Exploring Applications of State Space Models and Advanced Training Techniques in Sequential Recommendations

Baderko Makar[1] Kulibaba Stepan[1] Obozov Mark[1] Nikolay Kutuzov Alexander Gasnikov

[1] Equal contribution

USGM adaptive method

```
Algorithm Universal Stochastic Gradient Method
1: Initialize: $x_0 \in \mathrm{dom}\, f$, $D > 0$, $H_0 := 0$, $g_0 \sim \hat{g}(x_0)$.
2: for $k = 0, 1, \dots$ do
3:     $x_{k+1} = \arg\min_{x \in \mathrm{dom}\, f}\{\langle g_k, x\rangle + \frac{H_k}{2}\|x - x_k\|^2\}$.
4:     $g_{k+1} \sim \hat{g}(x_{k+1})$.
5:     $H_{k+1} := H_k + \frac{[\hat{\beta}_{k+1} - \frac{1}{2}H_k r_{k+1}^2]_+}{D^2 + \frac{1}{2}r_{k+1}^2}$,
6:     where $r_{k+1} = \|x_{k+1} - x_k\|$, $\hat{\beta}_{k+1} = \langle g_{k+1} - g_k, x_{k+1} - x_k\rangle$.
7: end for
```

Notation:

The authors of the original paper consider an upper bound for the stochastic approximation of the symmetrized Bregman distance for points x_k and x_{k+1} ($\hat{\beta}_{k+1}$) as follows:

$$\hat{\beta}_{k+1} = \langle f'(x_{k+1}) - f'(x_k) + \Delta_{k+1}, x_{k+1} - x_k\rangle \le L \tag{1.3}$$

where $f'(x_k) := \mathbb{E}_{\xi_k}[g_k] \in \partial f(x_k)$, $\Delta_{k+1} := \delta_{k+1} - \delta_k$ with $\delta_k := g_k - f'(x_k)$ being the error of the stochastic gradient (such that $\mathbb{E}\|\delta_k\|^2 \le \sigma^2$), and $\sigma_{k+1} := \|\Delta_{k+1}\|$.

Adaptive batching

In stochastic optimization, the variance of the gradient estimates reduces as the batch size increases. Specifically, the variance decreases proportionally to $\frac{1}{B}$. Concurrently, the standard deviation (σ) of the gradient estimates decreases at the rate of $\frac{1}{\sqrt{B}}$. From here 1.3 , we can derive the equation 2.1. Thus, the variance and standard deviation of the gradient estimations decreases as the batch size increases.

$$\hat{\beta}_{k+1} \le L_\nu r_{k+1}^{1+\nu} + \frac{\sigma_{k+1} r_{k+1}}{\sqrt{B}}, \tag{2.1}$$

As the coefficient $\hat{\beta}_{k+1}$ is computed on every iteration of the algorithm, and the batch size is known. We can use the Weighted Least Squares method (with loss defined by the equation 2.2) in linearized axes ($\frac{1}{\sqrt{B}}$ as the X axis and the $\hat{\beta}_{k+1}$ as the Y axis) to compute the values of $L_\nu r_{k+1}^{1+\nu}$ and $\sigma_{k+1} r_{k+1}$. The intercept and the slope of the linearized function respectively.

$$L = \sum_{k=0}^{K} (F(c_1, c_2, B_k) - \hat{\beta}_{k+1})^2 \cdot (1 - \alpha)^{K-k}, \tag{2.2}$$

where F is the sought function $F = c_1 + \frac{c_2}{\sqrt{B}} \approx \hat{\beta}_{k+1}$, c_1 and c_2 are the values of $L_\nu r_{k+1}^{1+\nu}$ and $\sigma_{k+1} r_{k+1}$ respectively, predicted by the WLS algorithm. The value of alpha was empirically set to $\alpha = 0.01$, s.t. the first points almost diminish.

Batch size properties

When the epoch ends, we lower the batch size ($B_i := \frac{B_{i-1_k}}{\lambda}$), since the perfect batch size (B_i^*) could have changed since the last epoch. However, if it's bigger than $\frac{B_{i-1_k}}{\lambda}$, the algorithm will increase it during the next iterations. By doing so, we give the algorithm the ability to not continue training with excessive resources if the calculated batch size turns out to be too big in the current circumstances or to increase it even more if needed.

Since $B_k := B_{k-1} + B_0$, $B_k = m_k\ B_0 : \forall k\ (m_k \in \mathbb{Z})$. That means we can use the original Dataloader class from PyTorch ((4)) wihout any modifications and take m_k batches of initial size (B_0) on each iteration.

Main contributions

◇ **New SOTA sequential recomendation model**

- ◇ We applied quasiseparable mixer framework on the case of sequential recommendations. The SOTA sequential recommendation model has been obtained.
- ◇ We examined various applications of selected spaces to the sequential recommendation problem..

◇ **Advanced training and optimization techniques.**

- ◇ ORPO preference optimization application to LLM recommedner systems
- ◇ First USGM based(**?**) method with non-emperical adaptive batch size choice.

Hydra4Rec framework

Architecture and matrix mixers

Hydra fully utilizes the matrix mixer framework to explore a novel bidirectional sequence mixer. To achieve linear complexity, a special class of quasiseparable matrix is considered. A matrix $\mathbf{M}$ is quasiseparable if every label element m_{ij} satisfies:

$$m_{ij} = \begin{cases} \overrightarrow{\mathbf{c}_i^T}\overrightarrow{\mathbf{A}_{i:j}^\times}\overrightarrow{\mathbf{b}_j}, & \text{if } i > j \\ \delta_i, & \text{if } i = j \\ \overleftarrow{\mathbf{c}_j^T}\overleftarrow{\mathbf{A}_{j:i}^\times}\overleftarrow{\mathbf{b}_i}, & \text{if } i < j \end{cases}, \tag{1}$$

where each δ_i is a scalar, $_{i,i} \in \mathbb{R}^{N\times 1}$, and $_i \in \mathbb{R}^{N\times N}$. Quasiseparable matrices can be used as matrix mixers. This framework is called **Hydra**, which achieves $O(L)$ complexity.

To apply the sequence-to-sequence potential of the Hydra block (2), we provide a custom Hydra layer that combines the Hydra block with a standard feed-forward network. The main part of our standard architecture consists of Hydra layers. We have found that Hydra layers work more effectively than standard Mamba layers in several cases. Then, we use the same PFFN that was introduced in Mamba4Rec.(3)

$$PFFN(H) = GELU(HW^{(1)} + h^{(1)})W^{(2)} + b^{(2)}$$

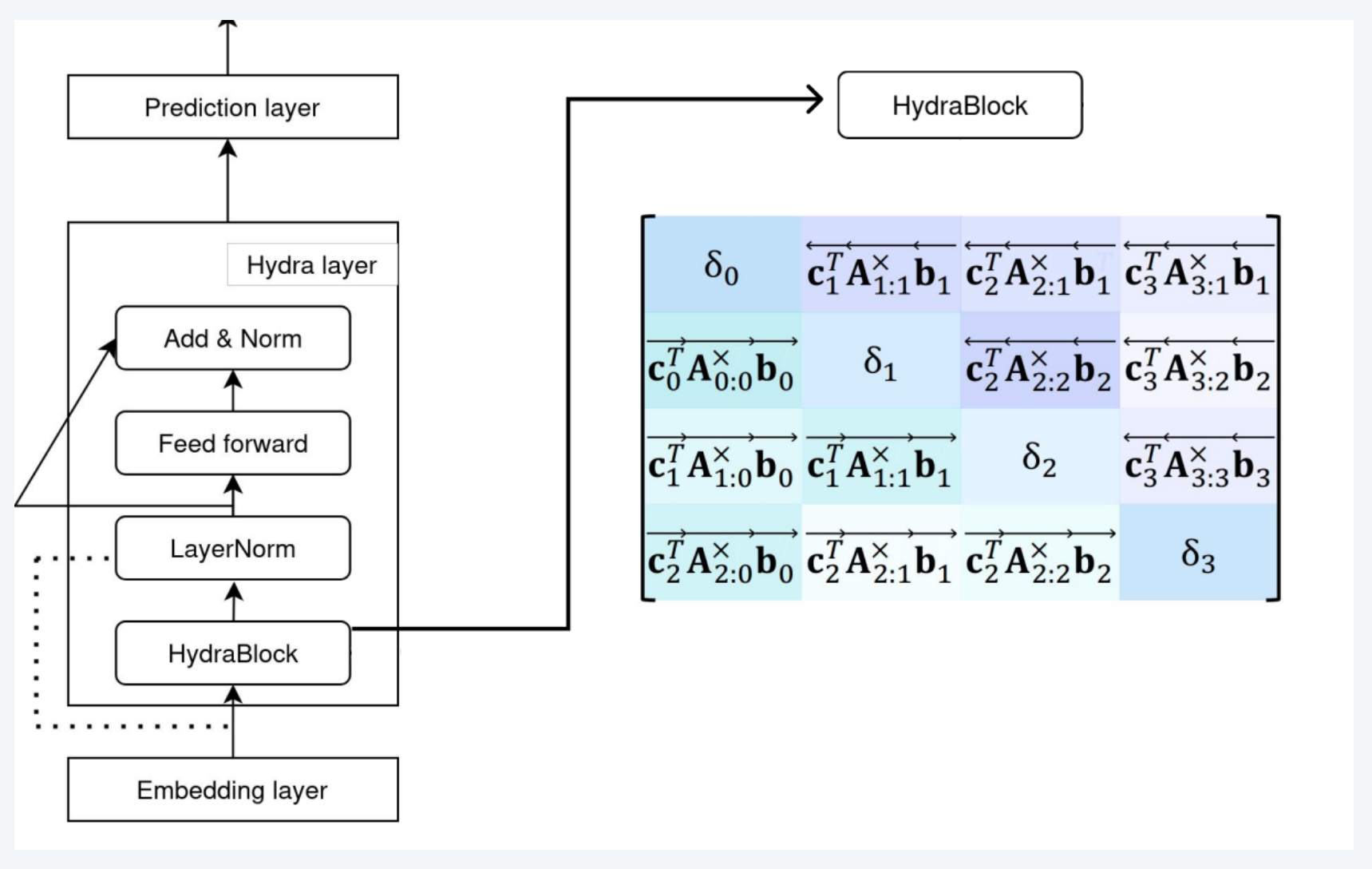

Figure: Hydra4Rec architecture

Where $W^{(1)} \in \mathbb{R}^{D\times 4D}$, $W^{(2)} \in \mathbb{R}^{D\times 4D}$, $b^{(1)} \in \mathbb{R}^D$ are parameters of two dense layers and we use the GELU activation.

Such framework provide us significant latency boost!

Monolitic No Reference Model Optimization

ORPO application

Odds Ratio Preference Optimization (`ORPO`), (1) - is a novel preference optimization framework that consolidates an odds ratio-based penalty. The odds of generating the output sequence y from a given input sequence x are defined by:

$$\log P_\theta(y|x) = \frac{1}{m}\sum_{t=1}^{m} \log P_\theta(y_t|x, y < t) \tag{2}$$

$$\mathbf{odds}_\theta(y|x) = \frac{P_\theta(y|x)}{1 - P_\theta(y|x)} \tag{3}$$

The odds ratio of the chosen response y_w over the rejected response y_l, $\mathbf{OR}_\theta(y_w, y_l)$, indicates how much more likely it is for the model θ to generate y_w than y_l given input x.

$$\mathbf{OR}_\theta(y_w, y_l) = \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)} \tag{4}$$

The objective function of `ORPO` consists of two components: 1) supervised fine-tuning (SFT) loss ($\mathcal{L}_{SFT}$); 2) relative ratio loss ($\mathcal{L}_{OR}$).

$$\mathcal{L}_{ORPO} = \mathbb{E}_{(x, y_w, y_l)}\left[\mathcal{L}_{SFT} + \lambda \cdot \mathcal{L}_{OR}\right] \tag{5}$$

We created a pair data set, where each pair consists of a winner and a loser. It consists of 2 parts. The first part was created from the last 2 items from the user history where the winner is the item with the higher rating and the loser is the opposite. To create the second part, we used negative sampling training LightFM model and took the last user's movie as the winner and the worst recommendation of LightFM as the loser.

ORPO Expiremental Setup

For the ORPO procedure we used lr = 8e-6, ORPO we used very low learning rates compared to traditional SFT or even DPO. This value of 8e-6 comes from the original paper. beta = 0.1 An appendix from the original paper shows how it's been selected with an ablation study. We have used paged adamw optimizer with gradient accumulation steps = 4. We have trained model for 1 training epoch and evaluated every 0.2 steps, also we set warmup steps to 10. We have used linear scheduler type. Other parameters are dependent on capabilities of your resources. Training was conducted on 1 A100 80GB videocard.

Experiments

It is evident that Hydra shows metrics that are similar, sometimes better to those of Mamba, but its latency is four to five times lower. Simultaneously, models such as LlamaRec exhibit superior per- formance. However, their substantial parameter count may constrain their applicability in real-world situations. Metrics-wise and latency-wise, SSM-based models outperform the baseline SASRec model on average. According to the study, SSMs can outperform LLMs while still operating at a greater speed since they require less number of parameters.

Table: Amazon Reviews '23 Video Games

Model	HT@K	NDCG@K	MRR@K
SASRec	0.119	0.073	0.059
Mamba4Rec	0.107	0.062	0.048
MamRec	0.083	0.033	0.025
GPT4Rec	0.080	0.042	0.026
2Mamba4Rec	0.118	0.061	0.048
Hydra4Rec	0.112	0.059	0.044
LlamaRec	**0.150**	**0.098**	**0.064**

Table: MovieLens-1M

Model	HT@K	NDCG@K	MRR@K
SASRec	0.224	0.117	0.084
Mamba4Rec	0.303	0.178	0.139
MamRec	0.201	0.072	0.064
GPT4Rec	0.212	0.074	0.060
2Mamba4Rec	0.340	0.193	0.148
Hydra4Rec	**0.308**	**0.179**	**0.140**
LlamaRec	0.148	0.067	-

Training process experiments

Since the number of iterations of the algorithm with adaptive batching and of those, with which it was compared, differs, since the equality $N_{\text{iterations}} = \frac{\text{Dataset sise}}{\text{Batch size}} = const$ is not true for the proposed algorithm, a scaled measure, which is proportional to the epoch number is used as the X axis in the comparison plot, displayed in the Figure 1.

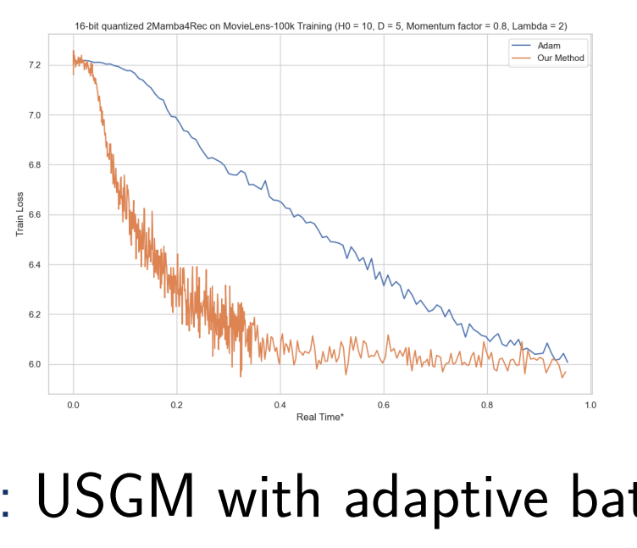

Figure: USGM with adaptive batching

ORPO procedure improvement is also pretty

References

[1] J. Hong, N. Lee, and J. Thorne. Orpo: Monolithic preference optimization without reference model, 2024.

[2] S. Hwang, A. Lahoti, T. Dao, and A. Gu. Hydra: Bidirectional state space models through generalized matrix mixers, 2024.

[3] C. Liu, J. Lin, J. Wang, H. Liu, and J. Caverlee. Mamba4rec: Towards efficient sequential recommendation with selective state space models, 2024.

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.