Lecture 15

Recommender Systems

Recommender systems are algorithms designed to suggest relevant items to users based on various information sources. They are widely used in e-commerce, streaming platforms, social media, and more.

Key Components:

- Users $(U_1, U_2, \dots, U_n)$: The individuals or entities receiving recommendations.
- Items $(M_1, M_2, \dots, M_m)$: The products, movies, songs, or content items to recommend.
- Interactions / Ratings (R_{ij}): Historical data indicating user preferences, e.g., explicit ratings, clicks, purchases.

Challenges in Recommender Systems:

1. Scale:
    a. Millions of users and items create massive datasets.
    b. Efficient algorithms needed to handle large-scale, real-time recommendations.
2. Cold Start Problem:
    a. How to recommend for:
    b. New users with no prior history
    c. New items without ratings
3. Data Sparsity:
    a. Most users rate only a small fraction of items.
    b. Matrix of user-item interactions is sparse, complicating similarity and model fitting.

How to Predict Ratings?

1. Neighbourhood methods:
    a. Data exists for both users and items
    b. Find similar users or similar items and predict based on their ratings.
2. Content-based filtering:
    a. Content-based filtering
    b. Recommend items similar to what the user liked based on item features.
3. Collaborative filtering:
    a. Only ratings data is available
    b. Leverages patterns of users' ratings to predict unknowns, without explicit features.

Neighbourhood Methods

Group users or items into “neighbourhoods” using similarity measures.

Types:

- User-User Similarity: Recommend items liked by users who have similar tastes.
- Item-Item Similarity: Recommend items similar to those a user already likes.

Pros:

- Intuitive and easy to explain.
- No explicit model training required.
- Can handle new users and items reasonably well.

Challenges:

- Users have different rating behaviours (bias).
- Ratings can change over time, requiring model updates

It's challenging to capture the right features that describe movies and users. For instance, movies might have different intensities of "comedy", but if you simply tag them as "comedy", you lose this nuance.

Users and movies may not be accurately described by simple categorical tags or direct features, making manual characterization limited or imprecise.

Aim is to discover the best features for representing users and movies automatically, without relying on inaccurate or incomplete manual feature definitions.

Content-Based: Assumes you already have features for movies (like genre, director, cast etc.) and want to learn the preferences/feature profile of users.

Collaborative Filtering: Seeks to learn features for both users and movies based on behavioural patterns (like ratings, clicks), not necessarily relying on predefined movie features.

Feature Extraction - Content-Based (Feature-to-Movie Matrix)

Suppose we have some features characterizing movies (e.g., Romance, Action). We can represent this as a feature-to-movie similarity matrix:

Features	**M1**	**M2**	**M3**	**M4**
F1 (Romance)	0.9	1	0.1	0
F2 (Action)	0	0.01	1	0.9

This matrix tells you how much each movie possesses a feature. For example, Movie 2 (M2) is strongly Romance (1), slightly Action (0.01), Movie 3 is strong on Action (1), and so on.

Question: Given the feature-to-movie matrix, how do you predict the rating for any user-movie pair, say User 2 and Movie 1?

To do that, we also need a user-to-feature similarity matrix, representing how much each user prefers each feature:

Users	**F1 (Romance)**	**F2 (Action)**
U1	5	0
U2	5	0
U3	0	5

We then multiply the user-to-feature matrix (P) with the feature-to-movie matrix (Q) to predict rating.

Example: User 2, Movie 1:

$$R_{21} = P^{(2)^{\mathrm{T}}} \cdot Q^{1} = [5, 0] \cdot [0.9, 0] = 5 * 0.9 + 0 * 0 = 4.5$$

So, user 2 is likely to rate movie 1 as 4.5 based on their preference and the movie's features.

Now we need to find for best user features

- P^j is the user-to-feature preference vector for user j.
- Q^i is the movie-to-feature vector for movie i.
- r_{ij} is the observed rating of user j for movie i.

Objective: How can we find the best user features P^j is given movie features Q^i and known ratings?

So, our **Optimization Problem** becomes:

$$P^i = arg\min_P \frac{1}{|M^j|} \sum_{i \in M^j} \left(P^T Q^i - r_{ij}\right)^2 + \lambda \left|\left|p\right|\right|^2$$

It is a convex optimization problem and can be It can be solved directly and exactly using:

- Closed-form solution (normal equations with regularization).
- Or numerical methods for convex optimization if the problem is large.

Thus,

1. The goal is to find the user preference P^i that minimizes the difference between predicted and actual ratings for all movies i rated by user j
2. $|M^j|$ is the number of movies rated by user j to normalize vector
3. The first term is the mean squared error of the prediction.
4. The second term, with parameter λ, is a regularization term. This penalizes overly large preferences to prevent overfitting and improve generalization.
5. This regularization term keeps the learned user preferences P small in magnitude.

Challenge with content-based approaches:

- We need predefined good features $f_1, \dots, f_k$ describing movies.
- We also need to find user preference vectors $p^1, \dots, p^n$.
- What if we don't have high-quality movie features in advance?
- Can we learn both sets of features automatically from the data?

Feature Extraction - Collaborative Filtering (Optimization Problem)

P = matrix of user features/preference vectors. (User-to-feature.)

Q = matrix of movie feature vectors. (Feature-to-movie.)

The key equation: $R = PQ$

Where R is is the user-to-movie rating matrix.

So, our **Optimization Problem** becomes:

$$arg\min_{P,Q} \sum_{i,J \in R} \left(r_{ij} - P_I^T Q_j\right)^2 + \lambda \left(||P||_F^2 + ||Q||_F^2 \right)$$

- This minimizes squared error between observed ratings r_{ij} and predicted ratings $P_I^T Q_j$.
- P_I^T and Q_j are the feature vectors for user i and movie j, respectively.
- Regularization term $\lambda \left(||P||_F^2 + ||Q||_F^2 \right)$ reduces overfitting.
- The summation only runs over known ratings (where data exists), handling sparsity.

It is a non-convex problem, can be solved using:

- Alternating Least Squares (ALS) offers a practical way to solve the joint optimization despite non-convexity.
- Gradient descent or stochastic gradient descent (SGD).