STAT 542 Final Project Presentation

Project Group 4:

- Charumeghana Samantula
- Chidharth Balasubramaniam
- David Peng
- Grant Edwards
- Ilia Lomasov
- Yasomitra Sampat Kota

Machine Learning Task/Goal

Task/Goal: Predicting and completing the *restaurant feedback matrix* using different matrix completion algorithms.

Our algorithms were chosen based on the those who improved the most over the 'naive' guess for each of our training datasets.

We also compared the MSE's of our chosen algorithms to see which performed the best

Algorithm	POPULAR	ALS	LIBMF	SVD	UBCF	IBCF

Training Datasets

Tripadvisor.com dataset

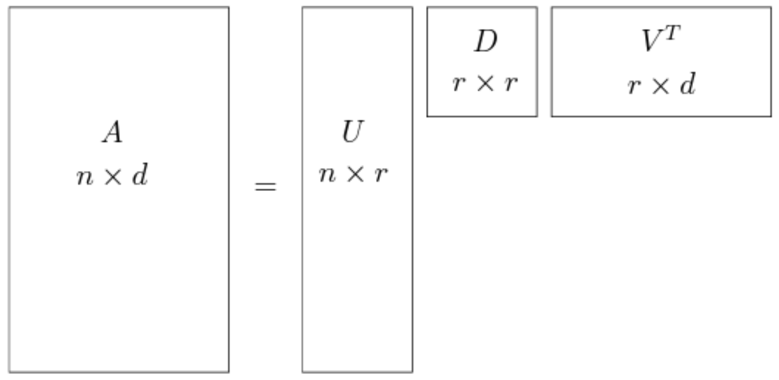
- Contains 980 users, 10 items (art gallery, beach, theatre, etc.)
- Complete matrix: average ratings (0-5) for a given item is used for each user
- Incomplete matrix for MSE calculation: 50% of values were removed randomly using set.seed(100) in R.

MovieLens Dataset

- This dataset describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service
- Dataset contain ratings from 600 users for 9000 different movies

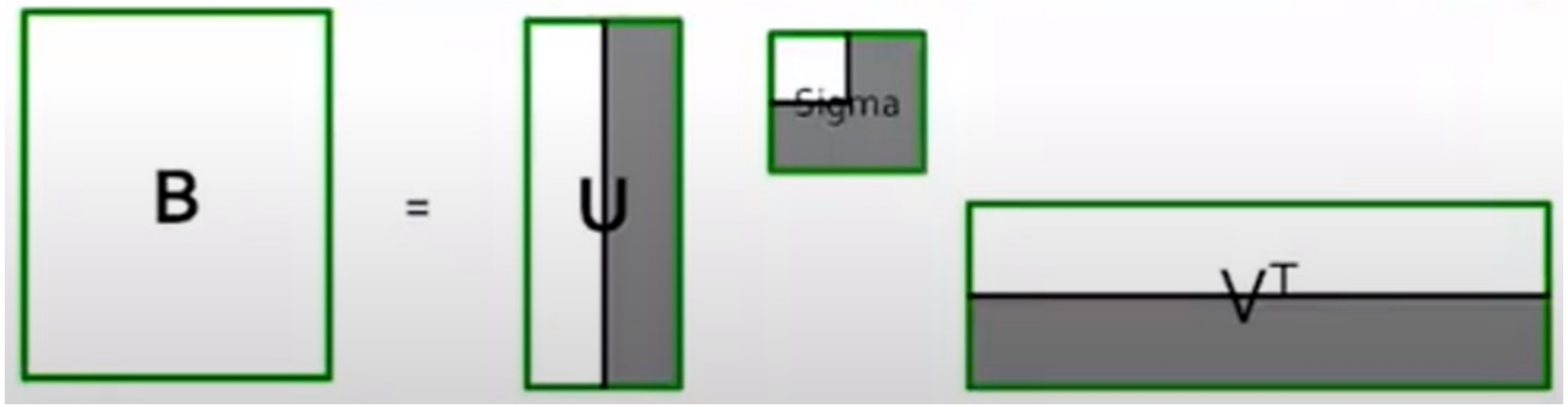
Algorithms Used - Singular Value Decomposition

- A: Input Data Matrix
  - example, n- users, d- movies
- U: Left singular vectors
  - example, n- users, r- genres
  - Eigenvectors of AA^T
- D: Singular values
  - Captures the variation
  - All nonnegative
- V: Right singular values
  - Example, r- genres, d- movies
  - Eigenvectors of $A^T A$
- $A = \sum_i d_{ii}\, u_i\, v_i^T$
- Best rank r approximation of A with respect to Frobenius norm

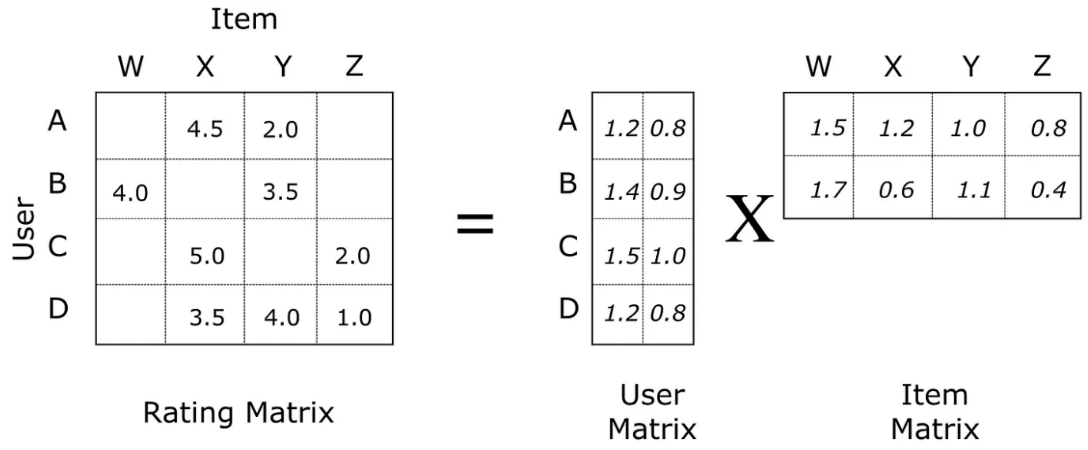


SVD Algorithm

- Impute missing values with column means
- Decompose the imputed matrix to the form, $A = U\Sigma V^T$
- Determine best approximation matrix, $B = U_r \Sigma_r V_r^T$
  - best rank, r computed with respect to Frobenius norm



Algorithms Used - Alternating Least Squares



$$\text{Factorization: } R \approx X^T Y, \text{ where } X \in \mathbb{R}^{k \times n},\ Y \in \mathbb{R}^{k \times m},\ R \in \mathbb{R}^{n \times m}$$

$$\text{Objective: } \min_{X,Y} \sum_{r_{ui}\ observed} \left[(r_{ui}^2 - x_u^T y_i)^2 + \lambda \left(||x_u||_2^2 + ||y_i||_2^2 \right) \right]$$

$$\text{where } x_u \in \mathbb{R}^k \text{ - user, } y_i \in \mathbb{R}^k \text{ - item}$$

Image source: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1

ALS: algorithm

Minimizes the loss functions using gradient descent alternating between X and Y:

```
Initialize $X, Y$
repeat
  for $u = 1 \dots n$ do
    $$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i$$
  end for
  for $i = 1 \dots m$ do
    $$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u$$
  end for
until convergence
```

Implemented in R using `Recommender(data, method = ”ALS”, param=list(lambda=l))`

Algorithm Used - LIBMF

$$\text{Objective: } \min_{P,Q} \sum_{r_{ui}\ observed} \left[f(p_u, q_i; r_{ui}) + \mu_p ||p_u||_1 + \mu_q ||q_u||_1 + 0.5\lambda_p ||p_u||_2^2 + 0.5\lambda_q ||q_i||_2^2\right]$$

$$\text{for } \mathtt{recommenderlab},\ f(p_u, q_i; r_{ui}) = ||r_{ui} - p_u^T q_i||_2^2;\ \mu_p = \mu_q = 0$$

- Optimization solved by SGD.
- Involves parallel computing to reduce the running time vs ALS.
- Plenty of parameters to tune (more on that in the report)

LIBMF Implementation in `R` using `recommenderlab`:

```
Recommender(data, method = ”LIBMF”, param=list(costp_l2,costq_l2))
```

Results: % improvement over ‘naive’ guess

Algorithm	POPULAR	ALS	LIBMF	SVD	UBCF	IBCF
Tripadvisor	58%	**74%**	***78%***	64%	61%	-
MovieLens	23.52%	24.12%	***26.84%***	**25.81%**	25.31%	24.60%

Hyperparameter tuning results:

Algorithm	ALS	LIBMF	SVD
Tripadvisor	$\lambda = 0.02$	$(\lambda_p, \lambda_q) = (0.015, 0.035)$	$r = 2$
MovieLens	$\lambda = 0.1$	$(\lambda_p, \lambda_q) = (0.1, 0.1)$	$r = 10$

Challenges

- High computation time
  - Using LIBMF is computationally faster than ALS while providing similar results, especially when tuning parameters (29 sec for 961-iter. double for loop for LIBMF vs. 186 sec for 20-iter. single loop for ALS)
- High sparsity in the movie dataset
  - Solution: Filtering movies with less than 50 ratings and users with less than 50 ratings
- Parameter Tuning
  - SVD Rank, Number of Users/Items for UBCF/IBCF, ALS/LIBMF penalty
- IBCF doesn’t complete the matrix for TripAdvisor: comparison is impossible

Thank You!