Recommender Systems - Simplified Guide

What Are Recommender Systems?

Recommender systems suggest items you might like based on patterns in data. Think Netflix recommendations or Amazon's "customers who bought this also bought..."

Two Main Approaches

1. Collaborative Filtering (Learning from the crowd)

Uses ratings from many users to find patterns

Two types:

Memory-Based (Simple but effective)

- **User-based**: "Users similar to you also liked..."
- **Item-based**: "People who liked this also liked..."

Model-Based (More scalable)

- Uses math (like SVD) to find hidden patterns
- Better for large datasets
- Can predict even with sparse data

2. Content-Based (Learning from features)

Recommends based on item attributes (genre, director, etc.)

How Collaborative Filtering Works

Step 1: Create a User-Item Matrix

	Movie1	Movie2	Movie3
User1	5	?	3
User2	4	2	?
User3	?	1	4

(? = no rating yet)

Step 2: Calculate Similarity

Find similar users or items using **cosine similarity -** measures the angle between rating vectors

Step 3: Make Predictions

- For user-based: "User A is similar to you and rated Movie X highly"
- For item-based: "Movie A is similar to movies you liked"

Key Concepts Simplified

Cosine Similarity

Imagine two arrows pointing in space. If they point in similar directions = high similarity, opposite directions = low similarity.

Matrix Factorization (SVD)

Break down the big user-item matrix into smaller pieces that capture hidden patterns:

- One piece represents user preferences
- One piece represents item characteristics
- Multiply them back together to fill in missing ratings

Real Example Walkthrough

MovieLens Dataset: 943 users, 1682 movies, 100,000 ratings

1. **Split data**: 75% training, 25% testing
2. **Build matrix**: rows = users, columns = movies, values = ratings
3. **Calculate similarity**: between all users (or items)
4. **Predict ratings**: for movies users haven't seen
5. **Evaluate**: Compare predictions to actual test ratings using RMSE

Evaluation: RMSE (Root Mean Squared Error)

Measures how far off predictions are from actual ratings:

- Lower RMSE = better predictions
- RMSE of 3.1 means predictions are off by ~3 points on average

---

Pros and Cons

Memory-Based CF

✅ Easy to implement
✅ Explainable ("similar users liked...")
❌ Doesn't scale well
❌ "Cold start" problem (new users/items)

Model-Based CF

✅ Scales to millions of users/items
✅ Handles sparse data better
✅ Can discover hidden patterns
❌ Less explainable
❌ Still struggles with cold start

The Code in Simple Terms

python

```
# 1. Load movie ratings data
df = pd.read_csv('u.data', sep='\t')

# 2. Create user-item matrix (users × movies)
matrix = np.zeros((n_users, n_items))

# 3. Calculate similarity between all users
user_similarity = pairwise_distances(matrix, metric='cosine')

# 4. Predict ratings based on similar users
prediction = similarity.dot(ratings) / similarity.sum()

# 5. Evaluate predictions
rmse = sqrt(mean_squared_error(prediction, actual))
```

What SVD Does:

```python
# Break matrix into 3 pieces
u, s, vt = svds(matrix, k=20)  # k = number of hidden features

# Multiply back together for predictions
predictions = u @ np.diag(s) @ vt
```

Real-World Applications

- **Netflix**: Movie/show recommendations
- **Spotify**: Music suggestions

- **Amazon**: Product recommendations
- **YouTube**: Video suggestions

Key Takeaway

Recommender systems work by finding patterns in user behavior:

- **Collaborative Filtering** = "People like you enjoyed..."
- **Content-Based** = "This is similar to what you liked..."
- **Hybrid** = Combine both for best results

The math looks complex, but the idea is simple: use past behavior to predict future preferences!