PROXIMUS:  
A Framework for Analysis of Very High Dimensional Discrete Attributed Datasets

Mehmet Koyutürk and Ananth Grama

Department of Computer Sciences

http://www.cs.purdue.edu/homes/koyuturk/proximus/
The Problem

Input: \( n \)

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & \ldots \\
0 & 0 & 1 & 1 & \ldots \\
1 & 1 & 0 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]

Output: \( n \)

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & \ldots \\
0 & 0 & 1 & 1 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]

\( k << m \), not predetermined

Each input vector is within bounded distance from some output vector.
An Example

Input Matrix: 

\[
A = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

Pattern Matrix: 

\[
V = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\]

Presence Matrix: 

\[
U = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[A \approx U^TV\]

Maximum Hamming Distance : 1
Motivation

Very high dimensional large discrete datasets in many applications

- **Data Mining**
  - Discrete feature space
  - Many sets over a large universal set (Association Rule Mining)

- **Bioinformatics**
  - Sequences over a finite alphabet, gene regulation, pattern discovery

- Information retrieval, scientific computing…
Singular Value Decomposition

\[ A_{m \times n} = U_{r \times m}^T \Sigma_{r \times r} V_{r \times n} \]

\( r \) : rank of \( A \)
\( \Sigma \) : singular values of \( A \), diagonal
\( U, V \) : singular vectors of \( A \), orthogonal
Each triple is a dominant pattern, in order

Truncated SVD: \( A_{m \times n} \approx U_{k \times m}^T \Sigma_{k \times k} V_{k \times n} \)

\( k < r \), error : \( k+1 \) th singular value of \( A \)
What is wrong with SVD?

- Orthogonality: negative values!
- Patterns represent overall matrix rather than clusters
- "Extracted" patterns: hard to interpret
- Computation is very expensive
SVD Variants

- **Semi-Discrete Decomposition (SDD)**
  - Restrict singular vectors to \{-1,0,1\}
  - Faster but orthogonalization is still a problem

- **Principle Direction Divisive Partitioning (PDDP)**
  - Partition based on first singular vector
  - Designed for real valued data

- **Centroid Decomposition**
  - Cluster patterns rather than overall
  - Hard to discretize
PROXIMUS: Basic Idea

- Approximation restricted to binary values
  - Easy interpretation
  - Faster computation

- Recursive decomposition
  - Partition based on first singular vector
  - Flexible stopping criteria: stop when approximation is adequate
Rank-one Approximation

How to compute first “singular” vector?

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
\end{bmatrix}
\approx
\begin{bmatrix}
0 \\
1 \\
1 \\
1 \\
0 \\
1 \\
1 \\
\end{bmatrix}
\times
\begin{bmatrix}
0 & 1 & 1 & 0 & 1
\end{bmatrix}
\]

Presence Vector    Pattern Vector
Formulation

Given $A \in \{0,1\}^{m \times n}$

find $x \in \{0,1\}^{m \times 1}$, $y \in \{0,1\}^{n \times 1}$

to minimize error:

$$||A-xy^T||_F^2$$

Similar to maximum clique problem:

Find a “dense” subgraph of a bipartite graph

NP-hard
Heuristic may work as desired!

Optimal Approximation:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
\end{bmatrix}
\times
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Error : 4

“Desirable” Approximation:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
\end{bmatrix}^T \times
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 \\
\end{bmatrix}
\]

Error : 6
Alternating Iterative Heuristic

Fix $y$, compute $x$ to maximize $2x^T Ay - \|x\|_2^2 \|y\|_2^2$.
Then fix $x$, compute $y$. Repeat until convergence.

Rows

Columns

Fixed pattern vector  Computed presence vector  Final approximation
Recursive Decomposition

\[ U = [x_2, x_4, x_5] \]
\[ V = [y_2, y_4, y_5] \]
\[ A \approx U^T V \]

\[ A \approx x_1 y_1^T \]
\[ x_1(i) = 1 \]
\[ A \approx x_2 y_2^T \]
\[ x_2(i) = 1 \ \forall i \]
\[ y_2 \text{ adequately represents matrix } x_2, y_2 \]

\[ A \approx x_3 y_3^T \]
\[ x_3(i) = 1 \]
\[ A \approx x_4 y_4^T \]
\[ x_4(i) = 1 \ \forall i \]
\[ y_4 \text{ adequately represents matrix } x_4, y_4 \]

\[ A \approx x_5 y_5^T \]
\[ x_5(i) = 1 \ \forall i \]
\[ y_5 \text{ adequately represents matrix } x_5, y_5 \]
Stopping Criteria

- Stop whenever both hold:
  - All rows are present in the approximation
  - Pattern represents these rows adequately: Hamming radius less than $\varepsilon$
    - Hamming radius: Maximum Hamming distance to representative pattern
    - $\varepsilon$: Predetermined bound, determines quality of approximation

- If not, partition based on rank-one approximation
  - 1’s in presence vector go to one part, 0’s to other
Initialization of Pattern Vector

- Crucial for convergence to desired local optima
- Must be fast (at most linear time)
- General idea: Find a rough cluster of rows, initialize pattern vector to their centroid
  - Partition along one dimension
  - Greedy graph growing
  - Random row’s neighborhood
Matrix with $m$ rows, $n$ columns, $p$ non-zeros

Rank-one approximation
- Initialization: $O(p)$
- Each iteration: mat-vec, $O(p)$
- Rapid convergence

Each level of recursion tree has $p$ non-zeros

Overall complexity: $O(h \times p)$
- $h \leq k$: height of recursion tree
Visual Results
### Association Rule Mining

#### Transaction Set

<table>
<thead>
<tr>
<th></th>
<th>beer</th>
<th>butter</th>
<th>milk</th>
<th>snacks</th>
<th>eggs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$T_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_4$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$T_1 = \{ \text{beer, snacks} \}$

$T_2 = \{ \text{milk, butter} \}$

$T_3 = \{ \text{milk, butter, eggs} \}$

$T_4 = \{ \text{beer, snacks} \}$

$T_5 = \{ \text{milk, eggs} \}$

$T_6 = \{ \text{milk, butter, eggs} \}$
Compressing the transaction set

- Compute decomposition $A \approx U^T V$
  - $V$ is the compressed (virtual) transaction set
  - $U$ assigns weight to virtual transactions

<table>
<thead>
<tr>
<th>Compressed Transaction Set</th>
<th>$w(T'_1)$</th>
<th>$w(T'_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T'_1 = {\text{beer, snacks}}$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$T'_2 = {\text{milk, butter, eggs}}$</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Mine the compressed transaction set!
## Datasets

**IBM Quest association data generator**

<table>
<thead>
<tr>
<th></th>
<th># of transactions</th>
<th># of patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>10K</td>
<td>20</td>
</tr>
<tr>
<td>Medium</td>
<td>100K</td>
<td>100</td>
</tr>
<tr>
<td>High</td>
<td>1M</td>
<td>500</td>
</tr>
</tbody>
</table>

**transactions**

<table>
<thead>
<tr>
<th>patterns</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>L100K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td>M100K</td>
<td>H100K</td>
</tr>
<tr>
<td></td>
<td>M1M</td>
<td>M100K</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>H100K</td>
</tr>
</tbody>
</table>
Performance on ARM

- Almost always over 95% precision & recall for all datasets
- Speedup in the order of tens

Performance on M100K dataset

2479 approximation vectors, preprocessing time: 5.89 seconds

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>13.76</td>
<td>1.06</td>
<td>332395</td>
<td>333046</td>
<td>331087</td>
<td>99.6</td>
<td>99.8</td>
</tr>
<tr>
<td>1.0</td>
<td>8.01</td>
<td>0.39</td>
<td>138745</td>
<td>139519</td>
<td>137200</td>
<td>98.3</td>
<td>99.0</td>
</tr>
<tr>
<td>1.5</td>
<td>5.12</td>
<td>0.18</td>
<td>60500</td>
<td>59580</td>
<td>59580</td>
<td>100.0</td>
<td>98.5</td>
</tr>
<tr>
<td>2.0</td>
<td>3.14</td>
<td>0.11</td>
<td>36061</td>
<td>39117</td>
<td>35366</td>
<td>90.4</td>
<td>98.1</td>
</tr>
<tr>
<td>2.5</td>
<td>0.86</td>
<td>0.03</td>
<td>14750</td>
<td>14645</td>
<td>14645</td>
<td>100.0</td>
<td>99.3</td>
</tr>
<tr>
<td>3.0</td>
<td>0.68</td>
<td>0.02</td>
<td>11180</td>
<td>11070</td>
<td>11070</td>
<td>100.0</td>
<td>99.0</td>
</tr>
</tbody>
</table>
Effect of Parameters

- Robust to increasing number of transactions
- More patterns, harder problem
  - More approximation vectors to preserve quality
  - More preprocessing time
  - Less speedup
  - Can still maintain quality!
- Bound on error
  - Tighter bound, higher quality but less speedup
  - Loose bound does not effect after some point
Performance on Real Data

- *Agaricus Lepiota* dataset
  - “mushrooms with bell-shaped and fibrous caps have brown gills”
  - 8124 species
  - 23 categorical attributes → 118 binary attributes
- Decomposition into 1142 approximation vectors in 15.73 seconds

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>46.49</td>
<td>7.49</td>
<td>497813</td>
<td>464546</td>
<td>464185</td>
<td>99.9</td>
<td>93.2</td>
</tr>
<tr>
<td>14.8</td>
<td>14.98</td>
<td>2.19</td>
<td>63413</td>
<td>63299</td>
<td>63210</td>
<td>99.9</td>
<td>99.7</td>
</tr>
<tr>
<td>19.6</td>
<td>7.43</td>
<td>1.30</td>
<td>32593</td>
<td>32589</td>
<td>32589</td>
<td>100.0</td>
<td>99.9</td>
</tr>
</tbody>
</table>
Scalability

Number of rows
Number of patterns
Transaction length
Conclusions & Future Work

- Software available
  
  http://www.cs.purdue.edu/homes/koyuturk/proximus/

- Application to other areas
  - Clustering, classification, information retrieval, gene regulation…

- Comparison to probabilistic subsampling

- Generalization to similar problems
  - Integer datasets
  - Real matrix = real matrix × binary matrix