Greedy Column Subset Selection: New Bounds and Distributed Algorithms

Jason Altschuler

Joint work with Aditya Bhaskara, Thomas Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam

ICML 2016
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy algorithm
4. (Distributed) coreset greedy algorithm
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy algorithm
4. (Distributed) coreset greedy algorithm
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
Low-Rank Approximation

Given (large) matrix $A$ in $\mathbb{R}^{m \times n}$ and target rank $k << m,n$:

$$\arg \min_{X, \text{rank}(X) = k} \| A - X \|_F^2$$

- Optimal solution: $k$-rank SVD
- Applications:
  - Dimensionality reduction
  - Signal denoising
  - Compression
  - ...
Column Subset Selection (CSS)

- Columns often have important meaning
- **CSS:** Low-rank matrix approximation in column space of $A$

$$\arg \min_{S \subset [n], \ |S| = k} \left\| A - \Pi_{A[S]} A \right\|_F^2$$
Why use CSS for dimensionality reduction?

• **Unsupervised**
  • Don’t need labeled data

• **Classifier independent**
  • Can reuse output for different classifiers

• **Interpretable**
  • Generate features by subselecting instead of arbitrary function

• **Efficient during inference**
  • Feature subselection (CSS) better than matrix multiplication (SVD) if:
    • **Latency sensitive**
    • SVD projection matrix prohibitively large
    • Sparse
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy algorithm
4. (Distributed) coreset greedy algorithm
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
(Very simplified) background on CSS

• **CSS is UG-hard** [Civril 2014]

• **Importance sampling** [Drineas et al. 2004, Frieze et al. 2004, ...]
  • Fast, but additive-error bounds

• **More complicated algorithms** [Desphande et al. 2006, Drineas et al. 2006, Boutsidis et al. 2009, Boutsidis et al. 2011, Cohen et al. 2015, ...]
  • Multiplicative-error bounds, but complicated $\rightarrow$ not as fast/distributable
(Very simplified) background on CSS

- **CSS is UG-hard** [Civril 2014]

- **Importance sampling** [Drineas et al. 2004, Frieze et al. 2004, ...]
  - Fast, but additive-error bounds

- **More complicated algorithms** [Desphande et al. 2006, Drineas et al. 2006, Boutsidis et al. 2009, Boutsidis et al. 2011, Cohen et al. 2015, ...]
  - Multiplicative-error bounds, but complicated → not as fast/distributable

- **Greedy** [Farahat et al. 2011, Civril et al. 2011, Boutsidis et al. 2015]
  - Multiplicative-error bounds and fast/distributable
Contributions

• Prove **tight approximation guarantee** for the greedy algorithm

• First distributed implementation with provable approximation factors

• Further optimizations for the greedy algorithm

• **Empirical results** showing these algorithms are **extremely scalable** and have **accuracy comparable with the state-of-the-art**
Generalized Column Subset Selection (GCSS)

CSS (A, k)

\[ \arg\min_{S \subseteq [n], |S| = k} \| A - \Pi_{A[S]} A \|_F^2 \]

GCSS(A, B, k)

\[ \arg\min_{S \subseteq [n], |S| = k} \| A - \Pi_{B[S]} A \|_F^2 \]

- GCSS(A, B, k) uses k columns of B to approximate A
- Note: GCSS(A, A, k) = CSS(A, k)
Convenient reformulation of GCSS

\[
\arg\max_{S \subseteq [n], |S|=k} \| \Pi_{B[S]} A \|_F^2 = \arg\min_{S \subseteq [n], |S|=k} \| A - \Pi_{B[S]} A \|_F^2
\]

denote by \( f(S) \)

original GCSS cost function

- GCSS \( \iff \) maximizing \( f \) subject to cardinality constraint
- Intuition: \( f \) measures how much of \( A \) is “covered/explained” by selected columns
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy algorithm
4. (Distributed) coreset greedy algorithm
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
GREEDY algorithm to maximize $f$

$$S \leftarrow \emptyset$$

for $i = 1 : k$

Pick column $B_j$ that maximizes $f(S \cup \{B_j\})$

$$S \leftarrow S \cup \{B_j\}$$

Return $S$
Our result: Analysis of GREEDY

Consider GCSS($A, B, k$) with accuracy parameter $\varepsilon > 0$. Let $\text{OPT}_k$ be the optimal set of $k$ columns from $B$. If $r = O\left(\frac{k}{\varepsilon \sigma_{\min}(\text{OPT}_k)}\right)$ then:

$$f(\text{GREEDY}_r) \geq (1 - \varepsilon) f(\text{OPT}_k)$$

And this is tight up to a constant factor.
Our result: Analysis of GREEDY

Consider GCSS$(A, B, k)$ with accuracy parameter $\varepsilon > 0$. Let $\text{OPT}_k$ be the optimal set of $k$ columns from $B$. If $r = O\left(\frac{k}{\varepsilon \sigma_{\text{min}}(\text{OPT}_k)}\right)$ then:

$$f(\text{GREEDY}_r) \geq (1 - \varepsilon) f(\text{OPT}_k)$$

And this is tight up to a constant factor.

- We expect vectors in $\text{OPT}_k$ to be well-conditioned (think “almost orthogonal”)
  - If $\frac{1}{\sigma_{\text{min}}(\text{OPT}_k)}$ bounded by a constant, then only need $r = O\left(\frac{k}{\varepsilon}\right)$ columns
- Significant improvement upon current bounds: depend on worst singular value of any $k$ columns
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy + approximation guarantees
4. (Distributed) coreset greedy + approximation guarantees
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
DISTGREEDY: GCSS(A,B,k) with L machines

\[
S_1 = \text{GREEDY}(A, T_1, \frac{32k}{\sigma_{\min}(\text{OPT}_k)})
\]

\[
S_2 = \text{GREEDY}(A, T_2, \frac{32k}{\sigma_{\min}(\text{OPT}_k)})
\]

\[
S_L = \text{GREEDY}(A, T_L, \frac{32k}{\sigma_{\min}(\text{OPT}_k)})
\]

\[
S = \text{GREEDY}(A, \bigcup_{i=1}^{L} S_i, \frac{12k}{\sigma_{\min}(\text{OPT}_k)})
\]
DISTGREEDY: first observations

- Easy/natural to implement in **MapReduce**

- **2-pass streaming algorithm** in **random arrival** model for columns

- Can also do multiple rounds/epochs. Good for:
  - Massive datasets
  - Getting better approximations (next slide)
Our results: Analysis of DISTGREEDY

Consider an instance GCSS(A, B, k)

1 round result: DISTGREEDY with \( r = O\left(\frac{k}{\sigma_{\min}(OPT)}\right) \) gives objective value \( \Omega\left(\frac{f(OPT_k)}{\kappa(OPT_k)}\right) \)

Condition number \( \frac{\sigma_{\max}(OPT_k)}{\sigma_{\min}(OPT_k)} \)

Multi-round result: \( O\left(\frac{\kappa(OPT)}{\varepsilon}\right) \) rounds gives objective value \( \Omega\left( (1 - \varepsilon)f(OPT_k) \right) \)
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy + approximation guarantees
4. (Distributed) coreset greedy + approximation guarantees
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
Scalable Implementation: GREEDY++

4 optimizations that preserve our $1 - \varepsilon$ approximation for $\text{GREEDY}(A \in \mathbb{R}^{m \times n_A}, B \in \mathbb{R}^{m \times n_B}, k)$

1. **JL Lemma** [Johnson & Lindenstrauss 1982, Sarlos 2006]: randomly project to $m' \approx \frac{k \log (\max(n_A, n_B))}{\varepsilon^2}$ rows while still preserving k-linear combos $\frac{\| \sum_{i=1}^{k} c_i v_i \|^2}{\| \sum_{i=1}^{k} c_i T(v_i) \|^2} \in [1 \pm \varepsilon]$.

2. **Projection-Cost Preserving Sketches** [Cohen et al. 2015]: sketch $A$ with $n'_A \approx \frac{k}{\varepsilon^2}$ columns.

3. “**Stochastic Greedy**” [Mirzasoleiman et al. 2015]: each iteration only uses $\frac{n_B}{k} \log \frac{1}{\varepsilon}$ marginal utility calls instead of $n_B$.

4. **Updating $A$ every iteration** [Farahat et al. 2013]: after each iteration, remove projections of $A$ and $B$ onto selected column. Reduces complexity of marginal utility from $k m n_A \rightarrow mn_A + mn_B$. 
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy + approximation guarantees
4. (Distributed) coreset greedy + approximation guarantees
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketches
“Small” dataset (mnist): to show **accuracy**

- **m = 60K instances**
- **n = 784 features**
- **10 classes**

- **Takeaway:** GREEDY, GREEDY++, and GREEDY-core have roughly same accuracy as state-of-the-art
Large dataset (news20.binary) to show scalability

• Takeaway: DISTGREEDY able to scale to massive datasets while still selecting effective features
Talk Outline

1. Background/motivation for Column Subset Selection (CSS)
2. Previous work + our contributions
3. (Single-machine) greedy + approximation guarantees
4. (Distributed) coreset greedy + approximation guarantees
5. Further optimizations
6. Experiments
7. [Time permitting] Proof sketch: analysis of GREEDY
Proof sketch: Analysis of GREEDY

- **Key lemma**: Exists element of $\text{OPT}_k$ that gives large marginal gain to $\text{GREEDY}_r$
  - Closes gap to $f(\text{OPT}_k)$
- Similar to submodular functions
Proof sketch: Analysis of GREEDY

\[ \sigma_{\min}(\text{OPT}_k) \frac{(f(\text{OPT}_k) - f(\text{GREEDY}_r))^2}{4kf(\text{OPT}_k)} \]

- **Key lemma:** Exists element of \(\text{OPT}_k\) that gives large marginal gain to \(\text{GREEDY}_r\)
  - Closes gap to \(f(\text{OPT}_k)\)
- **Similar to submodular functions**
Proof sketch: Analysis of GREEDY

\[ \sigma_{\text{min}}(\text{OPT}_k) \frac{(f(\text{OPT}_k) - f(\text{GREEDY}_r))^2}{4k f(\text{OPT}_k)} \]

- **Key lemma**: Exists element of OPT\(_k\) that gives large marginal gain to GREEDY\(_r\)
  - Closes gap to \(f(\text{OPT}_k)\)
- Similar to submodular functions
Proof sketch: Analysis of GREEDY

- **Key lemma**: Exists element of OPT\(_k\) that gives large marginal gain to GREEDY\(_r\)
  - Closes gap to \(f(\text{OPT}_k)\)
- Similar to submodular functions
Proof sketch: Analysis of GREEDY

- **Key lemma:** Exists element of $\text{OPT}_k$ that gives large marginal gain to $\text{GREEDY}_r$
  - Closes gap to $f(\text{OPT}_k)$
- Similar to submodular functions
Proof sketch: Analysis of GREEDY

- **Key lemma**: Exists element of $\text{OPT}_k$ that gives large marginal gain to $\text{GREEDY}_r$
  - Closes gap to $f(\text{OPT}_k)$
- Similar to submodular functions
Proof sketch: Analysis of GREEDY

- **Key lemma**: Exists element of $\text{OPT}_k$ that gives large marginal gain to $\text{GREEDY}_r$
  - Closes gap to $f(\text{OPT}_k)$
- Similar to submodular functions
Questions?