Generative alternatives

Skorik Sergey

MIPT, 2023

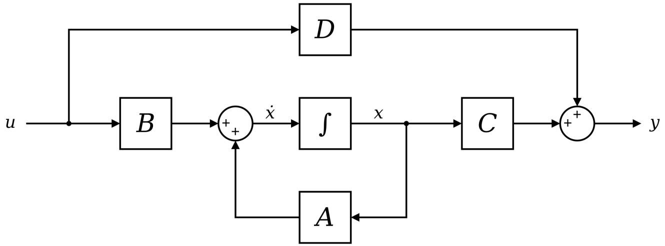
October 14, 2023

Backgrounds

State-space representation

The most general state-space representation of a linear system with p inputs, q outputs and n state variables is written in the following form:

$$\begin{cases} \dot{x}(t) = \mathbf{A}(t)x(t) + \mathbf{B}(t)u(t) \\ y(t) = \mathbf{C}(t)x(t) + \mathbf{D}(t)u(t) \end{cases} \tag{1}$$



Backgrounds

State-Space representation

State-Space representation is a mathematical model of a physical system. For example, for mass spring damper system (https://cookierobotics.com/008/) the following equations describes them:

$$
\begin{cases}
\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \cdot \begin{bmatrix} u(t) \end{bmatrix} \\
\begin{bmatrix} y(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}
\end{cases}
\tag{2}
$$

eal-world dynamics might not be fully known or may be subject to uncertainty. In such cases, A is used to capture the probabilistic or stochastic nature of the state dynamics. Instead of being a fixed matrix, A is modeled as a distribution over possible state transitions $A \sim \mathcal{N}(\mu, \Sigma)$.

HiPPO Framework

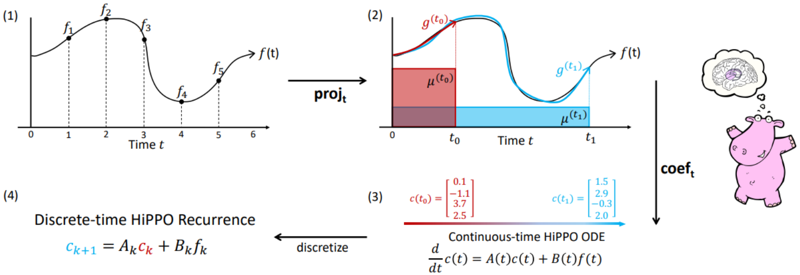


Figure 1: **Illustration of the HiPPO framework.** (1) For any function f, (2) at every time t there is an optimal projection $g^{(t)}$ of f onto the space of polynomials, with respect to a measure $\mu^{(t)}$ weighing the past. (3) For an appropriately chosen basis, the corresponding coefficients $c(t) \in \mathbb{R}^N$ representing a compression of the history of f satisfy linear dynamics. (4) Discretizing the dynamics yields an efficient closed-form recurrence for online compression of time series $(f_k)_{k\in\mathbb{N}}$.

HiPPO Framework

Connection between SSR and HiPPO

The composition $coef \circ proj$ is called hippo, which is an operator mapping a function f to the optimal projection coefficients c, i.e. $(hippo(f))(t) = coef_t(proj_t(f))$. The coefficient function $c(t) = coef_t(proj_t(f))$ has the form of an ODE satisfying $\dot{c}(t) = A(t)c(t) + B(t)f(t)$

SSM setup

SSM setup

The state space model maps a 1-D input signal $u(t)$ to an N-D latent state $x(t)$ before projecting to a 1-D output signal $y(t)$

$$\begin{cases} \dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \\ y(t) = \mathbf{C}x(t) \end{cases} \tag{3}$$

The discrete SSM is

$$\begin{cases} x_k = \bar{\mathbf{A}}x_{k-1} + \bar{\mathbf{B}}u_k, \quad \bar{\mathbf{A}} = (\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}(\mathbf{I} + \Delta/2 \cdot \mathbf{A}) \\ y_k = \bar{\mathbf{C}}x_k, \quad \bar{\mathbf{B}} = (\mathbf{I} - \Delta/2 \cdot \mathbf{A})^{-1}\Delta\mathbf{B}, \quad \bar{\mathbf{C}} = \mathbf{C} \end{cases} \tag{4}$$

The fundamental bottleneck in computing the discrete-time SSM (4) is that it involves repeated matrix multiplication by $\bar{\mathbf{A}}$. For example, naively as in the LSSL involves L successive multiplications by $\bar{\mathbf{A}}$ requiring $\mathcal{O}(N^2 \cdot L)$ operations and $\mathcal{O}(NL)$ space.

HiPPO

HiPPO

HiPPO specifies a class of certain matrices $\mathbf{A} \in \mathbb{R}^{N\times N}$ that when incorporated into (3), allows the state $x(t)$ to memorize the history of the input $u(t)$.

$$\textbf{(HiPPO Matrix)} \quad \mathbf{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases} \tag{5}$$

The ideal scenario is when the matrix $\mathbf{A}$ is diagonalizable by a perfectly conditioned (i.e., unitary) matrix. By the Spectral Theorem of linear algebra, this is exactly the class **normal matrices**. In particular, it does not contain the HiPPO matrix.

S4

Table 1: Complexity of various sequence models in terms of sequence length ($\boldsymbol{L}$), batch size ($\boldsymbol{B}$), and hidden dimension ($\boldsymbol{H}$); tildes denote log factors. Metrics are parameter count, training computation, training space requirement, training parallelizability, and inference computation (for 1 sample and time-step). For simplicity, the state size N of S4 is tied to H. Bold denotes model is theoretically best for that metric. Convolutions are efficient for training while recurrence is efficient for inference, while SSMs combine the strengths of both.

	Convolution[3]	Recurrence	Attention	S4
Parameters	LH	$\boldsymbol{H^2}$	$\boldsymbol{H^2}$	$\boldsymbol{H^2}$
Training	$\boldsymbol{\tilde{L}H(B+H)}$	BLH^2	$B(L^2H+LH^2)$	$\boldsymbol{BH(\tilde{H}+\tilde{L})+B\tilde{L}H}$
Space	$\boldsymbol{BLH}$	$\boldsymbol{BLH}$	$B(L^2+HL)$	$\boldsymbol{BLH}$
Parallel	**Yes**	No	**Yes**	**Yes**
Inference	LH^2	$\boldsymbol{H^2}$	L^2H+H^2L	$\boldsymbol{H^2}$

Kalman Filter

The more general formulation of the state space model described in the previous section as an observation equation

$$y_t = A_t x_t + V_t$$

and a state equation

$$x_t = \Theta x_{t-1} + W_t$$

where y_t is a $p \times 1$ vector, x_t is a $k \times 1$ vector, A_t is a $p \times k$ matrix and Θ is $k \times k$ matrix. We can think of $V_t \sim \mathcal{N}(0, S)$ and $W_t \sim \mathcal{N}(0, R)$.

Given an initial state x_0^0 and P_0^0, the prediction equations are (analogous to above)

$$\begin{aligned} x_1^0 &= \Theta x_0^0 \\ P_1^0 &= \Theta P_0^0 \Theta' + R \end{aligned}$$

and the updating equations are, given a new observation y_1,

$$\begin{aligned} x_1^1 &= x_1^0 + K_1(y_1 - A_1 x_1^0) \\ P_1^1 &= (I - K_1 A_1) P_1^0 \end{aligned}$$

where

$$K_1 = P_1^0 A_1' (A_1 P_1^0 A_1' + S)^{-1}.$$

In general, given the current state x_{t-1}^{t-1} and P_{t-1}^{t-1} and a new observation y_t, we have

$$\begin{aligned} x_t^{t-1} &= \Theta x_{t-1}^{t-1} \\ P_t^{t-1} &= \Theta P_{t-1}^{t-1} \Theta' + R \end{aligned}$$