Learning-Based Low-Rank Approximations

Piotr Indyk  Ali Vakilian  Yang Yuan
Low Rank Approximation

**Singular Value Decomposition (SVD)**

Any matrix $A = U \Sigma V$, where:

- $U$ has orthonormal columns
- $\Sigma$ is diagonal
- $V$ has orthonormal rows

Best Rank-$k$ approximation: $A_k = U_k \Sigma_k V_k$

Equivalently: $A_k = \arg\min_{\text{rank}(B)=k} \|A-B\|_F$
Learning-Based Low Rank Approx.

**Low Rank Approx.**
Find a rank-\( k \) \( A' \), s.t.
\[
\| A - A' \|_F \leq (1+\varepsilon) \| A - A_k \|_F
\]
More efficient than computing \( A_k \)
- Sarlos’06
- Clarkson-Woodruff’09,13

*Refer to them as SCW algorithm*

**Main Approach:** linear sketches
- Perform SVD on \( RA \)
- \( R \) can be dense or sparse
- \( R \) is usually a random matrix

**Our Technique (Learned Sketch)**
Sample matrices \( A_1...A_N \)
Find \( S \) that minimizes total loss, i.e.,
\[
\sum_i \| A_i - \text{SCW}(S, A_i) \|_F
\]

**Details:**
- Use sparse matrices \( S \)
- Optimize using SGD in Pytorch
- Need to differentiate the above w.r.t. \( S \)
- Represent SVD as a sequence of power-method applications (each is differentiable)

**Sketch Monotonicity:** augmenting \( R \) + *learned sketch* \( S \) cannot increase the total loss

*Our algorithm with (\( R + S \)) inherits worst-case guarantees from \( R \)***
Sarlos-ClarksonWoodruff Framework

**Streaming algorithm (two passes)**
- Compute SA (first pass)
- Compute orthonormal V that spans rowspace of SA
- Compute \( AV^T \) (second pass)
- Return \( SCW(S,A) := [AV^T]_k V \)

**Space Complexity**
- Suppose that A is \( n \times d \), S is \( m \times n \)
- Then SA is \( m \times d \), \( AV^T \) is \( n \times m \)
- Space proportional to m
- Theory: \( m = O(k/\varepsilon) \)

**Classic Approach:**
- \( S \): sparse random matrix
- Worst-case theoretical bounds

**Our Learning-Based Approach**
- \( S \): learned matrix from training
- Better empirical performance
- No worst-case guarantee

**Augmenting Learned + Random**
- \( \checkmark \) Worst-case theoretical bounds
- \( \checkmark \) Better empirical performance
Empirical Evaluation

Datasets:
1. Videos: MIT Logo, Friends, Eagle
2. Hyperspectral images (HS-SOD)
3. TechTC-300

- **Training phase:** 200/400 matrices. **Testing phase:** 100 matrices
- Compare empirical recovery error $\sum_i ||A_i - SCW(S, A_i)||_F - ||A_i - [A_i]_k||_F$ of our learned matrix $S$ to random matrix $R$
Fallback Option: Learned + Random

Learned matrices have **better empirical performance**, but no
guarantees per matrix

<table>
<thead>
<tr>
<th>k</th>
<th>m</th>
<th>Sketch</th>
<th>Logo</th>
<th>Hyper</th>
<th>Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>Learned</td>
<td>0.1</td>
<td>0.52</td>
<td>2.95</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>Mixed</td>
<td>0.2</td>
<td>0.78</td>
<td>3.73</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>Random</td>
<td>2.09</td>
<td>2.92</td>
<td>7.99</td>
</tr>
</tbody>
</table>

**Solution**: combine S with random rows R

**Sketch monotonicity Lemma**: augmenting R with additional (learned) matrix S cannot increase the error of SCW

\[
\text{err}\left(\begin{bmatrix} R \\ S \end{bmatrix}, SCW \right) \leq \min \left\{ \text{err}\left(\begin{bmatrix} R \end{bmatrix}, SCW \right), \text{err}\left(\begin{bmatrix} S \end{bmatrix}, SCW \right) \right\}
\]