Divide-and-Conquer Anchoring
for Near-Separable Nonnegative Matrix Factorization and Completion in High Dimensions

Tianyi Zhou,
Joint work with
Wei Bain and Prof. Dacheng Tao,
Centre for Quantum Computation & Intelligent Systems (QCIS)
University of Technology, Sydney (UTS)

ICDM 2013, Dallas
Big Data could be a Nightmare

- **Huge-scale Dataset**
  - Num. of Users in Social Network
  - Num. of Web texts, Images, Movies

- **High-dimensional Features**
  - Num. of Attributes, Items
  - Num. of Words, Pixels, Frames

- **Missing Values and Noise**
  - Dirty Data with Low Quality

- **Complex Structure**
  - Inconsistent to one Structure
Learning & Mining Bottleneck

- **Speed**, Speed, Speed...
- **Statistical Laws** lose power in High Dimensions
- **User is not satisfied and convinced**: Single artificial solution with weak interpretation
- **Sensitive** to Missing value, Noise, and Complex Structure
Example: Learn from Faces

• **Learner/Miner:** PCA (or Sparse PCA)?

  - Oh, it’s complicated...
  - Hellow! Who r u?

• **User:** The results of long-time optimization/decomposition, weak interpretation, blablabla
Example: Learn from Faces

- **Learner/Miner:** No? Then Manifold Learning (Linear approximation)? 😐

- **User:** Same problem!

![Image of facial expressions]
Idea: Why not use (partial) data itself to express the learning result?

- Select (find), not iteratively optimize
  - Big data already contains affluent resources to build learning result
  - User is happier to be convinced by real instances
  - Faster
Idea: Why not use (partial) data itself to express the learning result?

• Summarize, not Artificially Create
  – Human being learns from summarizing the world by representative samples for thousands of years.

• Distribute learning in low dim, not in high dim
  – Use the data redundancy to conquer
Divide-and-Conquer Anchoring (DCA)

• **Huge scale dataset**: Summarize it with representative instances *(anchors)*

• **High dimension**: learn in low-dim in parallel, and recover the high-dim solution low-dim solutions *(Divide-and-Conquer)*

• **Missing Values or Noise**: Use the data redundancy and compressible structure

• **Complex Structure**: Use REAL instances to express it *(Users like case study!)*
Outline

• Divide-and-Conquer Anchoring
  – Geometry: Anchors in High and Low Dimensions
  – Scheme: Divide-and-Conquer Anchoring (DCA)
  – Tricks: Ultra-Fast Anchoring in 1 or 2 Dimensions
  – Harder: Incomplete Data
• Comparison: Near Separable NMF Algorithms
• Experiments
  – Synthetic Data
  – Real Data
• Take Home Messages
Anchors (Representative Instances)

- Anchors are important faces, representative users, key frames, etc...
- So How to define “important”, “representative”, “key” in math?
- **One hint**: they can represent all the others!

\[ X = FX_A, \Pi F = \begin{bmatrix} I_k \\ F' \end{bmatrix} \]

- All data, one sample a row
- Anchors (row index A)
Anchors (Geometry)

- So how to define anchors in geometry (equally)? What do they look like in space?
- (Separable Assumption) Anchor points are vertices/extreme rays whose convex/conical hull contains all the other data points.

\[ \Delta(V) = \left\{ \sum_{i=1}^{k} \alpha_i v_i \mid v_i \in V, \alpha_i \in \mathbb{R}_+, \sum_{i=1}^{k} \alpha_i = 1 \right\} \]

\[ \text{cone}(R) = \left\{ \sum_{i=1}^{k} \alpha_i r_i \mid r_i \in R, \alpha_i \in \mathbb{R}_+ \right\} \]

\[ \text{cone}(X_A) \]

\[ X = \{ \bullet \} \]

\[ X_A = \{ \circ \} \]
Mirror Mirror is the anchor set Unique?

• Mirror tells you in simplex case, the above Separable Assumption guarantee the uniqueness of A.


• How about the cone case?

**Definition 2:** (Separability assumption) All the data points in $X$ reside in a conical hull of $R$, which is a subset $A \subseteq [n]$ of data points in $X$. Geometrically, the separability assumption is

$$\forall i \in [n], x_i \in \text{cone}(X_A), X_A = \{x_i\}_{i \in A}.$$  \hspace{1cm} (4)

Algebraically, separability assumption means

$$X = FX_A, \Pi F = \begin{bmatrix} I_k \\ F' \end{bmatrix}$$  \hspace{1cm} (5)

where $I_k$ is a $k$-by-$k$ identity matrix, $F' \in \mathbb{R}^{(n-k) \times k}$, and $\Pi$ is a row permutation matrix.

**Uniqueness:** a finitely generated and pointed (NMF is a special case) cone $\text{cone}(R)$ possesses a finite and unique set of extreme rays $R$, and $\text{cone}(R)$ is the conical hull generated by these extreme rays $R$.

-Yurii Nesterov
Idea: **Geometry** of Anchors in High dimensions partially preserved in Low dimensions (Simplex)

A is the set of anchors (vertex in this case), A bar is the set of anchors in low dimensions after projection, A bar belongs to A.
Idea: **Geometry** of Anchors in High dimensions partially preserved in Low dimensions (Cone)

A is the set of anchors (rays in this case), A bar is the set of anchors in low dimensions after projection, A bar belongs to A.
Conjecture: (Randomly) Project dataset for $k$ times, find anchors in low dim (sub-problem) by some algorithm. Can we find all the original anchors at last (with high probability)?
 Guarantee: Times of projections (num. of sub-problems) we need is $O(k \log k)!$ (k: num. of anchors)

\[ p^*_i = \Pr[i \in \bar{A}] \]

Probability of data point i identified as anchors in low-dim projections

**Theorem 2: (Identifiability)** Suppose $\min_{i \in A} p^*_i \geq \alpha/k > 0$, $i = 1, 2, ..., k$. Given the statistics from the solutions of s sub-problems finding low-dimensional anchors,

\[ g(i) = \sum_{j=1}^{s} I(i \in \bar{A}^j) \] (16)

by Theorem 1, clearly we have

\[ g(i : i \in A) \geq 0, \sum_{i \in A} g(i) = \sum_{i=1}^{s} |\bar{A}^i| \quad \text{and} \quad g(i : i \notin A) = 0; \] (17)

further, it holds with probability at least $1 - k \exp \left(-\frac{\alpha}{3k}\right)$ that

\[ \min_{i \in A} g(i) > 0. \] (18)

*alpha here is related to projection dimension d, random projection ensemble, and data.*
Scheme: Divide-and-Conquer Anchoring

• **Divide step**: Randomly project dataset for s times, find and record the low-dim anchors by any anchoring algorithm (which is usually faster and more accurate in low-dim).

• **Conquer step (robust to noise)**: Find the top k data points selected as anchors for most times. *in noisy case, non-anchors has small but nonzero probability to be selected

• **Hmm.. But solving sub-problems might be slow!**
Tricks: Anchoring in 1 or 2 dimensions

• In simplex case: anchors in 1-dim is the smallest and largest points!
Tricks: Anchoring in 1 or 2 dimensions

• In cone case: anchors in 2-dim is the two points with smallest and largest angles to x or y axis!
Harder: Incomplete datasets

- Cannot be handled by previous methods.
- In DCA:
  
  - In each sub-problem, randomly select some dimensions, find the anchors of data samples available on the selected dimensions.
Comparison: Near Separable NMF

• Solve similar problem: find anchors.
• Tells you separable NMF can be solved in polynomial time if the simplex is alpha-robust (less flat)


• Their algorithm uses the fact: vertex cannot be expressed as convex combination of other data points – search infeasible LP on n points
• Requires n LP, each on n-1 variables, slow 😞
Faster Algorithms are not fast:

- **Hottopixx:** $X = CX$, $C$ belongs to elegantly designed polyhedral, single LP with $n^2$ variables, slow 😞


- **Greedy Pursuit: SPA** is a modified Gram-Schmidt orthogonalization. **XRAY** selects a new anchor according to the residual of a randomly selected exterior data point. **Not distributable.** 😞


- **All algorithms need to compute distances (slow in high dimensions $p$), and never touch incomplete case.**
Experiments: Synthetic data tells us more properties about DCA (1)

**Recovery Rate**

- **Cone**
  - SPA
  - XRAY
  - DCA(s=75)
  - DCA(s=441)
  - DCA(s=624)

- **Simplex**
  - SPA
  - XRAY
  - DCA(s=30)
  - DCA(s=77)
  - DCA(s=100)

**-Recovery Error**

- **Cone**
  - SPA
  - XRAY
  - DCA(s=75)
  - DCA(s=441)
  - DCA(s=624)

- **Simplex**
  - SPA
  - XRAY
  - DCA(s=30)
  - DCA(s=77)
  - DCA(s=100)

**Time Cost (exp)**

- **Cone**
  - SPA
  - XRAY
  - DCA(s=75)
  - DCA(s=258)
  - DCA(s=441)
  - DCA(s=624)

- **Simplex**
  - SPA
  - XRAY
  - DCA(s=30)
  - DCA(s=53)
  - DCA(s=77)
  - DCA(s=100)
Experiments: Synthetic data tells us more about DCA (Incomplete, Cone) (2)

- k = 10, 50x100 matrix, 125 sub-problems out of 9900: $k \log(k)$.
- Tolerance of Noise Level: DCA can deal with large noise.
- Smaller sampling ratio than Matrix Completion: 10% entries will give accurate recovery of the whole matrix.
Experiments: Synthetic data tells us more about DCA (Incomplete, Cone) (2)

- **DCA Paradox:** Why recovery rate increases when increasing the matrix rank? This is in contrast to matrix completion!

- **Reason:** Difference between Selection and Optimization’s Complexity. To select a fewer anchors from a great number of data points is harder.
Experiments: Represent other data in real dataset by selected anchors

Table II

<table>
<thead>
<tr>
<th></th>
<th>SPA</th>
<th>XRAY</th>
<th>DCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIT scene</td>
<td>0.3351/0.085</td>
<td>0.3133/104.19</td>
<td>0.2766/0.038</td>
</tr>
<tr>
<td>MNIST</td>
<td>569.21/1.891</td>
<td>0.6128/233.69</td>
<td>0.5792/0.928</td>
</tr>
<tr>
<td>Grolier</td>
<td>0.5041/3.777</td>
<td>0.3589/17.89</td>
<td>0.2907/1.303</td>
</tr>
</tbody>
</table>

Reconstruction error and CPU seconds of SPA, XRAY and DCA on three datasets. The rank $k$ for reconstructing them is 30, 50, 50. Result format: $\ell_2$ error/CPU seconds.
Experiments: Find representative users in recommendation system and do collaborative filtering

Table I

<table>
<thead>
<tr>
<th></th>
<th>GreB</th>
<th>DCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>100k</td>
<td>0.15/0.94/0.57</td>
<td>0.11/0.87/0.12</td>
</tr>
<tr>
<td>1M</td>
<td>0.12/0.86/3.54</td>
<td>0.09/0.80/1.36</td>
</tr>
<tr>
<td>10M</td>
<td>0.11/1.04/20.67</td>
<td>0.08/0.95/5.98</td>
</tr>
</tbody>
</table>
Take Home Messages:

• High Level: Big Data Learning
  – Learn by Selecting: Find the anchors
  – Learn in Low-Dim: Divide-and-Conquer

• Middle Level: what DCA can solve
  – Matrix Factorization/Completion (could be negative)
  – Learning Graphical Models (LDA, Tree, BayesNet ...)
  – Feature Selection (Anchor features)

• DCA:
  – Fast: distributed \((k \log k \text{ threads})\), no iteration, only max/ min operations and entry-wise multiplication.
  – Easy: a few lines in MATLAB.
  – Nonlinear? Yes, Change Metric!
Thanks!

Q & A

Tianyi Zhou
http://tianyizhou.wordpress.com
tianyi.david.zhou@gmail.com