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

**Mehmet Koyutürk and Ananth Grama**

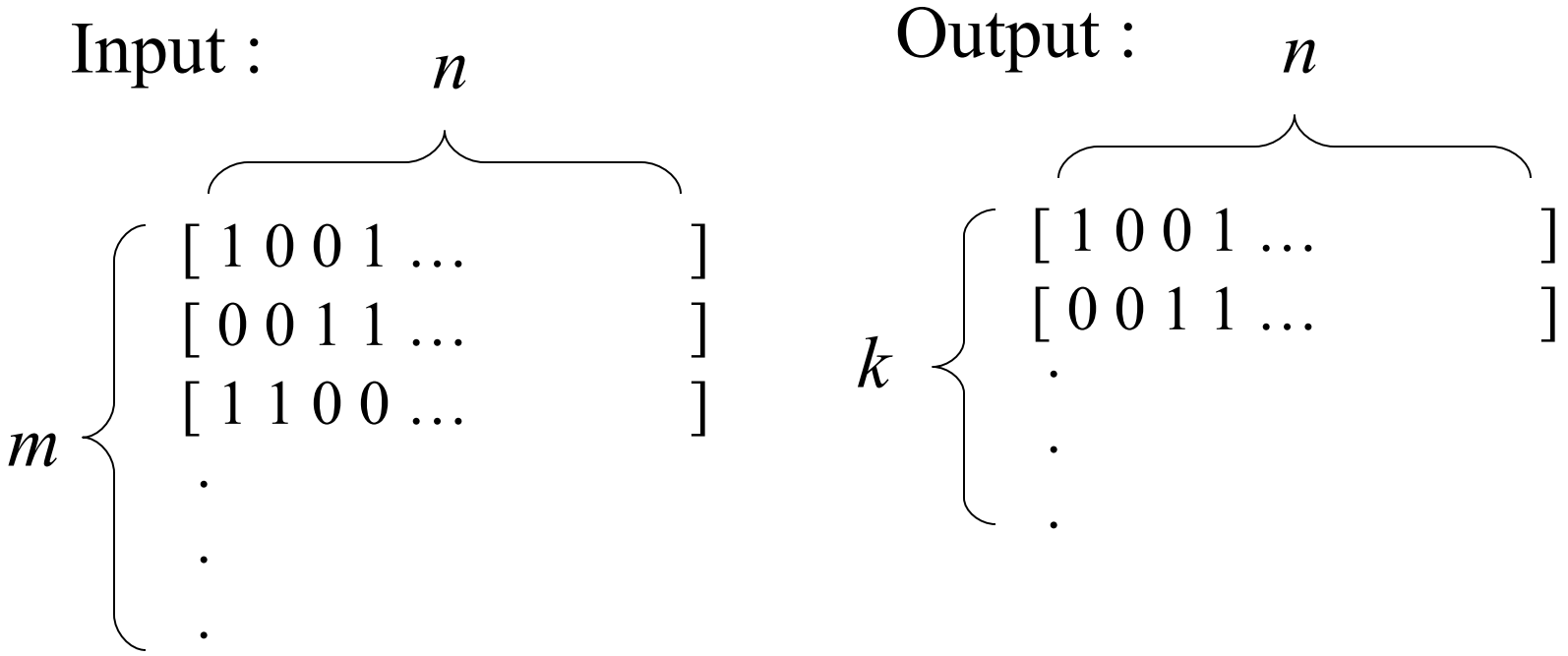
Department of Computer Sciences

PURDUE UNIVERSITY

CRI COMPUTING RESEARCH INSTITUTE

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

# The Problem




$k << m$, 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 :

$$U = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$A \approx U^{\mathrm{T}} V$$

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

$$A_{m\times n} = U_{r\times m}{}^{\mathrm{T}} \Sigma_{r\times r} V_{r\times n}$$

$r$ : 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: $A_{m\times n} \approx U_{k\times m}{}^{\mathrm{T}} \Sigma_{k\times k} V_{k\times n}$

$k <$ r, error : $k$+1$^{\mathrm{th}}$ singular value of $A$

# What is wrong with SVD?

$$A \approx \begin{bmatrix} 0 & 0.7 & 0 & 0 \\ 0.4 & 0 & 0.7 & 0.6 \\ 0.6 & 0 & 0 & -0.4 \\ 0 & 0.7 & 0 & 0 \\ 0.4 & 0 & -0.7 & 0.6 \\ 0.6 & 0 & 0 & -0.4 \end{bmatrix} \times \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \times \begin{bmatrix} 0 & 0.5 & 0.6 & 0 & 0.5 \\ 0.7 & 0 & 0 & 0.7 & 0 \\ 0 & 0.7 & 0 & 0 & -0.7 \\ 0 & -0.5 & 0.7 & 0 & -0.5 \end{bmatrix}$$

- 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?

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

**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!

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |

Optimal Approximation:

$$[\,1\;\;1\;\;1\;\;1\,]^T \times [\,1\;\;1\;\;1\;\;1\,]$$

Error : 4

“Desirable” Approximation:

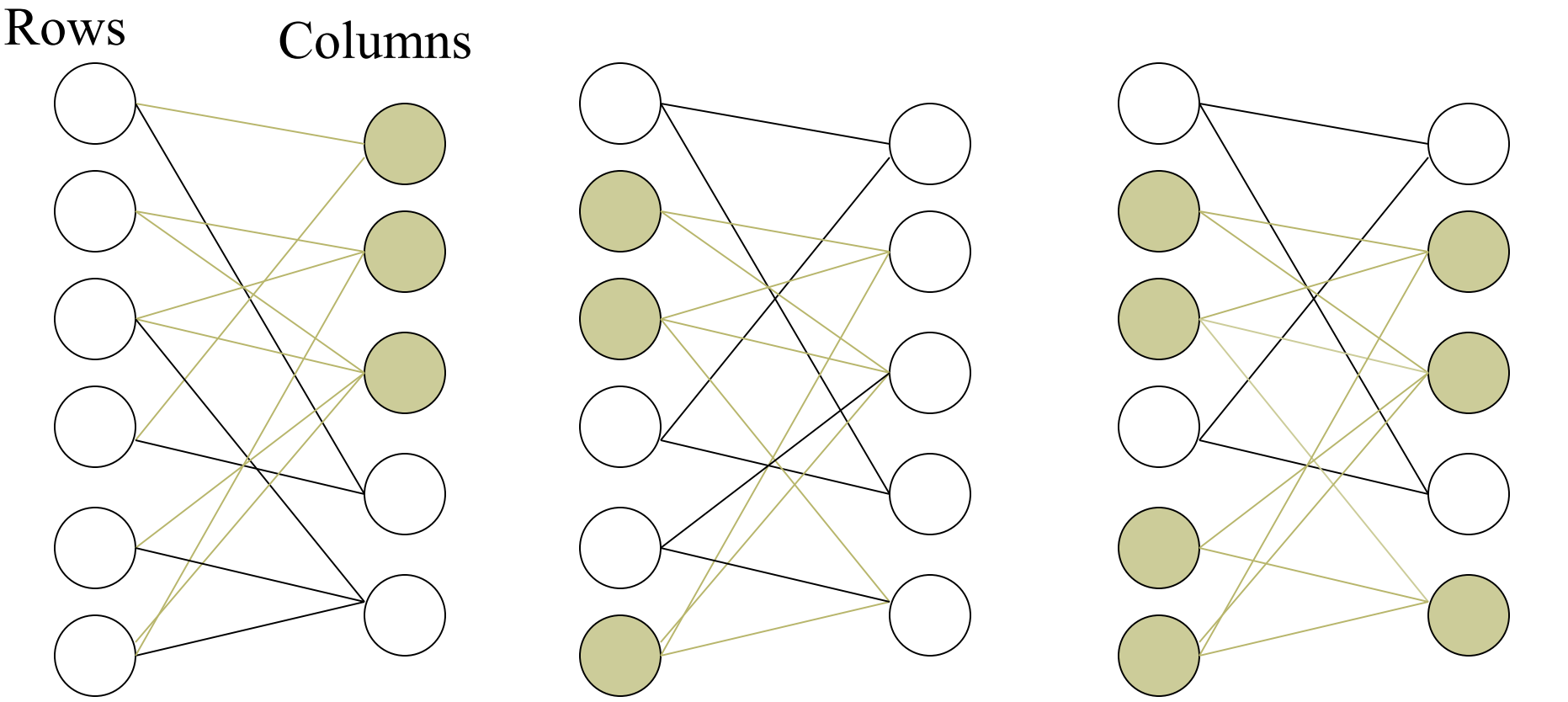
$$[\,1\;\;1\;\;0\;\;0\,]^T \times [\,1\;\;1\;\;1\;\;0\,]$$

Error : 6

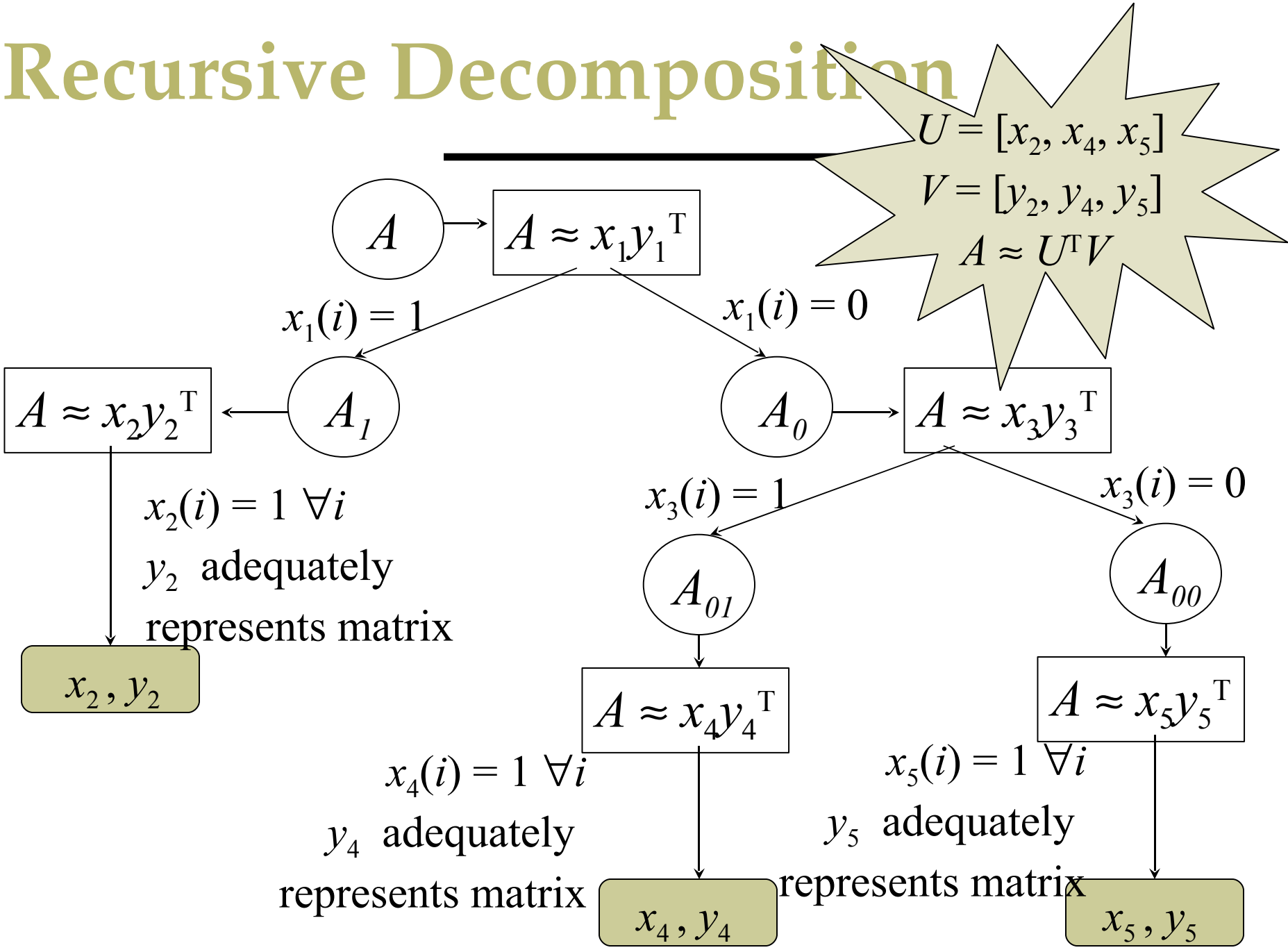
# Alternating Iterative Heuristic

Fix $y$, compute $x$ to maximize $2x^{T}Ay-\|x\|_2^2\|y\|_2^2$.

Then fix $x$, compute $y$. Repeat until convergence.



# Recursive Decomposition



# Stopping Criteria

- Stop whenever both hold:
  - All rows are present in the approximation
  - Pattern represents these rows adequately : Hamming radius less than ε
    - Hamming radius : Maximum Hamming distance to representative pattern
    - ε : 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

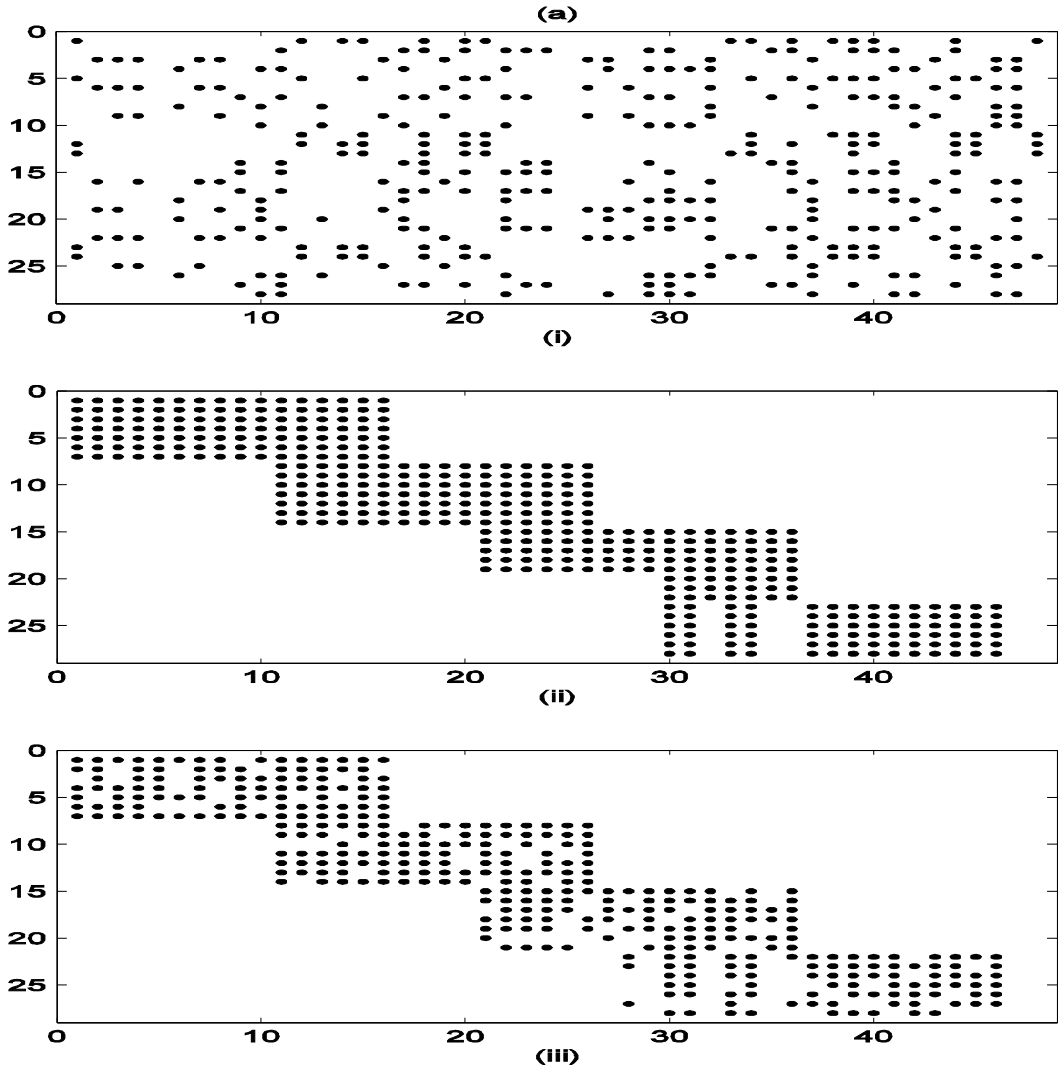
# Initialization of Pattern Vector

- 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

# Time Complexity

- 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 : $\boldsymbol{O(h \times p)}$
  - $h \leq k$ : height of recursion tree

# Visual Results



# Association Rule Mining

**Transaction Set**

$T_1$ = {beer, snacks }
$T_2$ = {milk, butter}
$T_3$ = {milk, butter, eggs}
$T_4$ = {beer, snacks }
$T_5$ = {milk, eggs}
$T_6$ = {milk, butter, eggs}

| | beer | butter | milk | snacks | eggs |
|---|---|---|---|---|---|
| $T_1$ | 1 | 0 | 0 | 1 | 0 |
| $T_2$ | 0 | 1 | 1 | 0 | 0 |
| $T_3$ | 0 | 1 | 1 | 0 | 1 |
| $T_4$ | 1 | 0 | 0 | 1 | 0 |
| $T_5$ | 0 | 0 | 1 | 0 | 1 |
| $T_6$ | 0 | 1 | 1 | 0 | 1 |

**Transaction Matrix**

# Compressing the transaction set

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

## Compressed Transaction Set

$T'_1$ = {beer, snacks } $w(T'_1) = 2$

$T'_2$ = {milk, butter, eggs} $w(T'_2) = 4$

Mine the compressed transaction set!

# Datasets

IBM Quest association data generator

| | # of transactions | # of patterns |
|---|---|---|
| Low | 10K | 20 |
| Medium | 100K | 100 |
| High | 1M | 500 |

**transactions**

| **patterns** | Low | Medium | High |
|---|---|---|---|
| Low | | L100K | |
| Medium | M1M | M100K | H100K |
| High | | H100K | |

# 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

| Min. Support | Time orig. | Time comp. | Rules orig. | Rules comp. | Rules match. | Precision % | Recall % |
|---|---|---|---|---|---|---|---|
| 0.5 | 13.76 | 1.06 | 332395 | 333046 | 331087 | 99.6 | 99.8 |
| 1.0 | 8.01 | 0.39 | 138745 | 139519 | 137200 | 98.3 | 99.0 |
| 1.5 | 5.12 | 0.18 | 60500 | 59580 | 59580 | 100.0 | 98.5 |
| 2.0 | 3.14 | 0.11 | 36061 | 39117 | 35366 | 90.4 | 98.1 |
| 2.5 | 0.86 | 0.03 | 14750 | 14645 | 14645 | 100.0 | 99.3 |
| 3.0 | 0.68 | 0.02 | 11180 | 11070 | 11070 | 100.0 | 99.0 |

# Effect of Parameters

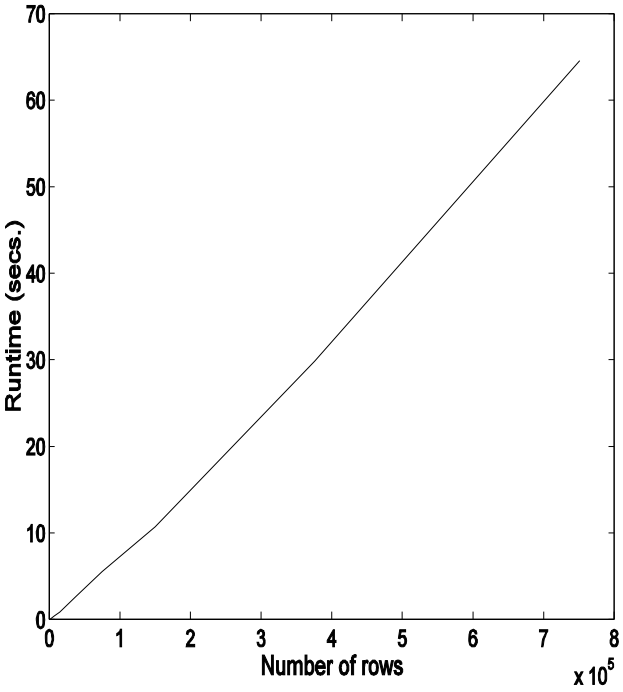
- Robust to increasing number of transactions
- More patterns, harder problem
  - More approximation vectors to preserve quality
  - More preprocessing time
  - Less speedup
  - Can still maintain quality!
- Bound on error
  - Tighter bound, higher quality but less speedup
  - Loose bound does not effect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

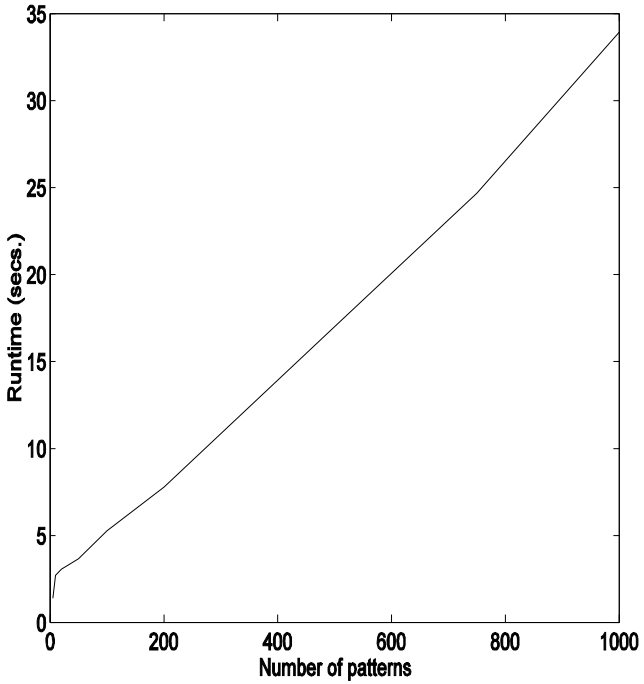
# Performance on Real Data

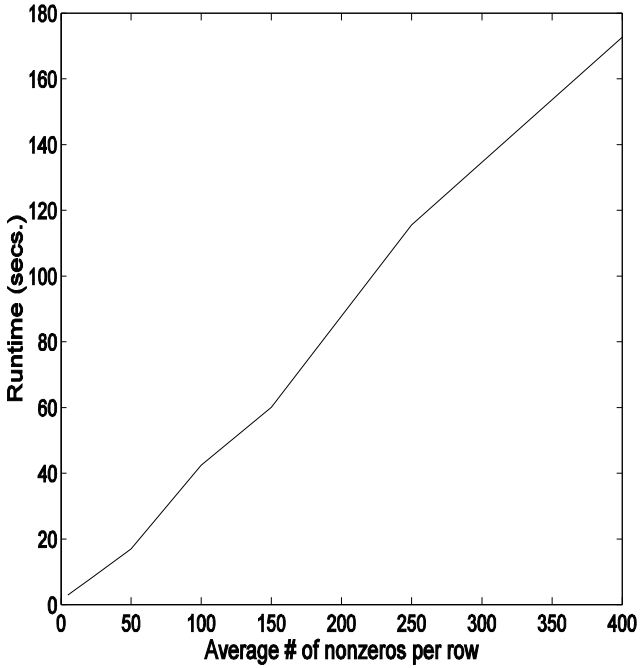
- *Agaricus Lepiota* dataset
  - "mushrooms with bell-shaped and fibrous caps have brown gills"
  - 8124 species
  - 23 categorical attributes → 118 binary attributes
- Decomposition into 1142 approximation vectors in 15.73 seconds

| Min. Support | Time orig. | Time comp. | Rules orig. | Rules comp. | Rules match. | Precision % | Recall % |
|---|---|---|---|---|---|---|---|
| 10.0 | 46.49 | 7.49 | 497813 | 464546 | 464185 | 99.9 | 93.2 |
| 14.8 | 14.98 | 2.19 | 63413 | 63299 | 63210 | 99.9 | 99.7 |
| 19.6 | 7.43 | 1.30 | 32593 | 32589 | 32589 | 100.0 | 99.9 |

# 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 $\times$ binary matrix