{1}{\tau}\mathbf{1}_{[t-\tau,t]}(x)$$

$$\langle f,g\rangle_{\mu^{(t)}} = \frac{1}{\tau}\int_{t-\tau}^{t} f(x)g(x)\,dx = \frac{1}{\tau}\int_{\max\{t-\tau,0\}}^{t} f(x)g(x)\,dx$$

(The second equality holds if we assume that $f(x) = 0$ for $x < 0$.)

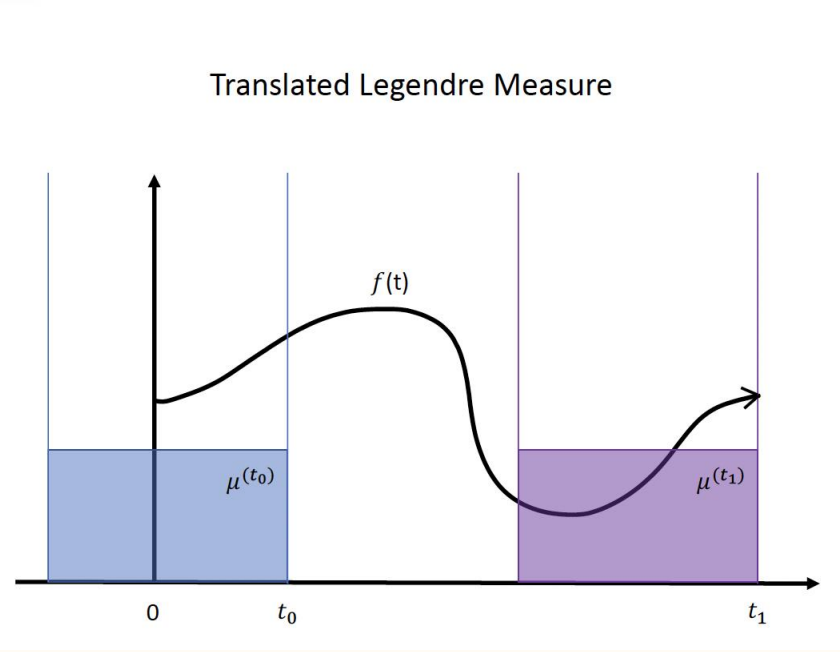
Summary of LegT HiPPO update:

$$\frac{d}{dt}c(t) = -Ac(t) + Bf(t), \qquad c(0) = 0 \in \mathbb{R}^n$$

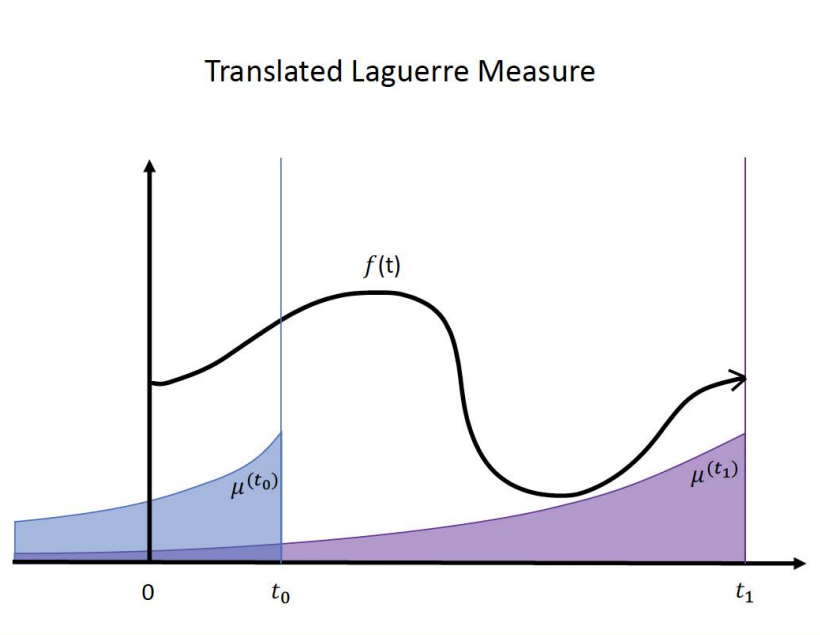
$$A_{nk} = \frac{1}{\tau}\begin{cases}(-1)^{n-k}(2n+1) & \text{if } n \geq k\\ 2n+1 & \text{if } n \leq k\end{cases} \qquad B_n = \frac{1}{\tau}(2n+1)(-1)^n$$

$$f(x) \approx \sum_{n=0}^{N-1} c_n(t)g_n(t,x), \qquad x \in [t-\tau, t] \qquad g_n(t,x) = (2n+1)^{1/2}P_n\left(\frac{2(x-t)}{\tau}+1\right)$$

where $\{P_n\}_{n\in\mathbb{N}}$ is the Legendre polynomials



Summary: LagT



Translated Laguerre (LagT) measures uses the exponentially decaying measure, assigning more importance to recent history.

$$\omega(t,x) = e^{-(t-x)}\mathbf{1}_{(-\infty,t]} \qquad \langle f,g\rangle_{\mu^{(t)}} = \int_{-\infty}^{t} e^{-(t-x)} f(x)g(x)\,dx = \int_{0}^{t} e^{-(t-x)} f(x)g(x)\,dx$$

The second equality holds if we assume that $f(x) = 0$ for $x < 0$.

Summary of LagT HiPPO update:

$$\frac{d}{dt}c(t) = -Ac(t) + Bf(t), \qquad c(0) = 0 \in \mathbb{R}^N$$

$$A_{nk} = \begin{cases} 1 & \text{if } n \geq k \\ 0 & \text{if } n < k \end{cases} \qquad B_n = 1$$

$$f(x) \approx \sum_{n=0}^{N-1} c_n(t)p_n(t,x), \qquad x \in [0,t] \qquad\qquad p_n(t,x) = L_n(t-x)$$

where $\{L_n\}_{n\in\mathbb{N}}$ are the Laguerre polynomials

Summary: Translated Fourier

The Fourier basis $e^{2\pi inx}$ (for $n = 0, \dots, N-1$) can be seen as an orthogonal polynomials basis z^n with respect to the uniform measure on the unit circle $\{z \in \mathbb{C} | |z| = 1\}$. By a change of variable $z \mapsto e^{2\pi ix}$ (and thus changing the domain from the unit circle to $[0,1]$), we obtain the usual Fourier basis $e^{2\pi inx}$. The complex inner product is defined as

$$\omega(t,x) = \frac{1}{\tau}\mathbf{1}_{[t-\tau,t]} \qquad \langle f, g \rangle_{\mu^{(t)}} = \int_{t-\tau}^{t} f(x)\overline{g}(x)\, dx$$

Summary of translated Fourier HiPPO update:

$$\frac{d}{dt}c(t) = -Ac(t) + Bf(t), \qquad c(0) = 0 \in \mathbb{C}^N$$

$$A_{nk} = \begin{cases} -1/\tau & \text{if } k \neq n \\ (2\pi in - 1)/\tau & \text{if } k = n \end{cases} \qquad B_n = \frac{1}{\tau}$$

$$f(x) \approx \sum_{n=0}^{N-1} c_n(t) e^{2\pi i \frac{t-x}{\tau}}, \qquad x \in [t-\tau, t]$$

Discretization of HiPPO-LegS

$$\frac{d}{dt}c(t) = -\frac{1}{t}Ac(t) + \frac{1}{t}Bf(t)$$

The simplest forward Euler discretization with $t = \Delta t \cdot k$, $c_k = c(\Delta t \cdot k)$, and $f_k = f(\Delta t \cdot k)$ gives us

$$c_{k+1} - c_k = -\frac{\Delta t}{\Delta t k}Ac_k + \frac{\Delta t}{\Delta t k}Bf_k$$

$$c_{k+1} = \underbrace{(I - \frac{1}{k}A)}_{=\bar{A}_k}c_k + \underbrace{\frac{1}{k}B}_{=\bar{B}_k}f_k$$

So,

$$c_{k+1} = \bar{A}_k c_k + \bar{B}_k f_k$$

(Other discretization schemes lead to slightly different $\bar{A}_k$ and $\bar{B}_k$.)

Discretization of HiPPO-LegS

$$\frac{d}{dt}c(t) = -\frac{1}{t}Ac(t) + \frac{1}{t}Bf(t)$$

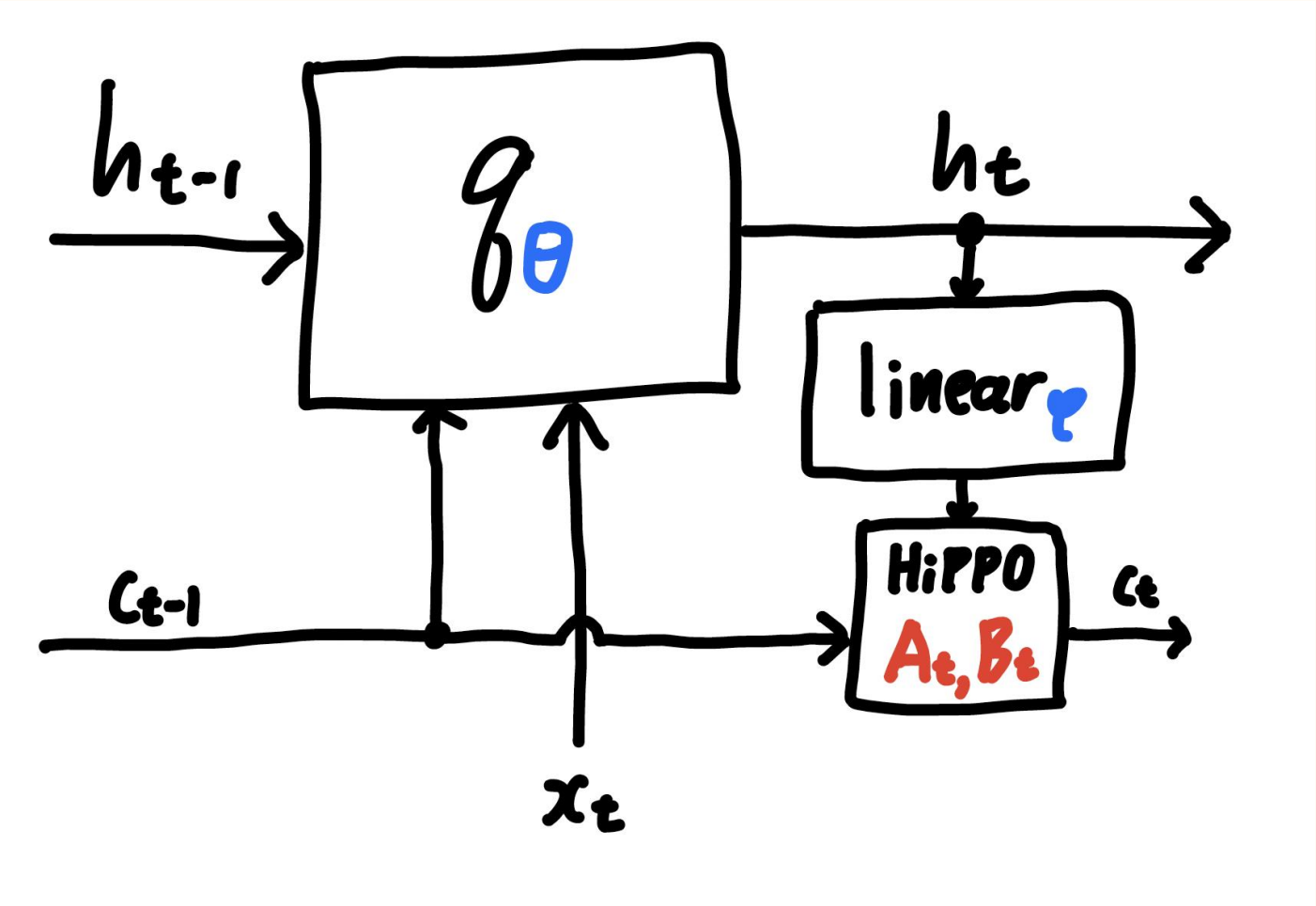
The GBT discretization with $\alpha \in (0,1)$, $t = \Delta t \cdot k$, $c_k = c(\Delta t \cdot k)$, and $f_k = f(\Delta t \cdot k)$ gives us

$$c_{k+1} = \underbrace{\left(I - \frac{1}{k+1}\alpha A\right)^{-1}\left(I + \frac{1}{k}(1-\alpha)A\right)}_{=\bar{A}_k} c_k + \underbrace{\frac{1}{k}\left(I - \frac{1}{k+1}\alpha A\right)^{-1} B}_{=\bar{B}_k} f_k,$$

So,

$$c_{k+1} = \bar{A}_k c_k + \bar{B}_k f_k$$

HIPPO+RNN architecture

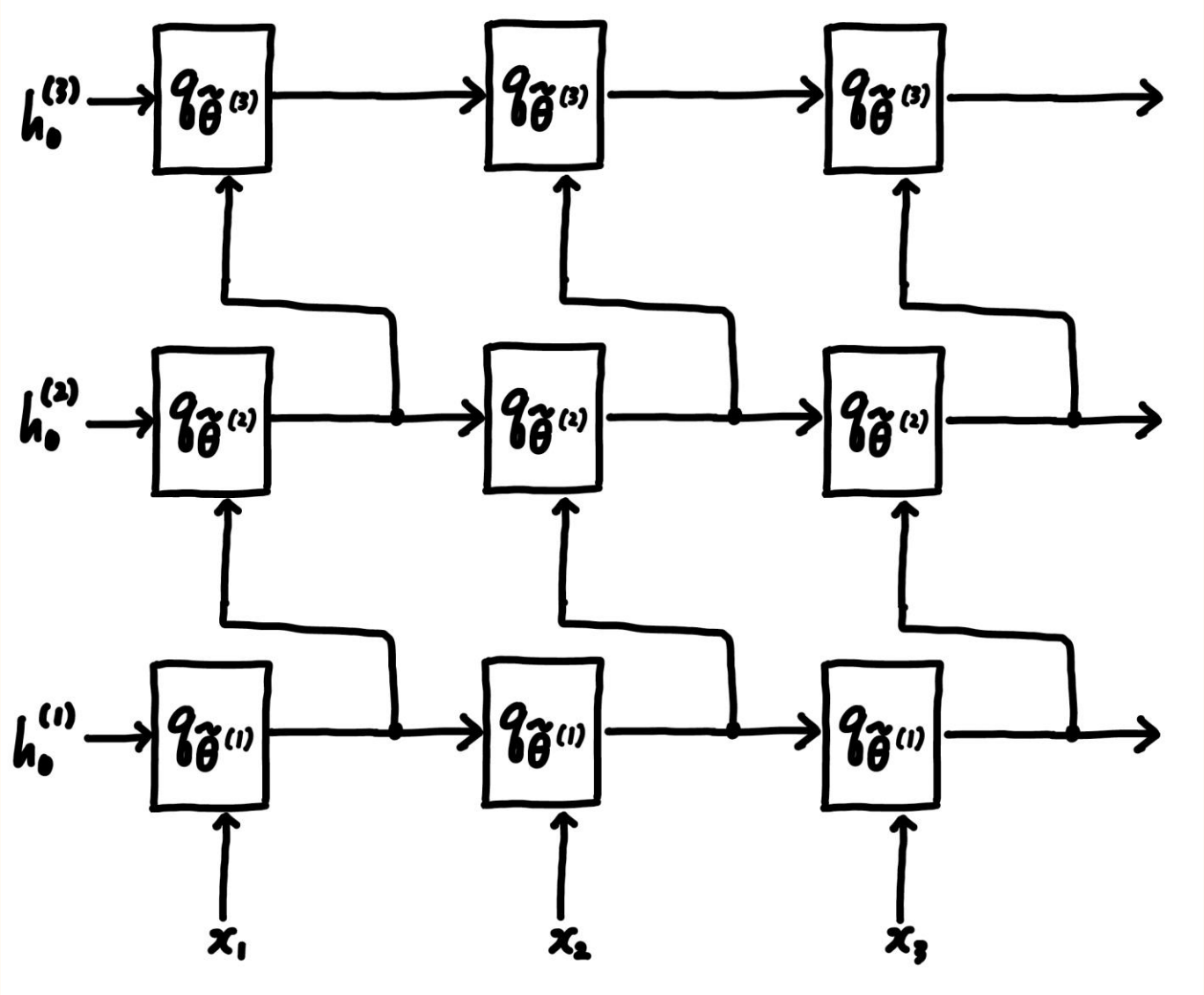


$$h_t = q_\theta(h_{t-1}, c_{t-1}, x_t) \in \mathbb{R}^d$$
$$f_t = Ch_t + d \in \mathbb{R}^1$$
$$c_t = A_t c_{t-1} + B_t f_t \in \mathbb{R}^N$$

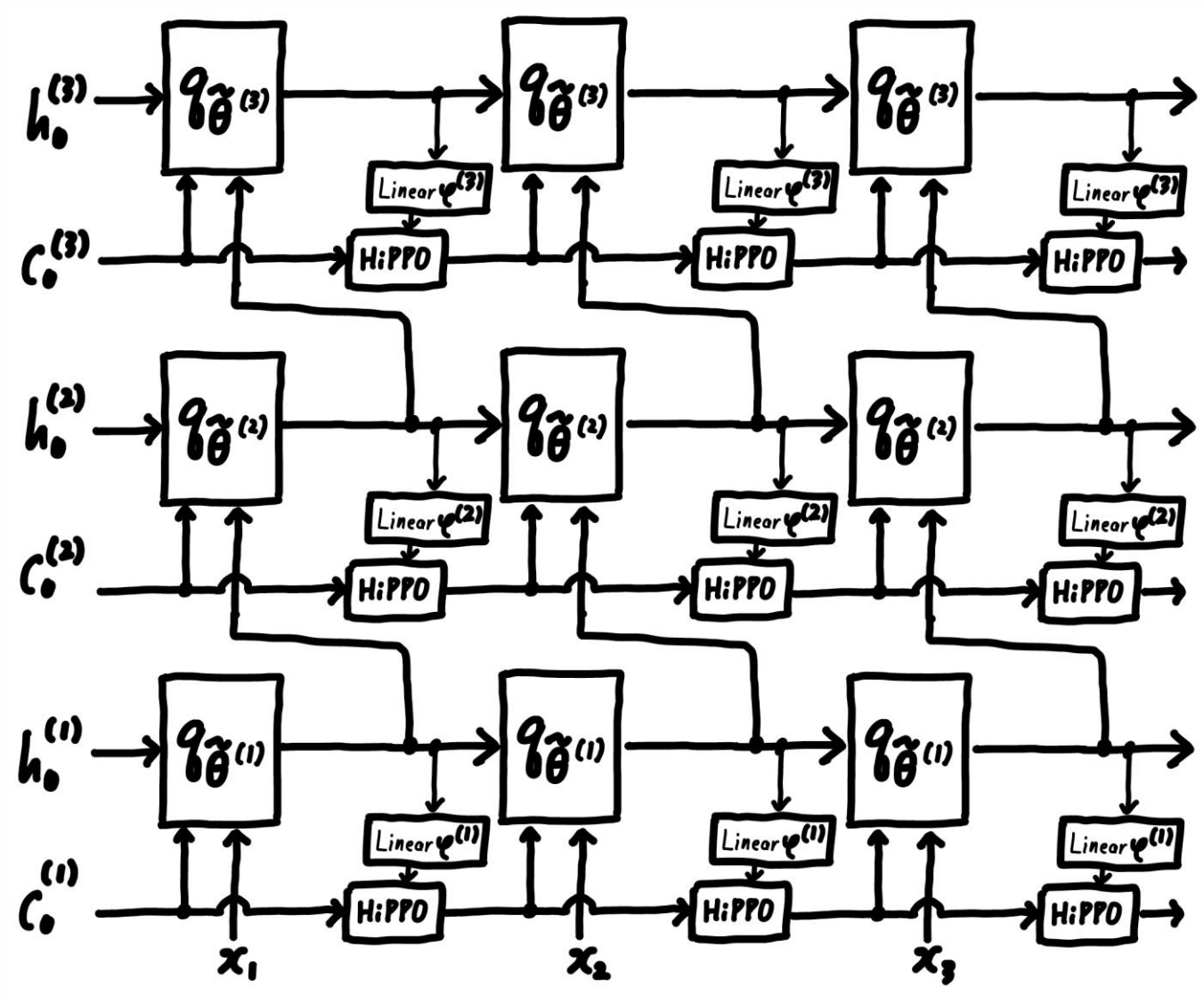
A_t and B_t are not trainable.

If we randomly initialize A_t and B_t and train it, this would be no different from a standard RNN.

RNN without HiPPO



HIPPO+RNN architecture



Background: Continuous-time Fourier transform and convolution theorem

Continuous-time forward and inverse Fourier transform:

$$\hat{f}(\omega) = \mathcal{F}[f](\omega) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i \omega t}\, dt, \qquad f(t) = \mathcal{F}^{-1}[\hat{f}](t) = \int_{-\infty}^{\infty} \hat{f}(\omega) e^{2\pi i \omega t}\, d\omega$$

Continuous-time convolution:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-s) g(s)\, ds$$

Convolution theorem) Convolution (in t) is pointwise multiplication in Fourier domain (in ω)

$$f * g = \mathcal{F}^{-1}\Big[\mathcal{F}[g] \cdot \mathcal{F}[g]\Big]$$

Background: Discrete Fourier transform

Discrete forward and inverse Fourier transform:

$$F = \mathcal{F}[f], \qquad f = \mathcal{F}^{-1}[F]$$

$$F_k = \sum_{n=0}^{L-1} f_n e^{-i2\pi \frac{kn}{N}}, \quad \text{for } k = 0, 1, \ldots, L-1 \qquad f_n = \frac{1}{L} \sum_{k=0}^{L-1} F_k e^{-i2\pi \frac{kn}{N}}, \quad \text{for } n = 0, 1, \ldots, L-1$$

Given $f \in \mathbb{R}^L$, computing $F \in \mathbb{C}^L$, requires $\mathcal{O}(L \log L)$ operations using the FFT, a divide-and-conquer algorithm. (Naïve implementation of sum above requires $\mathcal{O}(L^2)$ operations.)

Background: Discrete circular convolution

For $f, g \in \mathbb{R}^L$, the discrete circular convolution is defined as

$$(f \circledast g)[n] = \sum_{m=0}^{L-1} f[m]g[n-m] = \sum_{m=0}^{L-1} f[n-m]g[m], \qquad \text{for } n = 0, 1, \ldots, L-1$$

where $g[n-m] = g[n-m+L]$ if $n - m < 0$, i.e., we wrap around the index if the index is out of bounds. Computing $f \circledast g \in \mathbb{R}^L$ requires $\mathcal{O}(L^2)$ operations with a direct implementation of the definition.

Convolution theorem)

$$f \circledast g = \mathcal{F}^{-1}\Big[\mathcal{F}[g] \cdot \mathcal{F}[g]\Big]$$

Using the convolution theorem, $f \circledast g$ can be evaluated with $\mathcal{O}(L \log L)$ operations.

Background: Discrete non-circular convolution

For $f, g \in \mathbb{R}^L$, the discrete (non-circular) convolution is defined as

$$(f * g)[n] = \sum_{m=0}^{L-1} f[m]g[n-m] = \sum_{m=0}^{L-1} f[n-m]g[m], \qquad \text{for } n = 0, 1, \ldots, L-1$$

where $g[n-m] = 0$ if $n - m < 0$, i.e., the value is 0 when the index is out of bounds.

The discrete non-circular convolution doesn't have its own nice convolution theorem. However, $f * g$ can be evaluated with $\mathcal{O}(L \log L)$ operations using

$$f * g = \left(\begin{bmatrix} f \\ 0_L \end{bmatrix} \circledast \begin{bmatrix} 0_L \\ g \end{bmatrix} \right)_{0:L}$$

Linear State-Space Layers (LSSL)

RNN and Transformers are seq2seq models that transform a sequence $(u_n)_{n=0}^{L-1}$ to $(y_n)_{n=0}^{L-1}$ every layer.

Linear State-Space Layer (LSSL) defines a sequence-to-sequence (or function-to-function) transformation based on HiPPO-style linear dynamical systems.

Each layer maps $\{u(t)\}_{t\in[0,T]} \mapsto \{y(t)\}_{t\in[0,T]}$ via

$$\dot{x}(t) = Ax(t) + Bu(t) \in \mathbb{R}^n$$
$$y(t) = Cx(t) \in \mathbb{R}$$

A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, Combining recurrent, convolutional, and continuous-time models with linear state space layers, *NeurIPS*, 2021.

Continuous-time LSSL

Consider

$$\dot{x}(t) = Ax(t) + Bu(t) \in \mathbb{R}^n$$
$$y(t) = Cx(t) \in \mathbb{R}$$

for $t \in [0, T]$ with initial condition $x(0) = 0 \in \mathbb{R}^n$, which implies $y(0) = 0$.
So $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times 1}$, and $C \in \mathbb{R}^{1 \times n}$.

The solution is

$$y(t) = \int_0^t \left(Ce^{(t-s)A} B \right) u(s)\, ds = \int_{-\infty}^{\infty} K(t-s) u(s)\, ds = (K * u)(t)$$

where

$$K(t) = \begin{cases} Ce^{tA}B & \text{for } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad u(t) = \begin{cases} u(t) \geq 0 & \text{for } t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Initialization of A and B

Initialization of A is very important. A is initialized to so-called HiPPO LegS matrix

$$A_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

In the original HiPPO paper, LegS defines a linear time *variant* dynamical system, but the convolution formulation of LSSL requires a linear time *invariant* dynamical system. Nevertheless the LegS initialization is used.

B is randomly initialized.

Further interpretation of A-initialization is provided in: A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré, How to train your HiPPO: State space models with generalized orthogonal basis projections. *ICLR*, 2023.

Continuous-time to discrete-time LSSL

Use bilinear discretization:

$$x(t+\Delta t) = \underbrace{\left(I - \tfrac{\Delta t}{2}A\right)^{-1}\left(I + \tfrac{\Delta t}{2}A\right)}_{\overline{A}} x(t) + \underbrace{\Delta t\left(I - \tfrac{\Delta t}{2}A\right)^{-1}B}_{\overline{B}} u(t)$$

So the continuous-time dynamical system is discretized to

$$x_k = \overline{A}x_{k-1} + \overline{B}u_k$$
$$y_k = Cx_k$$

Discrete-time LSSL

$$u = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{L-1} \end{bmatrix} \in \mathbb{R}^L, \quad y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{L-1} \end{bmatrix} \in \mathbb{R}^L$$

Consider

$$x_k = \overline{A}x_{k-1} + \overline{B}u_k$$
$$y_k = Cx_k$$

for $k = 0, \ldots, L-1$ with initial condition $x_{-1} = 0 \in \mathbb{R}^n$.
So $\bar{A} \in \mathbb{R}^{n \times n}$, $\bar{B} \in \mathbb{R}^{n \times 1}$, and $C \in \mathbb{R}^{1 \times n}$.

The solution is

$$y_k = C\overline{A}^k\overline{B}u_0 + C\overline{A}^{k-1}\overline{B}u_1 + \cdots + C\overline{B}u_k, \qquad \text{for } k = 0, 1, \ldots, L-1$$

Alternatively,

$$y = \mathcal{K}_L * u$$

where

$$\mathcal{K}_L = (CB, CAB, CA^2B, \ldots, CA^{L-1}B) \in \mathbb{R}^L$$

Computation of $\mathcal{K}_L$

Naïve Computation of $\mathcal{K}_L$:

$O(N^2 L)$ cost $O(L)$ steps $O(N + L)$ memory

Efficient Computation of K using work-preserving prefix-scan:

$O(N^2 L)$ cost $O(\log L)$ steps $O(NL)$ memory

Prefix-scan: Standard algorithm for cumulative sums

$$x \in \mathbb{R}^n \mapsto cumsum(x) = y \text{ i.e., } , y_i = \sum_{j \leq i} x_j$$

More generally, $\alpha_1, \cdots, \alpha_L$ and binary associative operator $\otimes$ and an identity element, $\beta_i = \alpha_1 \otimes \cdots \otimes \alpha_i$ can be efficiently computed with prefix-scan. More on this later.

Multi-head LSSL (MH-LSSL)

- LSSL defines a linear transformation $\{u_k\}_{k=0}^{L-1} \mapsto \{y_k\}_{k=0}^{L-1}$. Since $u_k, y_k \in \mathbb{R}^H$, we really have

$$\{u_k^{(1)}, \dots, u_k^{(H)}\}_{k=0}^{L-1} \mapsto \{y_k^{(1)}, \dots, y_k^{(H)}\}_{k=0}^{L-1}$$

- $A \in \mathbb{R}^{N \times N}$ is shared globally (not trained).
- Each "head" $1, \dots, H$ has individual $B^{(1)}, \dots, B^{(H)} \in \mathbb{R}^{N \times 1}$.
- Each "head" $h = 1, \dots, H$ has M channels each $C^{(h,1)}, \dots, C^{(h,M)} \in \mathbb{R}^{1 \times N}$.
- Full architecture uses position-wise GeLU nonlinearity. (Otherwise architecture is linear.)
- The MH channels are projected back down to H channels with a position-wise linear layer.

Multi-head LSSL (MH-LSSL)

Stacked LSSL

- Layer norm and residual connection is used. (Both post-norm and pre-norm.)
- Unlike TF, position-wise MLP is not used.

Stacked LSSL

Trainable parameters (for “LSSL-fixed”)

C is trainable.

A, B, and Δt is are fixed and not trained. This allows Krylov matrix to be pre-computed.

$$K(\bar{A}, \bar{B}) = (\bar{B}, \bar{A}\bar{B}, (\bar{A})^2\bar{B}, (\bar{A})^3\bar{B}, \dots, (\bar{A})^{L-1}\bar{B}) \in \mathbb{R}^{N \times L}$$

So we have

$$\mathcal{K}_L = CK(\bar{A}, \bar{B})$$

Computational cost

Computational cost for LSSL-fixed:

- Parameters: $\mathcal{O}(HMN)$ in C
- Training: $\mathcal{O}(BL\log(L)HM)$ for convolution. ($\mathcal{K}_L$ is pre-computed and fixed.)
- $\mathcal{O}(BL\log(L)HM)$ for
- Memory: $\mathcal{O}(LHN)$ to store the Krylov matrix. $\mathcal{O}(BLH)$ for inputs/outputs.
- Inference: $\mathcal{O}(HMN^2)$ for matrix-vector multiplication by $\bar{A}$.

LSSL is both recurrent & convolutional

Also, LSSL trained on one sampling rate can be used on audio with another sampling rate.

XXX