What if Neural Networks had SVDs?

Presented by team “Oldy 684”:

Elizaveta Kovtun
Olesya Kuznetsova
Dmitrii Korzh

Motivation

- Want to perform gradient descent on SVD matrices separately

$$W = U\Sigma V^T$$

$$\Sigma' = \Sigma - \eta \nabla_{\Sigma}$$
$$U' = U - \eta \nabla_{U}$$
$$V' = V - \eta \nabla_{V}$$

- To preserve orthogonality, decompose orthogonal in a product of Householder

$$U = \prod_{i=1}^{d} H_i$$

$$H_i = I - 2\frac{v_i v_i^T}{\|v_i\|_2^2}$$

Motivation

- Want to perform gradient descent with SVD-decomposition in a such way, that provides speed up

- Why? To train NN faster, accelerating matrix operations. Previous results didn’t show supposed results in practise.

Methods: FastH

- Increasing the degree of parallelization of an underlying matrix multiplication H · X
- $O(d^2 m)$ algorithm to compute matrix product $A = H_1 \cdots H_d X$
- **Forward pass**:
  Lemma : For any $H_1, \cdots, H_m$ *there exist* $W, Y \in \mathbb{R}^{d \times m}$

$$\text{s.t.} \quad H_1 \cdots H_m = I - 2WY^T$$

Methods: FastH

- **Forward pass**:
  1. Compute Y_i, W_i for $P_i = H_{(i+1)m+1} \cdots H_{im}$ in parallel - $O(dm^2)$
  2. Compute $A_i = A_{i+1} - 2W_i(Y_i^T A_{i+1})$ sequentially - $O(dm^2)$
- **Backward pass:** $O(d^2 m)$ algorithm to compute $\frac{\partial L}{\partial X}$ and $\frac{\partial L}{\partial v_k}$ for all v_k - Householder vectors

Experiment

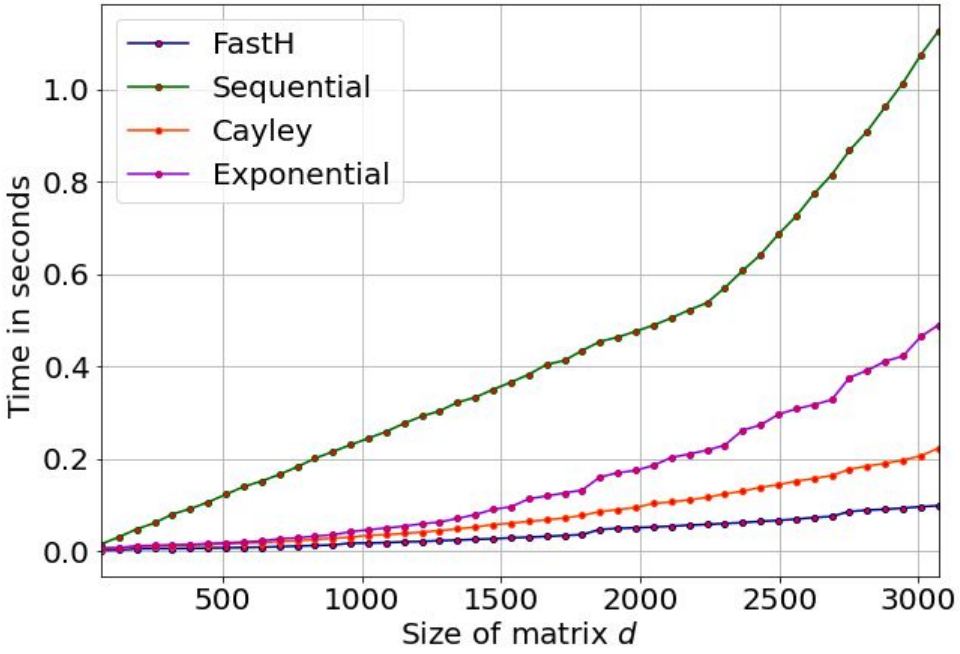
Input: V - square weight matrix, ϕ - function that reparameterizes V so ϕ(V) is orthogonal and we can perform gradient descent wrt. V

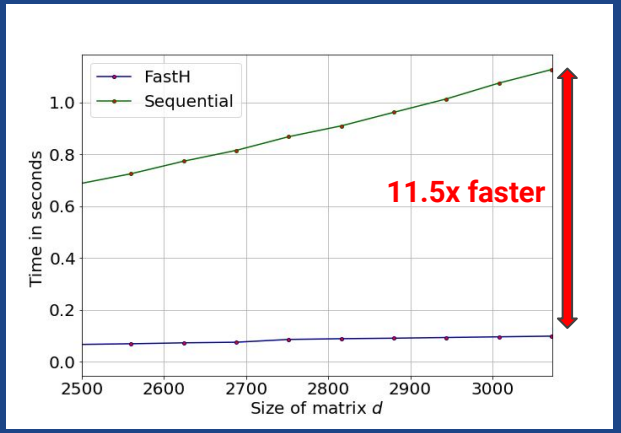
Considered ways to construct function ϕ:

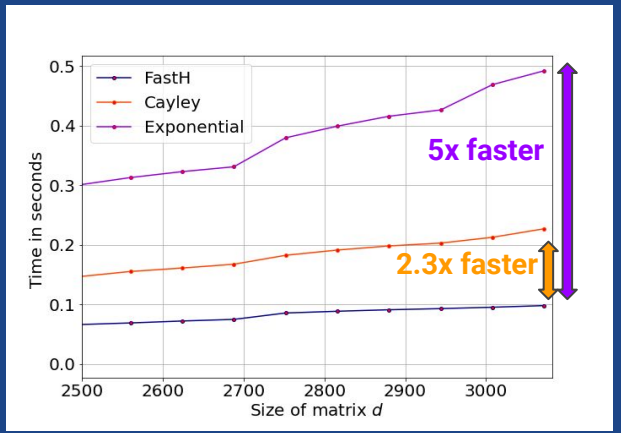
- Householder decomposition
- Matrix exponential $\phi_{exp} = e^{V}$
- Cayley map approach $\varphi_C(V) = (I - V)(I + V)^{-1}$

Goal: Compare the running time of a gradient descent step (computing $\phi(V)X$ and the gradients wrt. V) for FastH and alternative algorithms

Results of experiment







Spectral-RNN

- Problem: exploding gradient of RNN
- Solution: parameterize transition matrix W by its SVD and control $\|W\|_2$
- Also solves problem of vanishing gradient

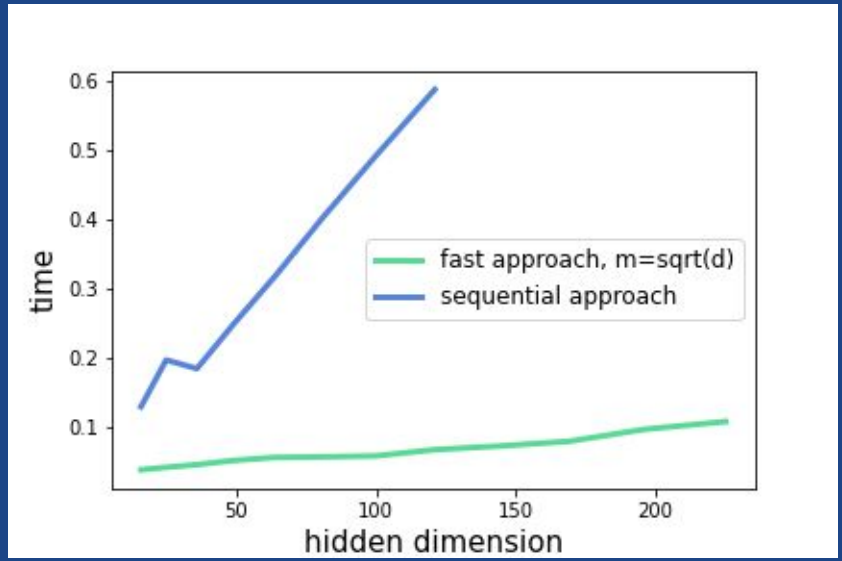
Spectral-RNN

- Comparison of time, consumed by one RNN loop for vanilla Spec-RNN and accelerated by FastH

- Expected acceleration

from $O(d)$ to $O\left(\frac{d}{m} + rm\right)$

Time (in seconds) for different hidden dimensions of RNN



Summary

- An efficient matrix multiplication parallelization on GPUs algorithm was studied

- Targets that were achieved:
  1. comparison of time consumption of a new and previous sequential algorithm;
  2. comparison of running time of RNN with SVD parametrization of transition matrix - fast and sequential approach

- TBD: more experiments with RNNs, different approaches of choosing m