Nonnegative matrix factorization with polynomial signals via hierarchical alternating least squares

Cécile HAUTECOEUR and François GLINEUR: UCLouvain, CORE & ICTEAM/INMA

Nonnegative Matrix Factorization (NMF)

Problem: Given nonnegative matrix $Y \in \mathbb{R}^{m \times n}$ and integer rank $r > 0$, find nonnegative matrices $A \in \mathbb{R}^{m \times r}$ and $X \in \mathbb{R}^{r \times n}$ such that $\|Y - AX\|$ is minimal ($Y \simeq AX$).

Hence NMF expresses each element in given data $(Y)$ as a nonnegative linear combinations of a few well-chosen common basis elements (in $X$).

NMF with polynomial signals (P-NMF)

A novel extension of usual NMF: The provided data, the $n$ columns of matrix $Y$, are observations of continuous signals, $Y = \{y(t)\}_{t=1}^n$, over interval $t \in [a, b]$. Columns of matrix $A$ are (observations of) polynomials with fixed degree $D$, $A = \{a_k(t)\}_{k=1}^m$, and not vectors anymore.

Two possible cost functions:

- **PS-NMF:** Data are discretized over $m$ points $\{t_j\}_{j=1}^m$:
  \[
  \min_{A,X} \sum_{i=1}^m (y(t_j) - \sum_{k=1}^r a_k(t_j)x_{k,i})^2.
  \]

- **PI-NMF:** Data are polynomials with known coefficients:
  \[
  \min_{A,X} \sum_{i=1}^m \int_a^b (y(t) - \sum_{k=1}^r a_k(t)x_{k,i})^2 dt.
  \]

Hierarchical alternating least squares (HALS): The idea of this algorithm is to update alternatively matrix $A$ and matrix $X$ until convergence. Moreover, during the update of $A$ or $X$, each column is successively updated, individually taking into account updates of previous columns [1].

Usual NMF:

\[
  a_{ij} \leftarrow \frac{Y_{kj} - \sum_{k \neq j} a_{ki}x_{k,j}}{x_{i,j}}, \quad x_{ij} \leftarrow \frac{a_{ij}Y - \sum_{k \neq j} a_{kj}x_{k,j}}{a_{ji}a_{ij}}
  \]

PS-NMF: We consider $B$ the matrix of coefficients of polynomials in $A$, $\Pi$ the Vandermonde matrix ($A = \Pi B$) and $\Pi = \Pi^T \Pi$. Moreover, $F$ is the set of polynomials nonnegatives over interval $[a, b]$. HALS become:

\[
  b_{ij} \leftarrow \frac{\Pi^T Y_{kj} - \sum_{k \neq j} b_{ki}x_{k,j}}{x_{i,j}}, \quad x_{ij} \leftarrow \frac{\Pi^T Y - \sum_{k \neq j} b_{kj}x_{k,j}}{b_{ij}x_{i,j}}
  \]

PI-NMF: We consider $Z$ the matrix of coefficients of polynomials in $Y$ and $M = \{ \Pi^T (\Pi(t))^T \Pi(t) \}_{t \in [a, b]} dt$. Surprisingly, updates are still closed form:

\[
  b_{ij} \leftarrow \frac{Z_{kj} - \sum_{k \neq j} b_{ki}x_{k,j}}{x_{i,j}}, \quad x_{ij} \leftarrow \frac{M \Pi^T Y - \sum_{k \neq j} b_{kj}x_{k,j}}{b_{ij}M x_{i,j}}
  \]

However both PS-NMF and PI-NMF require a new operation $\lfloor \cdot \rfloor_p$ namely projection over the set of nonnegative polynomials.

References and acknowledgments


This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160.

Projected onto the set of nonnegative polynomials

Problem to solve: Given a polynomial $f$, find the polynomial $g$ that is both nonnegative and closest to $f$ over interval $[a, b]$.

\[
  \min_{g} \int_a^b (g(t) - f(t))^2 dt \quad \text{such that } g \in P.
  \]

Parametrization of nonnegative polynomial over $[-1,1]$:

\[
  g(t) \geq 0 \quad \text{for } t \in [-1,1] \quad \Leftrightarrow \quad g(t) = a(t) + (1 - t^2)c(t) \quad \text{where } a(t), c(t) \geq 0 \quad \forall t.
  \]

$\Pi$ has degree $D$ (as $g$) and $c$ has degree $D - 2$ (Markov-Lukacs).

Parametrization of nonnegative polynomial over $\mathbb{R}$:

\[
  g(t) \geq 0 \quad \forall t \quad \Leftrightarrow \quad g(t) = \sum_{i=0}^{r-1} h_i(t)^2 \quad \text{(SOS)}.
  \]

Moreover, the set of nonnegative polynomials of degree $D$ can be represented using an LMI (linear matrix inequality) involving a positive semi-definite matrix of size $(D/2 + 1)$.\]

Algorithm: Using previous information it is possible to define our projection as a semidefinite optimization problem, and solve it with an appropriate solver (such as interior-point MOSEK).

Observations and results

The P-NMF problem has already been considered by Debals and al. in [2]. In this paper, the authors use a least-squares solver in a non-alternative way. We compare our algorithms to usual HALS and the least-square approach, denoted LS.

Signals recovered in $A$:

- Less sensitive to noise than HALS.
- Recover smoother signals.

Performances:

- Error similar to LS.
- CPU time increases slowly with problem size.

Further work

- Accelerate projection.
- Consider other parametrizable signals (such as splines).

Time spent in computations

Mean approximation error

<table>
<thead>
<tr>
<th>number of discretization points</th>
<th>time spent in computations (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10^2</td>
</tr>
<tr>
<td>100</td>
<td>10^3</td>
</tr>
<tr>
<td>150</td>
<td>10^4</td>
</tr>
<tr>
<td>200</td>
<td>10^5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>number of discretization points</th>
<th>mean approximation error (error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10^2</td>
</tr>
<tr>
<td>100</td>
<td>10^3</td>
</tr>
<tr>
<td>150</td>
<td>10^4</td>
</tr>
<tr>
<td>200</td>
<td>10^5</td>
</tr>
</tbody>
</table>