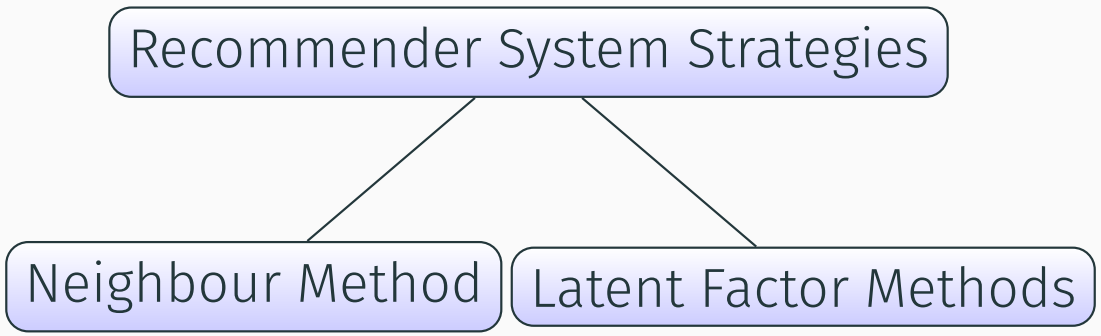
EE5327 : Optimization

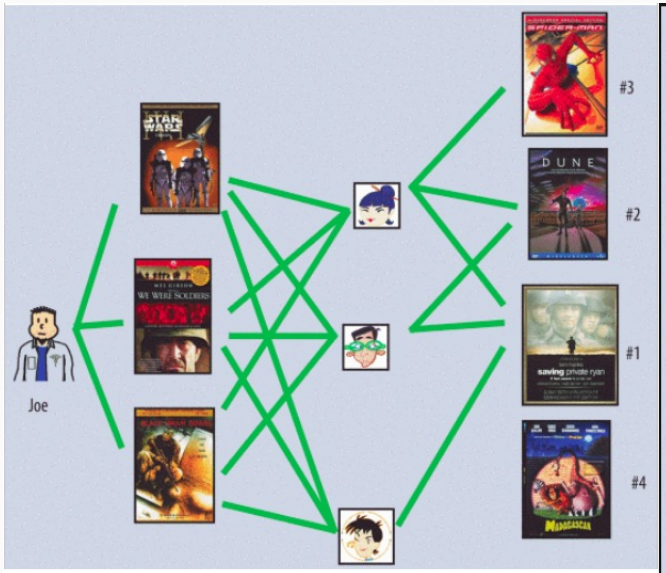
Harsh Raj - MA17BTECH11003
Aravind Reddy K V - MA17BTECH11010

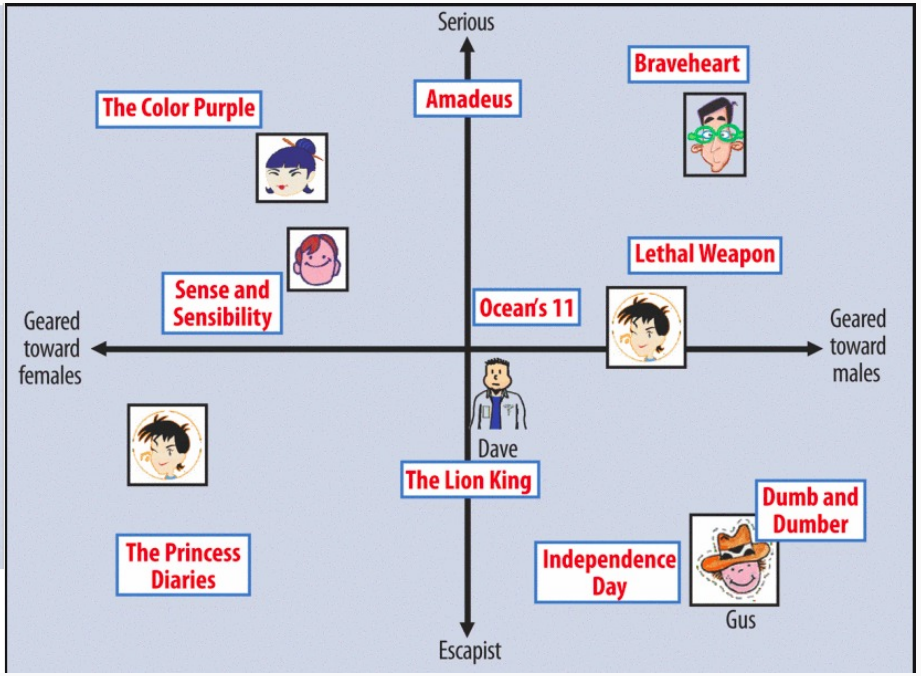
Mathematics and Computing, IIT-Hyderabad

Recommender Systems Strategies

Recommender Systems Strategies

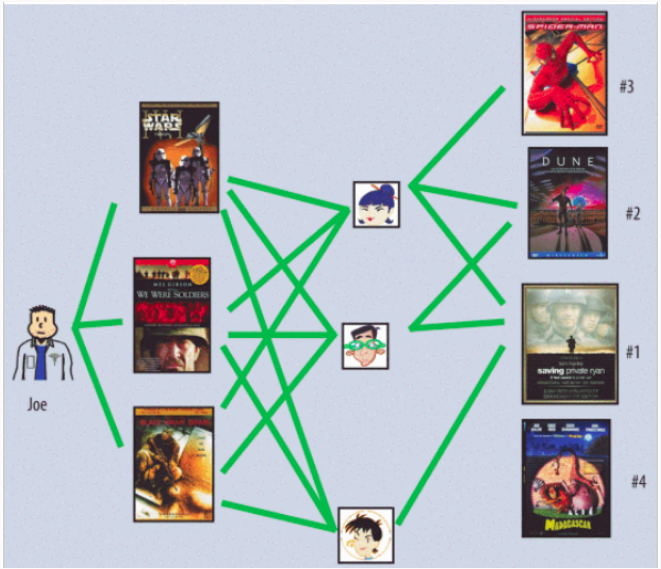






Neighbourhood Method

This method involves finding *K*-nearest neighbours (**K-NN** algorithm and it's variants).

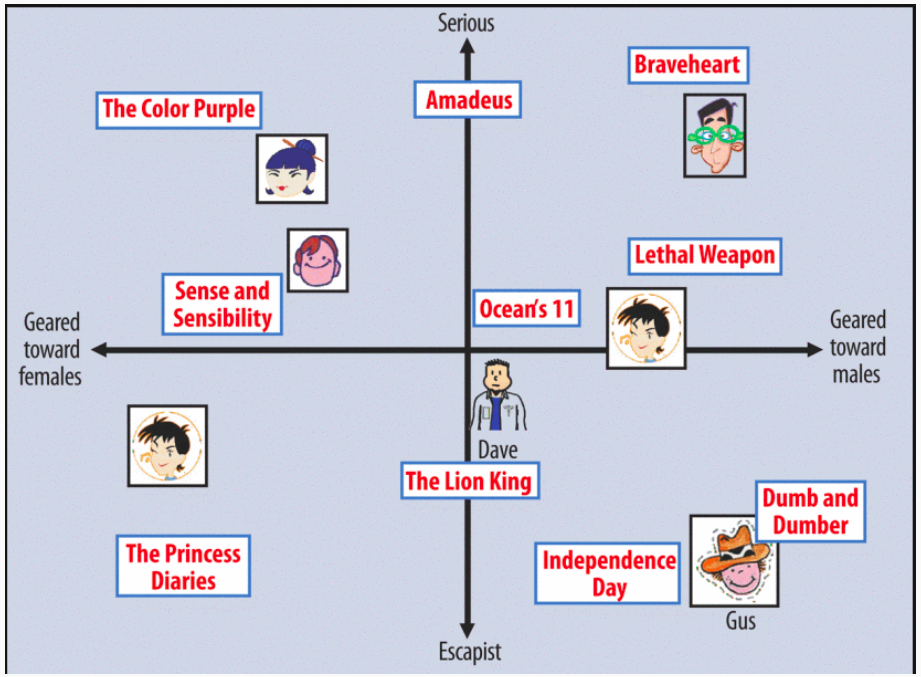


Why is this method very slow and also not accurate always?

- If predicting among all possibilities, requires **O(n)** iterations for each prediction.
- Only predicts within predetermined cluster if time reduced to **O(1)**.
- Cannot Incorporate Item/User based Bias

Latent Factor Models

Latent factor models try to explain the ratings by characterizing both items and users on factors inferred from the ratings patterns.



User's predicted relative rating for a movie =
Dot product of the movie's and user's location vectors on Latent Space.

Input Data

1. *Explicit Feedback*

    - Explicit input by users regarding their interest in products.
    - Comprises a **Sparse Matrix**, since any single user is likely to have rated only a small percentage of possible i
    - **High confidence** on this data.

2. *Implicit Feedback*

    - Observing user behavior, including purchase history, browsing history, search patterns etc.
    - Denotes the presence or absence of an event, so it is typically represented by a **Dense Matrix**
    - **Low confidence** on this data.

Matrix Factorization Model

Matrix factorization models map both users and items to a joint **Latent Factor Space** of dimensionality f.

Each item i is associated with a vector $q_i \in \mathbb{R}^f$, quantizing the amount of each attribute present in item i.
Each user u is associated with a vector $p_u \in \mathbb{R}^f$, quantizing the weightage of each attribute in the user's final decision.

The resulting dot product, $q_i^T p_u$, captures the user u's overall interest in the item i.

This approximates user u's estimated rating of item i, denoted by $\hat{r}_{ui}$:

$$\hat{r}_{ui} = q_i^T p_u. \tag{1}$$

Example

For 5 movies, 7 latent attributes, we get :

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} & q_{15} & q_{16} & q_{17} \\ q_{21} & q_{22} & q_{23} & q_{24} & q_{25} & q_{26} & q_{27} \\ q_{31} & q_{32} & q_{33} & q_{34} & q_{35} & q_{36} & q_{37} \\ q_{41} & q_{42} & q_{43} & q_{44} & q_{45} & q_{46} & q_{47} \\ q_{51} & q_{52} & q_{53} & q_{54} & q_{55} & q_{56} & q_{57} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{bmatrix} = \begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \\ \hat{r}_3 \\ \hat{r}_4 \\ \hat{r}_5 \end{bmatrix}$$

Optimization Problem

Introduce Regularization Parameter to avoid overfitting. To learn the factor vectors p_u and q_i, the system minimizes the regularized squared error on the set of known ratings:

$$\min_{q^\star, p^\star} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

The constant λ controls the extent of regularization, by keeping each attribute close to zero. λ is determined by cross-validation.

Learning Algorithm : SGD

One option is to use Stochastic Gradient Descent Algorithm, i.e.,

$e_{ui} = r_{ui} - {q_i}^T p_u$

$q_i \longleftarrow q_i + \gamma(e_{ui} p_u - \lambda q_i)$

$p_u \longleftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u)$

Problems :

- Requires **O(n)** operations for each iteration.
  . Feasible only for **Sparse Matrix**.
  . $\implies$ Cannot Use **Implicit Feedback** Data.
- All operations must be performed in serial order.

Learning Algorithm : ALS

Alternating Least Squares:

As both p_u and q_i are unknown, the objective is not Convex.

$$\sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{2}$$

If we fix one of the unknowns, the optimization problem becomes Quadratic Convex (QCP) and can be solved optimally.

ALS technique rotates between fixing q_i's and p_u's.
When all p_u's are fixed, the system recomputes the q_i's by Directly solving a least-squares problem, and vice-versa.
Each step decreases objective function until convergence.

Learning Algorithm : ALS

Repeat Until Convergence:

$$\text{(i)} \min_{q^\star} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

$$\text{(ii)} \min_{p^\star} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

Advantages :

- Feasible for **Dense Matrix**.
. $\implies$ Can Use **Implicit Feedback** Data.
- All p_i are computed independent of other factors (same for all q_i).
. $\implies$ Parallelization can be done here.

Adding Biases and Confidence

Incorporate **Bias** in this model -

(i) μ : Shifts the Prediction Mean from 0 to μ
. where μ= Overall Average Rating
(ii) b_i : Item Based Bias
. where b_i= Average Rating of Item i - Overall Average Rating
(iii) b_u : User Based Bias
. where b_u= Average Rating by User u - Overall Average Rating

Incorporate **Confidence** in this model -

(iv) c_{ui} : Confidence in observing r_{ui}

Final Recommender

Final Prediction is :
$\hat{r}_{ui} = c_{ui}(\mu + b_u + b_i + {p_u}^T q_i)$

Final form of Recommender:

$$\min_{q^\star, p^\star, b^\star} \sum_{(u,i)} c_{ui}(r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + \|b_u\|^2 + \|b_i\|^2)$$

subject to : $c_{ui} \geqslant 0 \ \forall (u, i)$
. $\lambda \geqslant 0$