Collaborative Filtering

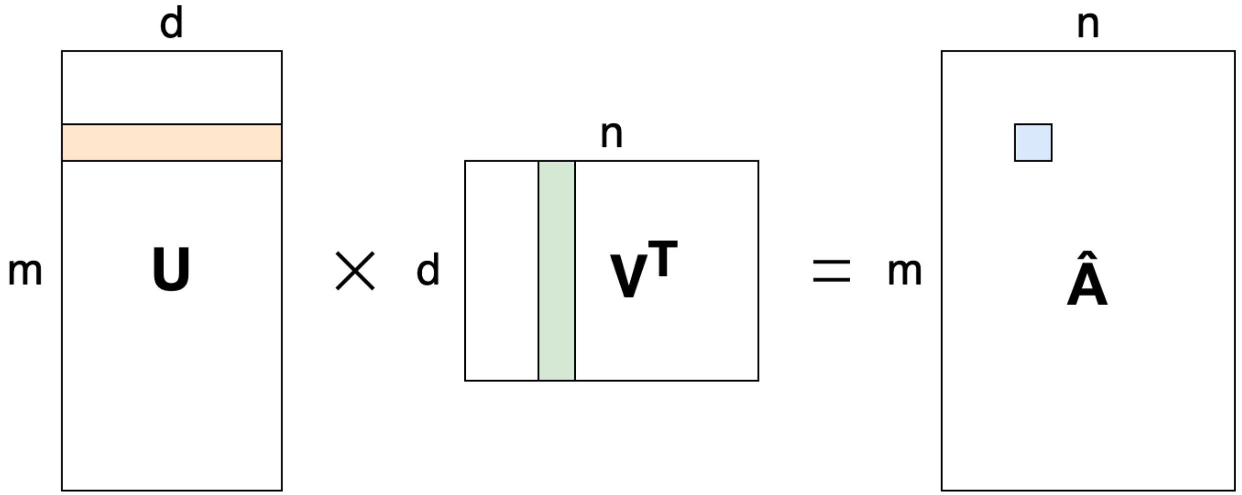
PCTeam: Nicolas Durat, Christian Kayo, Marc Schachtsiek

Problem Statement

Matrix factorization

- Embedding model
- Regression-type optimization problem

- Given the rating matrix $\mathbf{A} \in R^{mxn}$, (*m* : number of users / *n* number of items) the model learn
  - User-Embedding matrix $\mathbf{U} \in \mathbb{R}^{mxd}$, where row ***i*** is the embedding for user ***i***
  - Item-Embedding matrix $\mathbf{V} \in \mathbb{R}^{nxd}$, where row ***j*** is the embedding for item ***j***
- The embeddings are learned such that the product $\mathbf{UV}^{\mathrm{T}} = \widehat{\mathbf{A}}$ approximates $\mathbf{A}$
- Objective function : (Root) Mean Square Error (RMSE)



$$\min_{U,V} \|A - UV^T\|_F^2$$

Approach comparison

Non-negative Matrix Factorization (NMF)

Learns parts-based representation that combine additively

Summary [1]

- Non-negative constraint on matrices
- Decomposes into basis (W) and encoding matrix (H)
- Algorithms converge to local minima and results can vary by starting point

Optimization Problem

$$\min_{W_+, H_+} \frac{1}{2}\|X - WH\|_F^2 + \mu\|H^T H - I\|_F^2$$

Regularization

- *Orthogonality*
    - Introduces a stronger clustering effect to NMF
    - Can be applied to basis, encoding or both matrices (ONMF)
- *Other regularization methods such as Lasso can also be applied*

Sparse Principal Component Analysis (SPCA)

Learns holistic representations that combine linearly

Summary

- Reduction dimension technique adding sparsity structures to construct PCs
- Maximizing variance along a vector and reconstruct PCA with few inputs
- (Penalized) Regression-type optimization problem using elastic-net penalty [2]

Optimization Problem

$$\underset{U,V}{\operatorname{argmin}} \|A - AUV^T\|_F^2 + \lambda \sum_{j=1}^{k} \|v_j\|^2 + \sum_{j=1}^{k} \lambda_{1,j} \|v_j\|_1$$

Regularization

- *Lasso*
    - Shrinks the coefficients towards zero
    - Select variables to produce sparse model
- *Ridge*
    - Variable selection not limited by the n° of observations
    - Ensure PCs reconstruction

Elastic Net

Experiments & Evaluation Metrics

Non-negative Matrix Factorization (NMF)

- **Compare different NMF implementations**
  - (Accelerated) Multiplicative Update (MU)
  - (Accelerated) Hierarchical Alternating Least Squares (HALS)
  - Evaluate convergence rate
  - Evaluate computational complexity
- **Compare NMF to ONMF** [3]
  - Evaluate sparsity of encoding matrix via Misidentification Rate (MR)
  - Evaluate Root Mean Square Error (RMSE)

Sparse Principal Component Analysis (SPCA)

- **Compare different regularized SPCA**
  - Focus on SPCA (Elastic Net) and GPower (Lasso) [4]
  - Evaluate Percentage of Explained Variance (PEV)
  - Evaluate sparsity of encoding matrix via Misidentification Rate (MR)
  - Evaluate Root Mean Square Error (RMSE)

Comparison of approaches

Root Mean Square Error (RMSE)
Misidentification Rate (MR)
Computational Complexity (Time & Space)
Interpretability of U and V matrices

Implementation

Non-negative Matrix Factorization (NMF)

MU update of H

$$H^{(k)} = H^{(k)} \frac{{W^{(k)}}^T M}{{W^{(k)}}^T W^{(k)} H^{(k)}}$$

HALS update of H

for $p = 1{:}\, d$ **do**

$$H_{p:}^{(k)} = max\left(0, H_{p:} + A_{p:} - B_{p:} \times H\right)$$
$$with\ [A = W^T M, B = W^T W]$$

Normal NMF

- Update W <u>exactly</u> once, then H <u>exactly</u> once

Accelerated NMF [5]

- Update W <u>at least</u> once, then H <u>at least</u> once
  - Computations for A, B only performed once per consecutive update
- Hybrid criterion to determine update count with α, ϵ parameters
  - Fixed count $l = 1{:}\, \lfloor 1 + \alpha \rho_H \rfloor$ with $\rho_H = \frac{K+md}{nd+n}$
  - Dynamic stopping $\left\| H^{(k,l+1)} - H^{(k,l)} \right\|_F \leq \epsilon \left\| H^{(k,1)} - H^{(k,0)} \right\|_F$

Adaptation for input with missing data

Sparse Principal Component Analysis (SPCA)

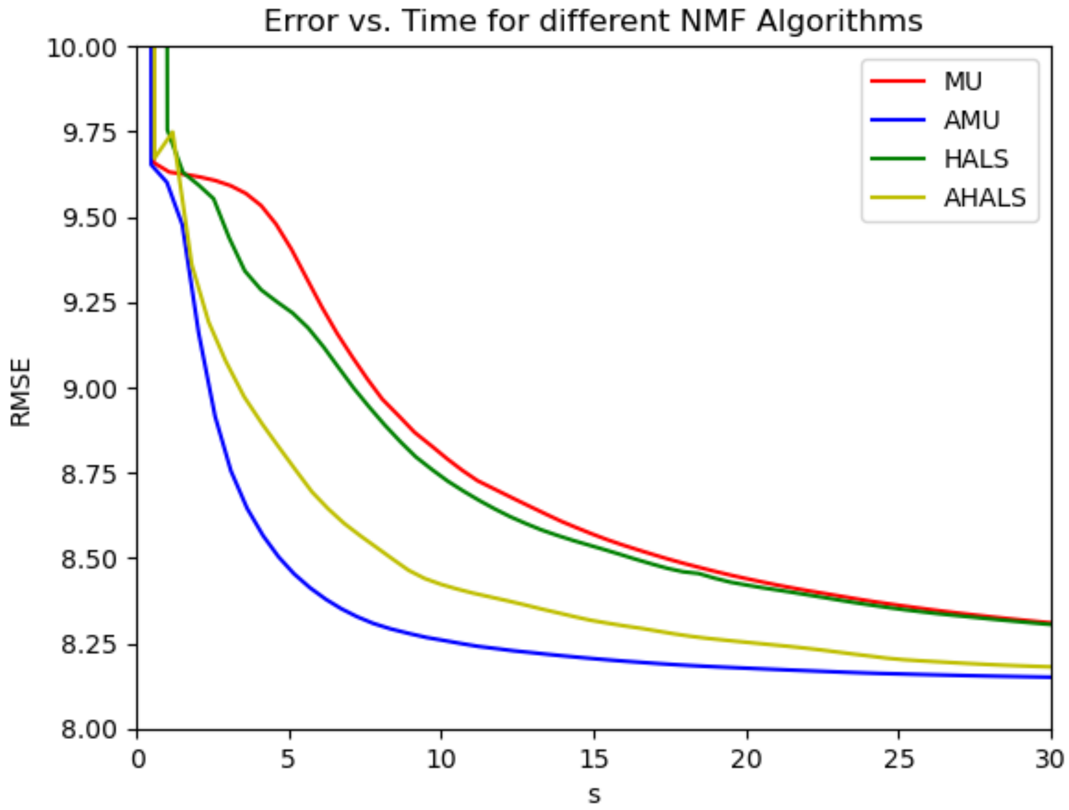
I. **Update B (PC loading) given fixed A (PC weights)**
   A. Elastic net / Lasso optimization by gradient descent
      a. Initialization weights A, L1 & L2 penalties and Gram matrix X^TX
      b. Calculate gradients
      c. Update loadings B

II. **Update A (PC weights) given fixed B (PC loading)**
   A. Compute SVD : $(X^TX)B = UDV^T$
      1) Update $A = UV^T$

III. **Repeat Steps I. and II. until convergence**

IV. **Eigenvectors (PCs axes) ridge normalization**

Preliminary Results



- Sparse rating matrix dataset [6]
- Generally decreasing RMSE
- Accelerated versions outperform original algorithms
- Performance of HALS not clear
  - HALS outperforms MU → as expected
  - AHALS outperforms AMU → not as expected
  - Implementation problem? Related to the dataset? Could a different metric provide better insight?

- *Possible comparison metric to show relative improvement w.r.t. time*

$$E(t) = \frac{e(t) - e_{min}}{e(0) - e_{min}} \quad \text{where} \quad e(t) = \|M - WH\|_F$$

References

[1] Daniel D. Lee, H. Sebastian Seung; Learning the parts of objects by non-negative matrix factorization. *Nature,* 1999; 401: 788–791

[2] Hui Zou, Trevor Hastie and Robert Tibshirani; Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics* 2015; 15 (2): 265–286

[3] Keigo Kimura, Yuzuru Tanaka, Mineichi Kudo; A Fast Hierarchical Alternating Least Squares Algorithm for Orthogonal Nonnegative Matrix Factorization. *Proceedings of the Sixth Asian Conference on Machine Learning,* PMLR 2015; 39: 129-141

[4] Michel Journee, Yurii Nesterov, Peter Richtarik and Rodolphe Sepulchre; Generalized Power Method for Sparse Principal Component Analysis. *Journal of Machine Learning Research* 2010; 11: 517–553

[5] Nicolas Gillis, François Glineur; Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. *Neural Computation* 2012; 24 (4): 1085–1105

[6] https://www.kaggle.com/code/washingtongold/movie-data-conversion/data