Combining Adagrad and Sketched ALS for CP Tensor Decomposition

Chris Jerrett, Matthew Scanlon, Chris Vanderloo, Min Yue

Rensselaer Polytechnic Institute

December 11, 2020

Overview

What is a Tensor

Matrix Representation

A matrix over a field $\mathbb{F}$ is written as the rectangular grid of elements of $\mathbb{F}$.

Generalization

A tensor can be similarly represented as a multidimensional array of elements of $\mathbb{F}$. Thus a tensor is sometimes called the generalization of a matrix

Definitions

Order

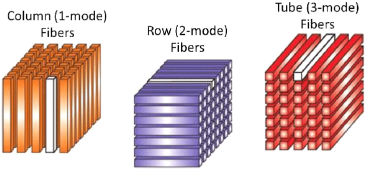
Equivalent to the dimensionality of the tensor. For example, a vector is an order-1 tensor, while a matrix is an order-2 tensor.

Fiber

A fiber of a tensor is a vector obtained by fixing all the indices of all except for one.

Mode n-Fiber

A fiber obtained by fixing all but the nth dimension.



Tensor Rank

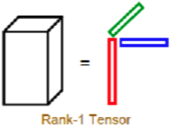
Let $\mathcal{X}$ be an order-n Tensor.

Rank 1 Tensors

The tensor $\mathcal{X}$ is rank 1 if $\mathcal{X}$ can be written as the outer product of n vectors. Thus $\mathcal{X} = \mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \cdots \otimes \mathbf{x}_n$.

Tensor Rank

The rank of tensor $\mathcal{X}$ is the minimum number of rank 1 tensors summed to obtain $\mathcal{X}$. The rank R of $\mathcal{X}$ is the minimum r for which $\mathcal{X} = \sum_{i=1}^{r} \mathbf{x}_{1,i} \otimes \mathbf{x}_{2,i} \otimes \cdots \otimes \mathbf{x}_{n,i}$



[1] We assume all tensors from now are in $\mathbb{R}^{I_1 \times I_2, \ldots I_n}$

Tensor Rank Decomposition

Let $\mathcal{X}$ be an order-n rank r Tensor.

CP Decomposition[1,2]

We want to find the value of
$\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, ..., \mathbf{x}_{1,n}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2} ..., \mathbf{x}_{2,n}, ..., \mathbf{x}_{r,1}, \mathbf{x}_{r,2}, ..., \mathbf{x}_{r,n}$.

Kruskal Form

For simplicity let X_i be the matrix with each jth column be the vector $\mathbf{x}_{i,j}$. Let $[\![X_1, X_2, ..., X_n]\!]$ be the sum of the outer product of the corresponding columns.

We want to find $X_1, X_2, ...X_n$, but this is a hard problem ☹

[1] Also sometimes called the CANDECOMP or PARAFAC decomposition

Optimization Problem

Let $\mathcal{X}$ be an order-n rank r tensor.

Instead solve optimization problem: $\min_{X_i \in \mathbb{R}^{I_i \times r}} \|\mathcal{X} - [\![X_1, X_2, ..., X_n]\!]\|_F$

This is problem is still hard, possibly ill-posed and non-convex but we can still try to solve it. Even computing the rank of a tensor is NP-Hard.

Low Rank Approximation

Let $\mathcal{X}$ be an order-n rank r Tensor.

Model the tensor $\mathcal{X}$ as a low-rank tensor with a small noise tensor, $\mathcal{X} = [\![\mathbf{X}_1, \mathbf{X}_2, ...\mathbf{X}_n]\!] + \epsilon$. Where the number of columns of $\mathbf{X}_1, \mathbf{X}_2, ...\mathbf{X}_n << r$.

Then try to recover $[\![\mathbf{X}_1, \mathbf{X}_2, ...\mathbf{X}_n]\!]$ as our low-rank approximation.

Data as Tensors

Temporal Data

Whenever data at a single point of time can be represented as a matrix the tensor allows us to also show how the data changes over time.

Multi-relational Data

Constructing data into tensor form can reveal the relationships in multiple ways. Such as in hyper graphs or complex social networks.

Decomposition Applications

Latent Variable Modeling

Latent variables are indirectly observed variables that can be revealed through a decomposition. For example, we could construct a tensor from neuron activity over time and over multiple trials. Decomposing this would reveal constituent parts that have between-trial factors, temporal factors, and factors between neurons.

Compression

Finding an efficient method to decompose a tensor into smaller arrays can decrease data sizes by orders of magnitude, while minimizing loss. This technique is also used in decreasing computational complexity in deep neural networks.

ALS [1]

Let $\mathcal{X}$ be a third order tensor. We want to find the factor matrices A, B, C such that $\mathcal{X} = [\![A, B, C]\!]$.

ALS algorithim

We solve the Alternating Least Squares problem:

$$A_{t+1} \leftarrow \operatorname{argmin}_{A_{t+1}} \|\mathcal{X} - [\![A_{t+1}, B_t, C_t]\!]\|_F \tag{1}$$

$$B_{t+1} \leftarrow \operatorname{argmin}_{B_{t+1}} \|\mathcal{X} - [\![A_{t+1}, B_{t+1}, C_t]\!]\|_F \tag{2}$$

$$C_{t+1} \leftarrow \operatorname{argmin}_{C_{t+1}} \|\mathcal{X} - [\![A_{t+1}, B_{t+1}, C_{t+1}]\!]\|_F \tag{3}$$

Each problem can be solved either using the psuedo inverse or by solving the linear system by other means for both the regularized and unregularized case.

[1] For the rest of the presentation we will be assuming all tensors are order 3

Sketched ALS

Sketching ALS Subproblems

Each subproblem in ALS involves solving a linear system that is often over determined. We can sketch the columns of the linear system.

We can also add regularization to make decomposing ill-conditioned tensors easier and still converge to the same local minimum.

AdaGrad

Algorithm 1: input : N -order tensor $X \in \mathbb{R}^{I_1 \times \ldots \times I_N}$; rank F; sample size B, initialization $\left\{\boldsymbol{A}_{(n)}^{(0)}\right\}$

Result: factor matrices $A_{(n)}^{(i)}$, an array of factor matrices.

while *until stopping conditions satisfied* **do**

  sample n from $\{1, \ldots, N\}$; ▷ `Pick mode`

  sample $\mathcal{F}_n$ uniformly from $\{1, \ldots, J_n\}$; ▷ `pick fibers`

  form the stochastic gradient $G^{(r)}$

  determine the step size

end

How to Pick Sketching rates

Use many different sketching rates

Assign each of N sketching rates that you want to use a weight such that the $\sum_{i=1}^{N} w_{i,0} = 1$. Then each iteration treat these weights as a probability distribution and randomly pick a sketching rate

Updating the Weights

Let $\epsilon \in (0, 1]$ be the probability a iteration updates the weights of the sketching rates.

Update algorithm

$w_{i,t+1} = w_{i,t} \exp\left(-\frac{\eta \ell_t(s_i)}{\varepsilon}\right)$ for $i = 1, \ldots, N$. Where η is our aggressiveness and ℓ_t is the per unit time change in the loss function $\ell_t(s) = \frac{\|\mathcal{X} - [\![\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}]\!]\|_F - \|\mathcal{X} - [\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]\|_F}{\text{runtime}(t)\|\mathcal{X}\|_F}$ for the ith sketching rate.

Combining First and Second Order methods

Treat Adagrad as another arm

We can use the same update algorithm as the original CPDMWU paper and instead treat both sketched ALS and Adagrad as possible branches to choose.

Reasoning

1. ALS's per iteration cost is higher than adagrad
2. If at a certain time a gradient algorithm would preform better then sketched ALS on a per unit time basis then we should choose it.
3. We would expect initially gradient approaches would perform better then sketched ALS and over time the algorithm would weight sketched ALS and larger sketching rates higher.

Decomposition on Video

We convert a black and white video to a tensor and then decompose it. We try to find the low rank approximation with the parameters:

Rank	400
Size	$568 \times 568 \times 568$
Time Cap	1000 seconds
Sketching rates	0.1 , 0.325, 0.55 , 0.775, 1.0

Adagrad (with MWU)

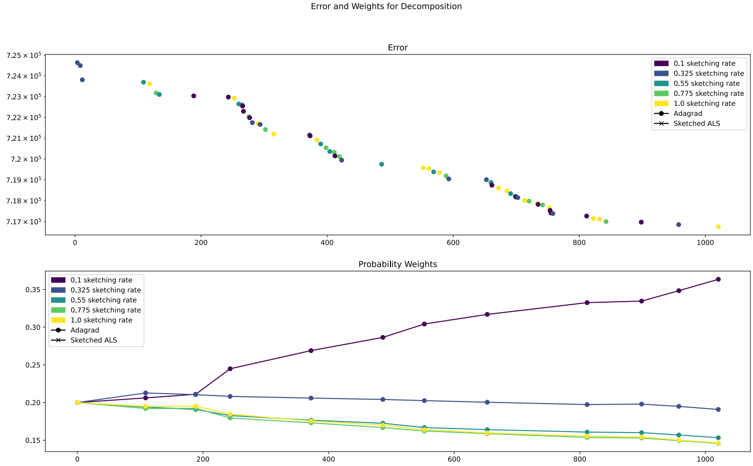


Figure: Decomposition of a video tensor using just adagrad

CPDMWU

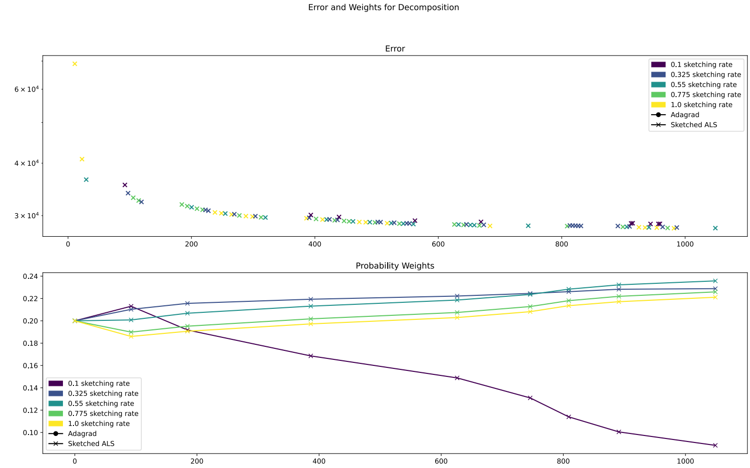


Figure: Decomposition of a video tensor using just sketched ALS

Both algorithms

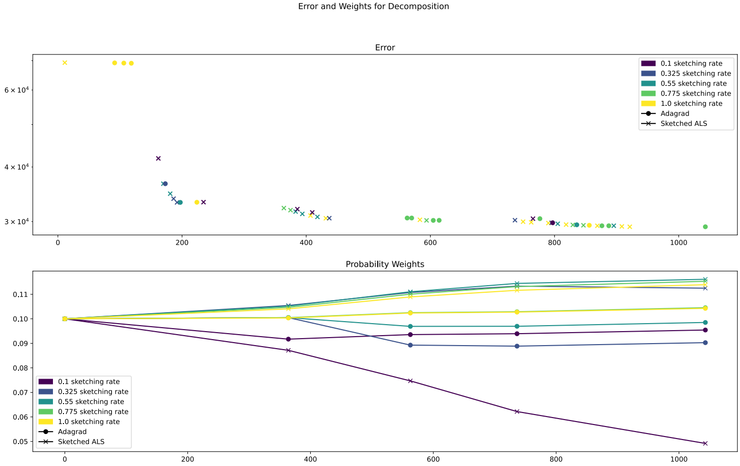


Figure: Decomposition of a video tensor using both adagrad and sketched ALS

Both algorithms

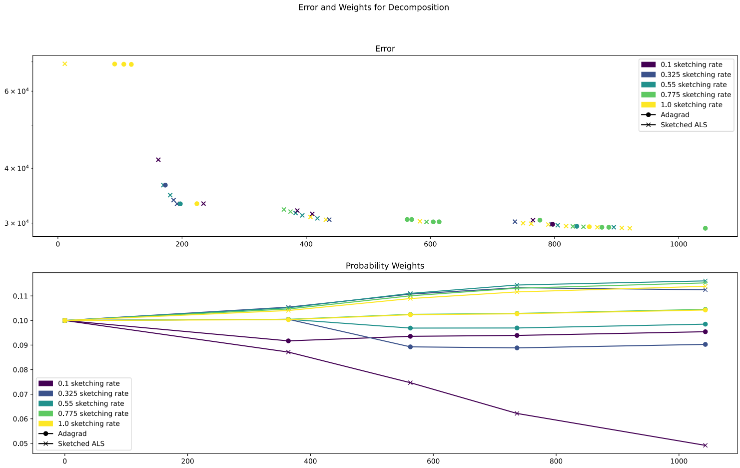


Figure: Decomposition of a video tensor using both adagrad and sketched ALS

Decomposition with known rank

We decompose tensors of varying sizes and ranks. In all cases CPDMWU with just sketched ALS performed better.

Adagrad

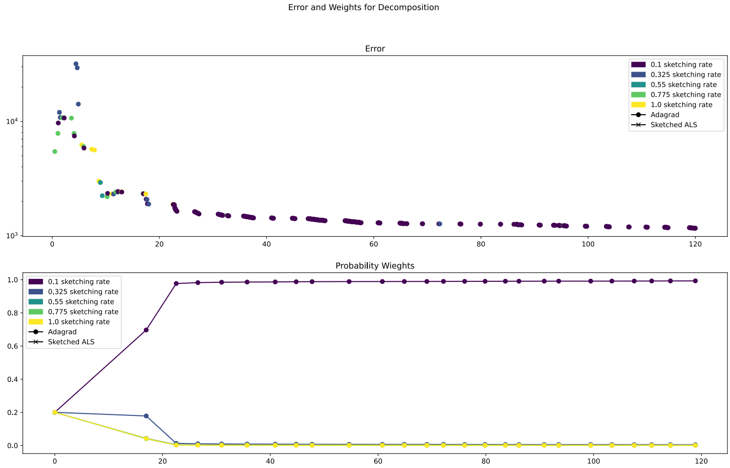


Figure: Adagrad with Rank = 50 and size = 200

Sketeched ALS

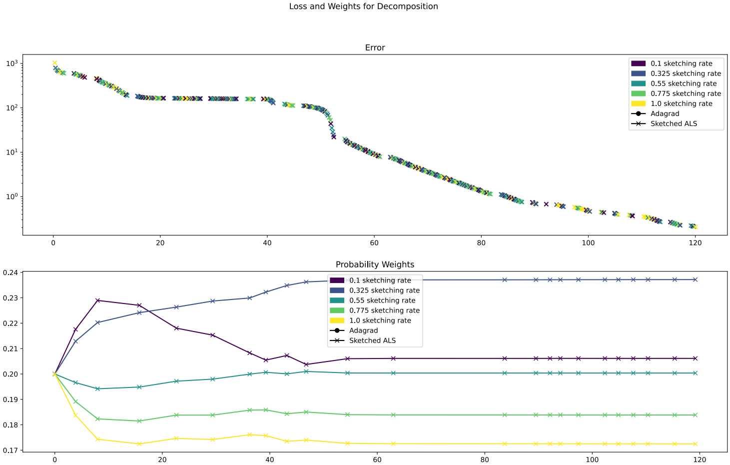


Figure: Sketeched ALS with rank = 50 and size = 200

Both

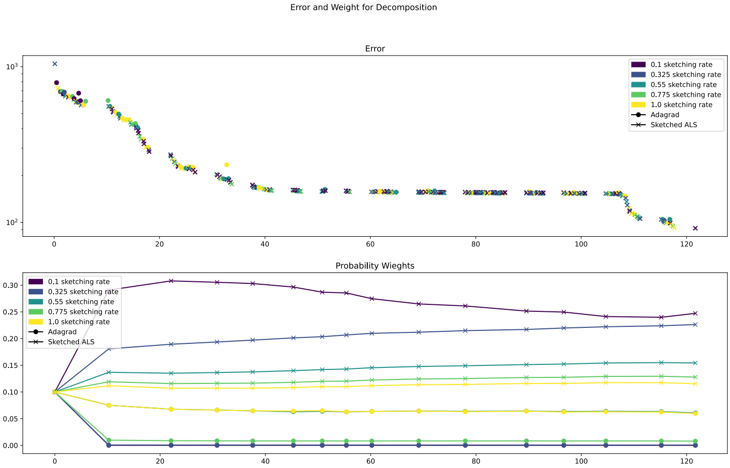


Figure: Sketeched ALS with rank = 50 and size = 200

Adagrad

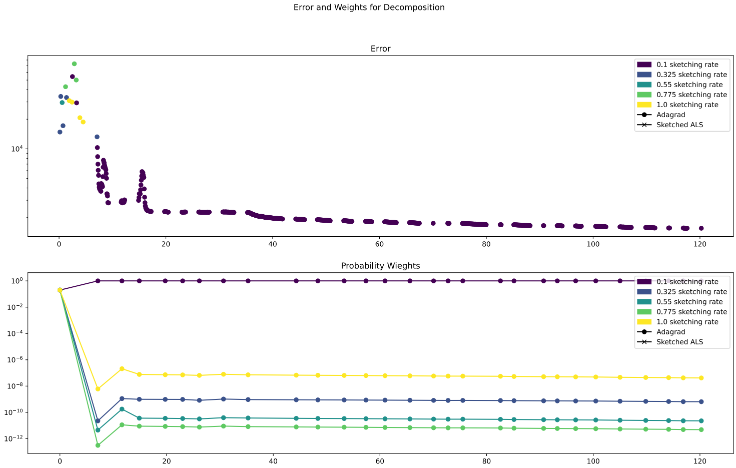


Figure: Adagrad with rank = 100 and size = 200

Sketched ALS

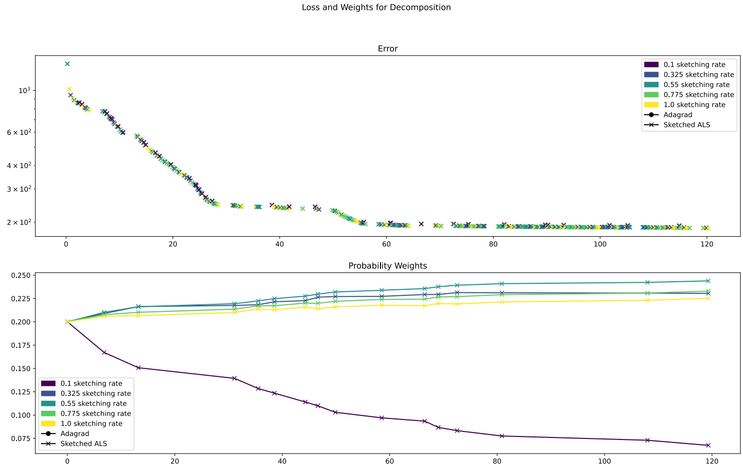


Figure: Sketched ALS with rank = 100 and size = 200

Both

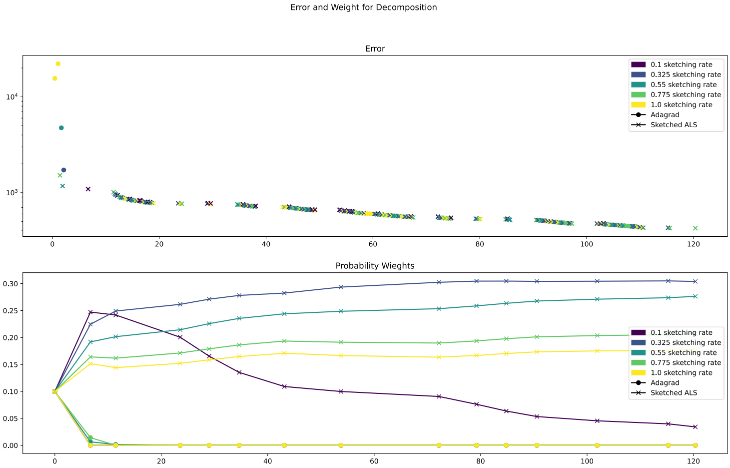


Figure: Adagrad and Sketched ALS with Rank = 100 and size = 200

Both

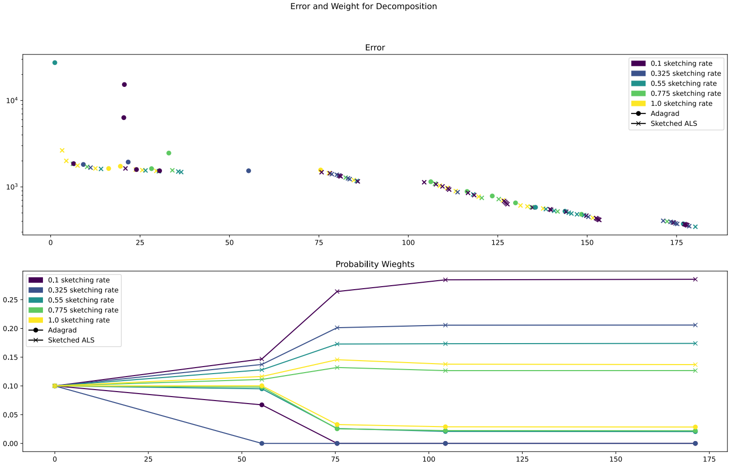


Figure: Adagrad and Sketched ALS with Rank = 300 and size = 50

Large Number of Trials - Loss

	average	median	standard deviation
Sketched ALS	1401.93	1501.96	308.18
Adagrad	26701.94	16282.73	24386.11
Both	2245.94	2295.10	262.60

Table: Statistics for the decompositions of 100 200x200x200 rank 100 tensors after 90 seconds

Conclusion

1. The initial benefits of using a faster gradient based algorithm quickly faded.
2. For the range of sizes and ranks of tensor tested, it is better to use sketched ALS.
3. Gradient algorithms would quickly not make any or very little improvements (less than 0.01% after a few iterations), while sketched ALS would.
4. The increase cost of updating the weights encouraged less frequent testing for optimal weights.
5. One iteration of Adagrad is significantly faster than ALS. So even if it made a small improvement, the algorithm was biased towards it.
6. The increased number of hyperparameters to tune also increase the complexity and could lead to numerical issues.

References

Kareem Aggour, Alex Gittens, Bülent Yenner, 2020
Adaptive Sketching for Fast and Convergent Canonical Polyadic Decomposition
arXiv.

Xiao Fu, Cheng Gao, Kejun Huang, Shahana Ibrahim, Hoi-To Wai, 2019
Block-Randomized Stochastic Proximal Gradient for Low-Rank Tensor Factorization
arXiv.