PMAI LAB

Mathematics • Python • Artificial Intelligence

LINEAR ALGEBRA FOR MACHINE LEARNING

Interactive Case Study 02

E-Commerce Recommendation with Cosine Similarity and SVD

Your role Act as a recommendation-system team. Compare neighbor-based and low-rank models, then recommend unseen products for a target user.

Dr. Hira Benish

Associate Professor of Mathematics

Student Case-Study Handout | Suggested time: 90–120 minutes

1 Platform Brief and Learning Challenge

An e-commerce platform records user engagement with six technology products. A larger score indicates stronger observed interest through browsing, cart activity, or purchase behavior. The platform must recommend two unseen products to User E.

The dataset is a realistic classroom simulation. A score of zero means no observed interaction; it does not prove that the user dislikes the product.

Learning Outcomes

By completing this case study, students will be able to:

- represent user-item behavior as a matrix;
- calculate cosine similarity between user vectors;
- construct a weighted neighbor recommendation;
- compute a truncated SVD and reconstruction error;
- interpret user and item latent factors as embeddings;
- compare recommendations across different ranks;
- identify cold-start, missing-data, and feedback-loop risks; and
- communicate a ranked recommendation with evidence.

1.1 Platform Decision

Produce a ranked list of two unseen products for User E. Your group must compare:

1. user-based cosine recommendation; and
2. rank-k SVD recommendation for at least three values of k.

Practice Tasks

Select the method and rank you would test first on the platform. Defend your decision using similarity structure, reconstruction error, ranking behavior, and practical limitations.

1.2 Team Roles

Role	Responsibility
Data architect	Define rows, columns, zeros, and observed entries
Similarity analyst	Calculate neighbor scores and weighted ranking
SVD analyst	Compare singular values, ranks, and latent factors
Product lead	Recommend an experiment and identify risks

2 User-Item Interaction Matrix

Products are coded as follows:

$$\begin{aligned} P_1 &= \text{Laptop Stand}, & P_2 &= \text{Wireless Mouse}, \\ P_3 &= \text{Keyboard}, & P_4 &= \text{Webcam}, \\ P_5 &= \text{Headphones}, & P_6 &= \text{Speaker}. \end{aligned}$$

User	P_1	P_2	P_3	P_4	P_5	P_6
A	5	4	5	0	0	1
B	4	5	4	1	0	0
C	0	1	0	5	4	5
D	1	0	0	4	5	4
E	5	4	0	0	1	0
F	0	0	1	5	4	4

The interaction matrix is

$$R \in \mathbb{R}^{6\times 6}.$$

Rows are user vectors and columns are product vectors. User E has observed interactions with P_1, P_2, and P_5, so only P_3, P_4, and P_6 are eligible for recommendation in this exercise.

2.1 Pattern Prediction - 5 Minutes

Before calculating anything:

1. Which user appears most similar to User E?
2. Which unseen product should be ranked first?
3. Do the products appear to form more than one interest group?
4. What might the two strongest latent factors represent?

Observed versus Missing

Treating every zero as a true rating of zero is a simplifying classroom assumption. In a production system, unobserved entries require special treatment because a user may simply never have seen the item.

Always remove already-observed products from the final recommendation list, even if the model assigns them the highest scores.

3 Activity 1: Cosine Neighbor Recommendation

For users i and j with row vectors $\mathbf{r}_i$ and $\mathbf{r}_j$,

$$s_{ij} = \frac{\mathbf{r}_i^T \mathbf{r}_j}{\|\mathbf{r}_i\|_2 \|\mathbf{r}_j\|_2}.$$

3.1 Manual Warm-Up

For Users E and A,

$$\mathbf{r}_E^T \mathbf{r}_A = 5(5) + 4(4) + 1(0) = 41,$$
$$\|\mathbf{r}_E\|_2 = \sqrt{42}, \qquad \|\mathbf{r}_A\|_2 = \sqrt{67}.$$

Therefore,

$$s_{EA} = \frac{41}{\sqrt{42}\sqrt{67}} \approx 0.773.$$

Practice Tasks

Calculate s_{EB} manually. Then use Python to calculate similarity between User E and every other user. Rank the neighbors from most to least similar.

3.2 Weighted Product Score

For an unseen product j, calculate

$$\text{score}(E, j) = \frac{\sum_{i \neq E} s_{Ei} r_{ij}}{\sum_{i \neq E} |s_{Ei}|}.$$

After scoring, set already-observed products to $-\infty$ before selecting the largest values.

3.3 Decision Pause

Discuss these questions in pairs:

1. Should all neighbors be used, or only the top two?
2. Does a large interaction magnitude make two users appear more similar?
3. How would mean-centering change the interpretation?
4. Which business metric should be used to evaluate the top two recommendations?

Machine-Learning Connection

Cosine recommendation uses inner products and norms. It is interpretable and fast for small problems, but sparse behavior and changing user interests can make similarities unreliable.

4 Activity 2: Low-Rank SVD Recommendation

Compute

$$R = U\Sigma V^T$$

and form the rank-k approximation

$$R_k = U_k \Sigma_k V_k^T.$$

The reconstructed entry $(R_k)_{ij}$ becomes a latent-factor score for user i and product j.

4.1 Embedding Interpretation

Define

$$P = U_k \Sigma_k^{1/2}, \qquad Q = V_k \Sigma_k^{1/2}.$$

Then

$$R_k = PQ^T, \qquad (R_k)_{ij} = \mathbf{p}_i^T \mathbf{q}_j.$$

The rows of P are user embeddings and the rows of Q are product embeddings.

4.2 Rank Experiment

For $k = 1, 2, 3, 4$, record

$$\text{relative error}(k) = \frac{\|R - R_k\|_F}{\|R\|_F}$$

and the highest-scoring unseen product for User E.

k	Relative error	Top unseen product	Interpretation
1	________	________	________
2	________	________	________
3	________	________	________
4	________	________	________

Rank Is a Model Choice

A lower reconstruction error does not automatically produce better unseen-item ranking. Larger k can reproduce observed zeros and noise rather than generalizable preference structure. Select rank using held-out interactions, not the full matrix.

4.3 Group Challenge

Hide one positive interaction from each user, fit the model to the remaining matrix, and test whether the hidden item appears in the top two recommendations. Report recall at 2.

5 Python Lab

```python
import numpy as np

products = np.array([
    "Stand", "Mouse", "Keyboard",
    "Webcam", "Headphones", "Speaker"
])
R = np.array([
    [5,4,5,0,0,1],
    [4,5,4,1,0,0],
    [0,1,0,5,4,5],
    [1,0,0,4,5,4],
    [5,4,0,0,1,0],
    [0,0,1,5,4,4]
], dtype=float)

target = 4                          # User E
user = R[target]
norms = np.linalg.norm(R, axis=1)
similarity = R @ user / (norms * np.linalg.norm(user))
similarity[target] = 0.0

neighbor_scores = (
    (similarity[:, None] * R).sum(axis=0) /
    np.abs(similarity).sum()
)
neighbor_scores[user > 0] = -np.inf
print("Similarities:", similarity)
print("Cosine recommendation:",
      products[np.argmax(neighbor_scores)])

U, s, Vt = np.linalg.svd(R, full_matrices=False)
print("Singular values:", s)

for k in [1, 2, 3, 4]:
    Rk = (U[:, :k] * s[:k]) @ Vt[:k, :]
    error = np.linalg.norm(R - Rk, "fro") / np.linalg.norm(R, "fro")
    scores = Rk[target].copy()
    scores[user > 0] = -np.inf
    order = np.argsort(scores)[::-1]
    print(k, error, products[order[:2]])

# User and item embeddings for k = 2
k = 2
root_s = np.sqrt(s[:k])
user_embedding = U[:, :k] * root_s
item_embedding = Vt[:k, :].T * root_s
assert np.allclose(
    user_embedding @ item_embedding.T,
    (U[:, :k] * s[:k]) @ Vt[:k, :]
)
```

Visualize the two-dimensional user and product embeddings with a scatter plot. Label each point and explain the two visible clusters.

6 Self-Check Results and Product Decision

Complete the activities independently before using this page to check your results.

6.1 Cosine Checkpoints

For User E, the similarities are approximately:

User A	User B	User C	User D	User F
0.773	0.810	0.151	0.203	0.081

The weighted unseen-product scores are approximately:

$$P_3 = 3.56, \qquad P_4 = 1.38, \qquad P_6 = 1.32.$$

The cosine method therefore ranks **Keyboard** first.

6.2 SVD Checkpoints

The singular values are approximately

$$13.95, \quad 11.71, \quad 3.71, \quad 1.83, \quad 1.06, \quad 0.41.$$

k	Relative error	Highest unseen score for User E
1	0.666	Webcam
2	0.229	Keyboard
3	0.115	Keyboard
4	0.061	Keyboard

Initial Product Recommendation

Both cosine similarity and rank-2 SVD support Keyboard as the first candidate. A reasonable second candidate is Webcam, but its evidence is weaker. The platform should test the ranked list on held-out interactions or through a controlled online experiment before deployment.

6.3 Discussion Questions

1. Why does rank 1 recommend Webcam while rank 2 recommends Keyboard?
2. What do the first two singular directions appear to capture?
3. Why can reconstruction error continue decreasing while recommendation quality worsens?
4. How would popularity bias affect new or niche products?

Machine-Learning Connection

The final decision combines vector similarity, low-rank approximation, embeddings, ranking, and validation rather than relying on one matrix output.

7 Student Submission and Assessment

7.1 Required Submission

Submit one notebook and a one-page product memo containing:

1. the meaning, dimensions, and limitations of the interaction matrix;
2. manual cosine calculations for Users E–A and E–B;
3. neighbor similarities and ranked unseen-product scores;
4. singular values and rank-k results for $k = 1, 2, 3, 4$;
5. a two-dimensional embedding visualization;
6. the final top-two recommendation and proposed evaluation method; and
7. discussion of cold start, missing interactions, and feedback loops.

7.2 Assessment Rubric

Criterion	Marks
Matrix representation and similarity calculations	20
SVD, low-rank approximation, and embeddings	25
Python implementation and reproducibility	20
Ranking evaluation and model comparison	20
Interpretation, risks, and communication	15
Total	100

7.3 Extension Tasks

- Compare user-based and item-based cosine recommendation.
- Mean-center observed interactions before factorization.
- Add regularized matrix-factorization training with gradient descent.
- Design a strategy for a new user with no interaction history.

Practice Tasks

Write a 100-word reflection: Which model would you test first, what evidence supports it, and what failure would you monitor most carefully?

Core connection: User-item matrix → cosine similarity → SVD embeddings → ranked recommendation → evaluation.