Matrix Factorizations over Non-Conventional Algebras for Data Mining

Pauli Miettinen
28 April 2015
Chapter 1.
A Bit of Background
<table>
<thead>
<tr>
<th>long-haired</th>
<th>well-known</th>
<th>male</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✘</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✘</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Data

long-haired
well-known
male

\begin{pmatrix}
\text{male} & 1 \\
\text{long-haired} & 1 \\
\text{well-known} & 1 \\
0 & 0
\end{pmatrix}
Factorization point of view

\[
\begin{pmatrix}
1 \\
1 \\
1 \\
0
\end{pmatrix}
\times
\begin{pmatrix}
1 \\
1 \\
0 \\
0
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
1 \\
0 \\
0
\end{pmatrix}
\times
\begin{pmatrix}
1 \\
1 \\
0 \\
0
\end{pmatrix}
Chapter 2.
Boolean Matrix Factorization
In the sleepy days when the provinces of France were still quietly provincial, matrices with Boolean entries were a favored occupation of aging professors at the universities of Bordeaux and Clermont-Ferrand. But one day...

Gian-Carlo Rota
Foreword to Boolean matrix theory and applications by K. H. Kim, 1982
Boolean products and factorizations

- The **Boolean matrix product** of two binary matrices $A$ and $B$ is their matrix product under the Boolean semi-ring

  $$(A \circ B)_{ij} = \bigvee_{i=1}^{k} a_{ik} b_{kj}$$

- The **Boolean matrix factorization** of a binary matrix $A$ expresses it as a Boolean product of two binary factor matrices $B$ and $C$, that is,

  $$A = B \circ C$$
Matrix ranks

• The (Schein) **rank** of a matrix $A$ is the least number of rank-1 matrices whose sum is $A$
  
  • $A = R_1 + R_2 + \ldots + R_k$

• Matrix is rank-1 if it is an outer product of two vectors

• The **Boolean rank** of binary matrix $A$ is the least number of **binary** rank-1 matrices whose element-wise *or* is $A$
  
  • The least $k$ such that $A = B \circ C$ with $B$ having $k$ columns
Comparison of ranks

• Boolean rank can be less than normal rank
  
  • \( \text{rank}_B(A) = O(\log_2(\text{rank}(A))) \) for certain \( A \)

  \( \Rightarrow \) Boolean factorization can achieve less error than SVD

• Boolean rank is never more than the non-negative rank

\[
\begin{pmatrix}
1 & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 1
\end{pmatrix}
\]
The many names of Boolean rank

- Minimum tiling (data mining)
- Rectangle covering number (communication complexity)
- Minimum bi-clique edge covering number (Garey & Johnson GT18)
- Minimum set basis (Garey & Johnson SP7)
- Optimum key generation (cryptography)
- Minimum set of roles (access control)
Boolean rank and bicliques
Boolean rank and sets

• The Boolean rank of a matrix $A$ is the least number of subsets of $U(A)$ needed to cover every set of the induced collection $C(A)$

• For every $C$ in $C(A)$, if $S$ is the collection of subsets, have subcollection $S_C$ such that

$$\bigcup_{S \in S_C} S = C$$
Approximate factorizations

• Noise usually makes real-world matrices (almost) full rank

• We want to find a good low-rank approximation
  • The goodness is measured using the Hamming distance

• Given $A$ and $k$, find $B$ and $C$ such that $B$ has $k$ columns and $|A - B \odot C|$ is minimized
  • No easier than finding the Boolean rank
The many applications of Boolean factorizations

- Data mining
  - noisy itemsets, community detection, role mining, ...
- Machine learning
  - multi-label classification, lifted inference
- Bioinformatics
- Screen technology
- VLSI design
- ...
The bad news

• Computing the Boolean rank is NP-hard
  • Approximating it is (almost) as hard as Clique [Chalermsook et al. ’14]
• Minimizing the error is hard
  • Even to additive factors [M. ’09]
• Given one factor matrix, finding the other is NP-hard
  • Even to approximate well [M. ’08]
Some algorithms

- Exact / Boolean rank
  - reduction to clique [Ene et al. ’08]
  - GreEss [Bělohlávek & Vychodil ’10]
- Approximate
  - Asso [M. et al. ’06]
  - Panda+ (error & MDL) [Lucchese et al. ’13]
  - Nassau (MDL) [Karaev et al. ’15]
Chapter 3.
Dioids Are Not Droids
Intuition of matrix multiplication

- Element \((AB)_{ij}\) is the inner product of row \(i\) of \(A\) and column \(j\) of \(B\)
Intuition of matrix multiplication

• Matrix $AB$ is a sum of $k$ matrices $a_ib_i^T$ obtained by multiplying the $l$-th column of $A$ with the $l$-th row of $B$
Remember at least this slide

• A matrix factorization presents the input matrix as a sum of rank-1 matrices

• A matrix factorization presents the input matrix as an aggregate of simple matrices

• What “aggregate” and “simple” mean depends on the algebra
Dioids are not droids

• Dioid is also not a diode

• Dioid is an idempotent semiring

\[ S = (A, \oplus, \otimes, \mathbf{0}, \mathbf{1}) \]

• Addition \( \oplus \) is idempotent
  
  • \( a + a = a \) for all \( a \in A \)

• Addition is not invertible
Some examples (1)

- The **Boolean algebra** $B = (\{0,1\}, \lor, \land, 0, 1)$
- The **subset lattice** $L = (2^U, \cup, \cap, \emptyset, U)$ is isomorphic to $B^n$
- The **Boolean matrix factorization** expresses matrix $A$ as $A \approx B \otimes_B C$ where all matrices are Boolean
Some examples (2)

• **Fuzzy logic** $F = ([0, 1], \text{max}, \text{min}, 0, 1)$

• Generalizes (relaxes) Boolean algebra

  • Exact $k$-decomposition under fuzzy logic implies exact $k$-decomposition under Boolean algebra
Fuzzy example

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1
\end{pmatrix}
\cong
\begin{pmatrix}
1 & 0 \\
1 & 1 \\
0 & 1 \\
0 & 1
\end{pmatrix}
\otimes_F
\begin{pmatrix}
1 & 1 & 0 & 1 \\
0 & 1 & 2/3 & 1
\end{pmatrix}
\]
Some examples (3)

• The or-Łukasiewicz algebra

• $\mathfrak{L} = \{[0,1], \text{max}, \otimes_\mathfrak{L}, 0, 1\}$

• $a \otimes_\mathfrak{L} b = \max(0, a + b - 1)$

• Used to decompose matrices with ordinal values [Bělohlávek & Krmelova ’13]
Some examples (4)

- The max-times (or subtropical) algebra
  \[ M = (\mathbb{R}_{\geq 0}, \max, \times, 0, 1) \]

- Isomorphic to the tropical algebra
  \[ T = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0) \]

- \[ T = \log(M) \text{ and } M = \exp(T) \]
Why max-times?

• One interpretation: *Only strongest reason matters* (a.k.a. *the winner takes it all*)
  
  • Normal algebra: rating is a linear combination of movie’s features
  
  • Max-times: rating is determined by the most-liked feature
Max-times example

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1
\end{pmatrix}
\approx
\begin{pmatrix}
1 & 0 \\
1 & 1 \\
0 & 2/3 \\
0 & 1
\end{pmatrix}
\otimes_M
\begin{pmatrix}
1 & 1 & 0 & 1 \\
0 & 1 & 2/3 & 1
\end{pmatrix}
\]

\[
\approx
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 2/3 & 1 \\
0 & 2/3 & 4/9 & 2/3 \\
0 & 1 & 2/3 & 1
\end{pmatrix}
\]
On max-times algebra

• Max-times algebra relaxes Boolean algebra (but not fuzzy logic)

• Rank-1 components are “normal”
  • Easy to interpret?

• Not much studied
On tropical algebras

• A.k.a. max-plus, extremal, maximal algebra

• Much more studied than max-times

  • Can be used to solve max-times problems, but needs care with the errors

  • If \( \| \mathbf{X} - \tilde{\mathbf{X}} \| \leq \alpha \) in max-plus then
    \[ \| \mathbf{X}' - \tilde{\mathbf{X}}' \| \leq M^2 \alpha \] in max-times, where
    \[ M = \exp(\max_{i,j} \{ X_{ij}, \tilde{X}_{ij} \}) \]
More max-plus

• Max-plus linear functions:
  \( f(x) = f^T \otimes x = \max \{ f_i + x_i \} \)

• \( f(\alpha \otimes x \oplus \beta \otimes y) = \alpha \otimes f(x) \oplus \beta \otimes f(y) \)

• Max-plus eigenvectors and values:
  \( X \otimes v = \lambda \otimes v \quad (\max_j \{x_{ij} + v_j\} = \lambda + v_i \text{ for all } i) \)

• Max-plus linear systems: \( A \otimes x = b \)

• Solving in pseudo-P for integer \( A \) and \( b \)
Computational complexity

- If exact $k$-factorization over semiring $\mathbb{K}$ implies exact $k$-factorization over $\mathbb{B}$, then finding the $\mathbb{K}$-rank of a matrix is NP-hard (even to approximate)

- Includes fuzzy, max-times, and tropical

- N.B. feasibility results in $\mathbb{T}$ often require finite matrices
Anti-negativity
and sparsity

• A semiring is **anti-negative** if no non-zero element has additive inverse
  • Some dioids are anti-negative, others not
  • Anti-negative semirings yield sparse factorizations of sparse data
Chapter 4. 
Even More General
Community detection

- Boolean factorization can be considered as a community detection method
- But not all communities are cliques
  - “Beyond the blocks”
- Are matrix factorizations outdated models for graph communities before they even took off?
Generalized outer product

• A generalized outer product is a function \( o(x, y, \theta) \)

  • Returns an \( n \)-by-\( m \) matrix \( A \)

  • If \( x_i = 0 \) or \( y_j = 0 \), then \( (A)_{ij} = 0 \)

  • Compare to \( xy^T \)
Example

• Generalized outer product for biclique core
  • Binary vector $\mathbf{x}$ to select the subgraph
  • Set $C$ to define the nodes in the core
    • $(o(\mathbf{x}, \mathbf{x}, C))_{ij} = 1$ if $\mathbf{x}_i = \mathbf{x}_j = 1$ and exactly one of $i$ and $j$ is in $C$

\[
\begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
1 \\
\vdots \\
1
\end{pmatrix} = C
\]
Generalized decomposition

- A generalized matrix decomposition decomposes input matrix $A$ into a sum of generalized outer products

  $$A = o(x_1, y_1, \theta_1) \oplus o(x_2, y_2, \theta_2) \oplus \ldots$$
  $$\oplus o(x_k, y_k, \theta_k)$$

- Sum can be over any semi-ring

- The generalized rank is defined as expected
Why generalize?

• Provides an unifying framework
• Some algorithms and many computational hardness results generalize well
  • Depend more on the addition $\oplus$ than on the outer product
Some results

• Finding the largest-circumference rank-1 submatrix is NP-hard if the outer product is hereditary
  • Generalizes results for nestedness
• Given a set of binary rank-1 matrices, finding the smallest exact sub-decomposition from them is NP-hard if addition is either OR, AND, or XOR
  • But exact hardness depends on the algebra
Chapter 5.
The Chapter to Remember
Conclusions

• Matrix factorizations are just a way to express complex data as an aggregate of simple parts

• Normal and Boolean algebras are the best-studied ones
  • but by no means the only possible ones

• Generalizing the outer product gives even more versatile language