Unsupervised Learning on Amazon Fine Food Reviews

CSCA 5632: Unsupervised Algorithms in Machine Learning

PHPO 4876: Philipp Adrian Pohlmann

University of Colorado **Boulder**

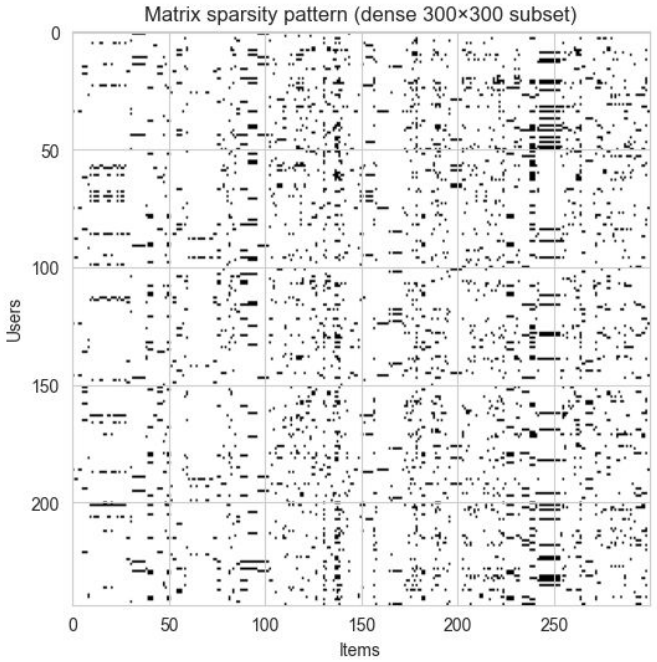
The Problem: How to Recommend Effectively

The Challenge:

- How do we predict user preferences from sparse rating data?
- Users rate only a tiny fraction of available items
- Need to discover latent structure in user-item interactions

Unsupervised Learning Approach:

- Treat as user-item utility matrix problem
- Apply PCA, clustering, and collaborative filtering
- Compare memory-based vs matrix factorization methods



The Dataset: Amazon Fine Food Reviews

Dataset Characteristics

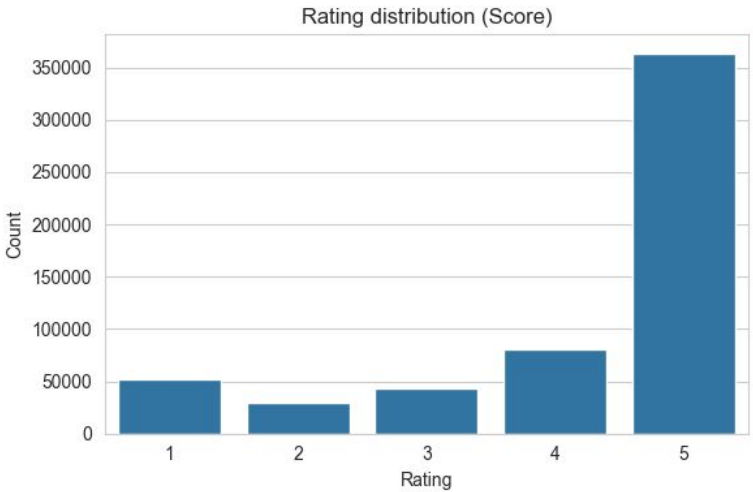
Total Size: 568,454 reviews from Kaggle

- Users: 256,059 unique reviewers
- Items: 74,258 unique products
- Ratings: 1-5 stars (heavily skewed to 5 stars)

Key Challenge:

- Extremely sparse matrix (most users rate very few items)
- Long-tail distribution of user activity and item popularity

Data Split: 90% train (511,608) / 10% validation (56,846)



Exploratory Data Analysis

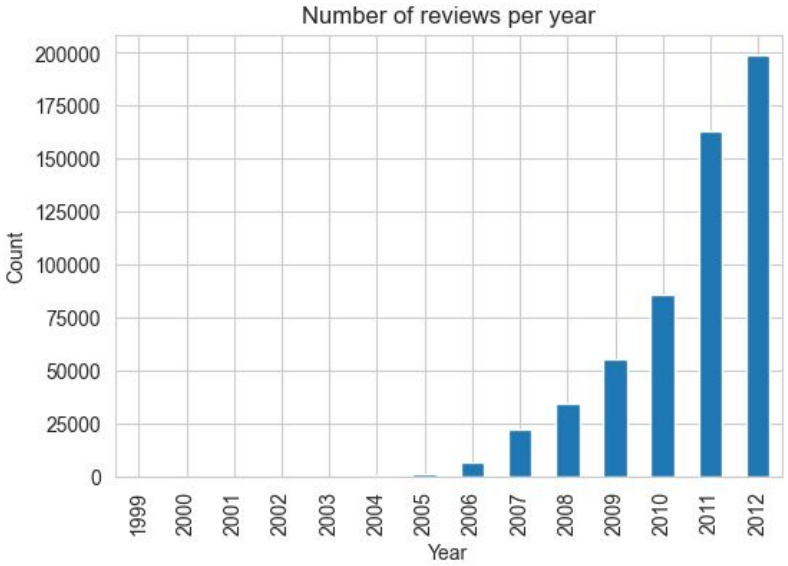
Findings

Rating Distribution (heavily skewed):

- 5 stars: 363,122 (64% of all reviews)
- 4 stars: 80,655 | 3 stars: 42,640 | 2 stars: 29,769 | 1 star: 52,268

Sparsity & Activity:

- Matrix is extremely sparse - most cells are empty
- Long-tail: most users rate few items, most items have few ratings
- Only 1,632 items have ≥50 ratings (2% of all items)



Dimensionality Reduction: PCA Analysis

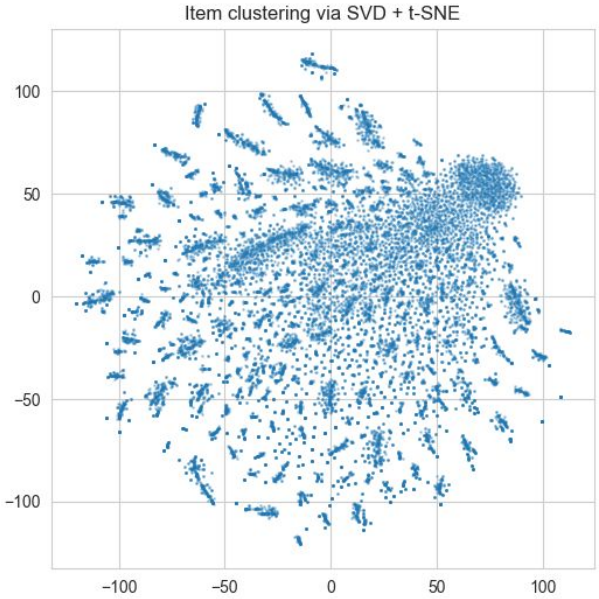
PCA on Item Rating Vectors

Goal: Understand latent structure of items

- Applied Truncated SVD to reduce dimensionality
- Projected items into low-dimensional space

Key Finding:

- Only 3.2% variance explained by first component
- Variance spread across many small components
- Typical for extremely sparse recommender systems
- No dominant latent factor - complex multi-dimensional structure



Clustering: K-Means on Item Embeddings

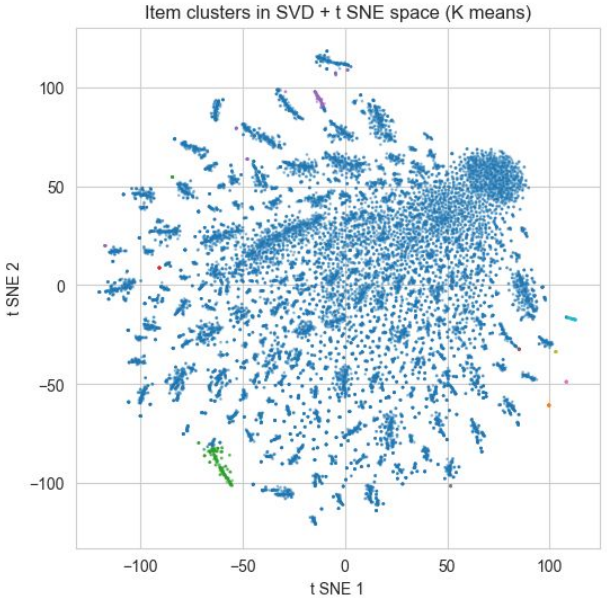
Item Clustering in Latent Space

Applied K-means clustering to item embeddings

- Grid search over different K values
- Selected K=10 as compromise between granularity and interpretability

Purpose:

- Discover natural item groupings based on rating patterns
- Items in same cluster rated similarly by similar users
- Useful for understanding product categories in latent space



Three Collaborative Filtering Approaches

Model Comparison Strategy

1. Global Mean Baseline:

- Predicts same constant rating for all users and items
- Simplest possible baseline - no personalization

2. Item-Based Collaborative Filtering:

- Memory-based: uses item-item cosine similarity
- Predicts from k=20 most similar items
- Restricted to items with ≥50 ratings for computational efficiency

3. Matrix Factorization (SGD):

- Learns 30-dimensional latent factors for users and items
- Includes user and item biases + L2 regularization
- Trained with SGD over 5 epochs

```
Global mean rating (train): 4.182245000078185
Epoch 1 / 5  -  approx train RMSE: 1.2240
Epoch 2 / 5  -  approx train RMSE: 1.1821
Epoch 3 / 5  -  approx train RMSE: 1.1492
Epoch 4 / 5  -  approx train RMSE: 1.1178
Epoch 5 / 5  -  approx train RMSE: 1.0854
```

Results: Matrix Factorization Wins

Validation Performance Comparison

Model Performance (Validation Set):

- Global Mean: RMSE ~1.90 | MAE ~0.80
- Item-Based CF: RMSE ~1.80 | MAE ~0.58
- **Matrix Factorization: RMSE ~1.16 | MAE ~0.89 ✓**

Key Observations:

- Matrix Factorization achieves best RMSE (35% improvement)
- Item-based CF strong on MAE - good at local neighborhoods
- Global mean surprisingly decent - ratings heavily skewed to 4-5 stars

Analysis: Why Matrix Factorization Wins

Understanding Model Performance

1. Matrix Factorization captures global structure:

- Compresses high-dimensional matrix into 30 latent factors
- Can generalize across many more items than local methods
- Learns user preferences and item characteristics simultaneously

2. Item-based CF limited by locality:

- Only considers k=20 nearest neighbors
- Restricted to popular items (≥50 ratings) for efficiency
- Can't leverage broader patterns across entire matrix

3. Trade-offs matter in practice:

- MF requires more training time and hyperparameter tuning
- Item-based CF simpler to implement and update incrementally

Key Takeaways

#1 Extreme sparsity requires careful model choice

- Most users rate very few items, most items have few ratings
- Models must generalize from limited observations
- PCA shows variance spread across many dimensions (3.2% in PC1)

#2 Matrix factorization captures global structure best

- RMSE 1.16 vs 1.80 for item-based CF (35% improvement)
- Learns latent factors that generalize across entire matrix

#3 Real-world trade-offs matter beyond RMSE

- Training complexity, update speed, and coverage are critical
- Skewed rating distribution (64% are 5-star) affects all models

Thank You!

University of Colorado **Boulder**