Group Meeting - October 1, 2021

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Original MMF vs. Random-index learnable MMF (1)

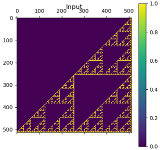
Original MMF:

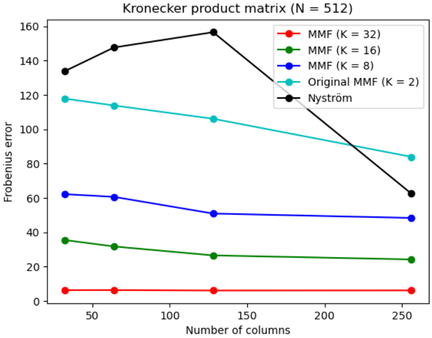
- $K = 2$
- For each rotation/resolution, random the first index, and then exhaustive search to find the optimal second index.

Random-index learnable MMF:

- $K > 2$
- For each rotation/resolution, every index is selected randomly.

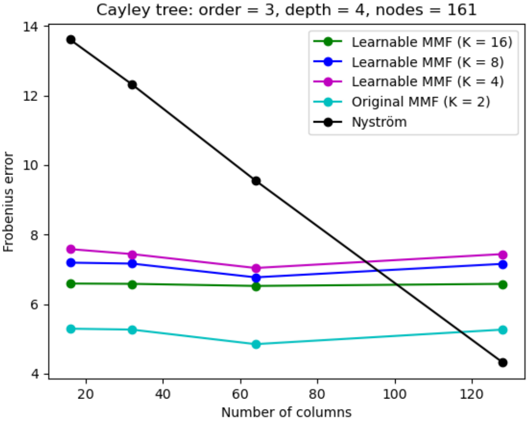
Original MMF vs. Random-index learnable MMF (2)





Bigger Ks, the better!

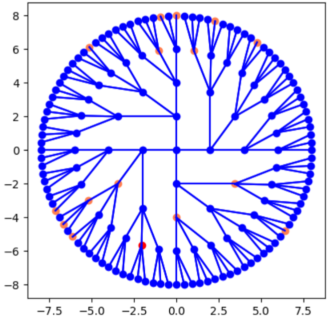
Original MMF vs. Random-index learnable MMF (3)

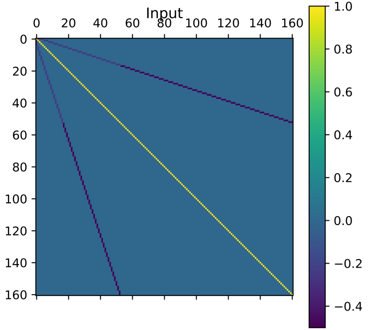


The original MMF with $K = 2$ with exhaustive search outperforms random-index learnable MMFs with $K > 2$.

Original MMF vs. Random-index learnable MMF (4)

In summary, the original MMF outperforms the random-index learnable one in the case of Cayley tree. Why does that happen?





There exists an optimal strategy of selecting a pair of nodes for Cayley tree: select 2 nodes of the same level that have exactly the same topology.

Reinforcement Learning (1)

Gradient policy network (REINFORCE): Use GNNs as RL agents (policy networks) to learn to select the sequence of indices, that is similar to learning to solve combinatorics problems (NP-hard).

function REINFORCE
  Initialise θ arbitrarily
  for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
    end for
  end for
  return θ
end function

In MMF, we need 2 RL policy networks: (i) one to select the wavelet index (also to drop it out), and (ii) another one to select the rest of $K - 1$ indices.

Algorithm 1 MMF learning algorithm optimizing problem 1

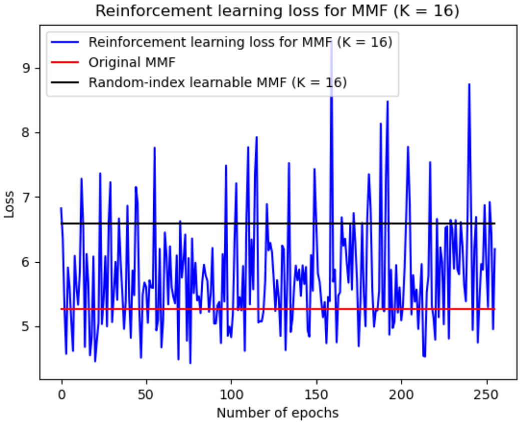
Given matrix $\boldsymbol{A}$, number of resolutions L, and constants k, γ, η, and ω.
Initialize the policy parameter θ at random.
while true **do**
  Start from state s_0 ▷ $s_0 \triangleq (\boldsymbol{A}, [n], 0)$
  Initialize $\mathbb{S}_0 \leftarrow [n]$ ▷ All rows/columns are active at the beginning.
  for $\ell = 0, .., L - 1$ **do**
    Sample action $a_\ell = (\mathbb{I}_{\ell+1}, \mathbb{T}_{\ell+1})$ from $\pi_\theta(a_\ell | s_\ell)$. ▷ See Section 4.3.
    $\mathbb{S}_{\ell+1} \leftarrow \mathbb{S}_\ell \setminus \mathbb{T}_{\ell+1}$ ▷ Eliminate the wavelet index (indices).
    $s_{\ell+1} \leftarrow (\boldsymbol{A}_{\mathbb{S}_{\ell+1}, \mathbb{S}_{\ell+1}}, \mathbb{S}_{\ell+1}, \ell + 1)$ ▷ New state with a smaller active set.
  end for
  Given $\{\mathbb{I}_\ell\}_{\ell=1}^L$, minimize objective 7 by Stiefel manifold optimization to find $\{\boldsymbol{O}_\ell\}_{\ell=1}^L$. ▷ $\boldsymbol{U}_\ell = \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell$
  for $\ell = 0, .., L - 1$ **do**
    Estimate the return g_ℓ based on Eq. 13, Eq. 14, and Eq. 15.
    $\theta \leftarrow \theta + \eta \gamma^\ell g_\ell \nabla_\theta \log \pi_\theta(a_\ell | s_\ell)$ ▷ REINFORCE policy update in Eq. 18
  end for
  Terminate if the average error of the last ω iterations increases. ▷ Early stopping.
end while

Reinforcement Learning (3)

The Algorithm 1 is expensive due to the Stiefel manifold optimization in line 11 to find the optimal rotations that are used to compute the rewards. In practive, I propose a 2-phase process that is more efficient:

- **Phase 1:** Reinforcement learning to find the sequence of indices as in Algorithm 1, but instead of Stiefel manifold, we just use the closed-form solutions to estimate the rewards.
- **Phase 2:** Given a sequence of indices found by Phase 1, we apply Stiefel manifold optimization with many iterations to actually find the optimal rotations accordingly.

Phase 1

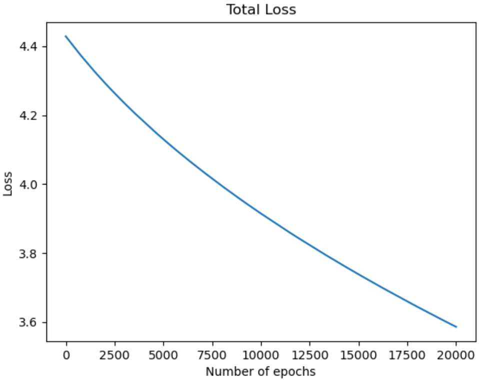


The red line is the baseline of the original MMF. The black line is the learnable MMF but with completely random indices. The RL training is unstable, but by-average it is better than the random choices.

Learning to solve combinatorics problems

- Because the problem is an **NP-hard** one, so the optimal solution is only guaranteed by an exhaustive brute-force search over all possibilities. Given a limited number of tries (each epoch is a try or sample sequence of the policy), the question is how close we can get to the optimal solution.
- In our case, the RL reached to a solution better than the original baseline after 10 tries.
- Fundamentally, learning to solve combinatorics problems is different from learning to solve convex problems: one has the discrete search space, the another one continuous. Maybe the convergence behavior is only observable when we solve convex problems and the search space is continuous.

Phase 2



The best loss found from Phase 1 is 4.4. Stiefel manifold optimization further improves into 3.6. It **can** go down further given more epochs.

In summary, after Phase 2, we get 32% improvement comparing to the original MMF (e.g, 3.6 vs 5.2) in the case of Cayley tree.