# Recommender Systems Lecture —5

## Netflix Prize Solution

The Netflix Prize was a landmark competition in the field of machine learning and recommender systems. The challenge was to improve Netflix's movie recommendation engine's accuracy by 10%, measured by Root Mean Square Error (RMSE). The competition spurred numerous innovative solutions, culminating in a winning solution that significantly advanced the field.

**Final Solution Overview:**

- **Complex Ensemble:** The winning solution was an ensemble of 3040 different models, incorporating techniques like bagging and boosting.
- **Research Outcomes:** Despite its complexity and the difficulty in productionising, the competition yielded substantial research insights, particularly promoting the use of Matrix Factorization (MF) in recommender systems.

**Key Components of the Winning Solution:**

1. **Matrix Factorization:** The core idea was to represent both users and movies in a shared latent factor space, facilitating the prediction of unseen ratings.

    - $(R_{ui} \approx p_u \cdot q_i)$
    - $(p_u)$ : d-dimensional user vector.
    - $(q_i)$: d-dimensional movie vector.
    - $(R_{ui})$: Rating of movie $(i)$ by user $(u)$.

2. **Regularization:** To prevent overfitting, a regularization term was added, making the model more generalizable to unseen data.

$$(\min_{p,q} \sum (R_{ui} p_u \cdot q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2))$$

3. **Biases Inclusion:** Incorporating user and item biases to account for varying rating levels across the user base and movies, improving the accuracy of predictions.

$$(R_{ui} = \mu + b_u + b_i + p_u \cdot q_i)$$

$(\mu)$: Global average rating.
$(b_u), (b_i)$: User and item biases.

4. **Temporal Dynamics:** Recognizing that user preferences and item popularity can change over time, the solution incorporated time-dependent models to capture these dynamics.

**Temporal Dynamics Modeling:**

- The innovative approach of modeling ratings as a function of time significantly improved recommendation quality by acknowledging the temporal shifts in user preferences and item relevance.

**Training Methodology:**

- **Time-Segmented Training:** Data was segmented based on time intervals, acknowledging that preferences evolve, which allowed for more nuanced modeling of user behavior and item popularity over time.

**Performance Visualization:**

- The visualized RMSE vs. the number of parameters highlighted the effectiveness of incorporating biases and temporal dynamics, showcasing significant improvements over simpler models.

## Build a Time-Sensitive Recommendation System at YouTube

Building a time-sensitive recommendation system at YouTube requires accounting for the dynamic nature of user preferences and content relevance. Here's an approach to tackle this challenge:

**Idea 1: Periodic Parameter Updates**

- **Periodic Updates:** Regularly update the model parameters $((p_u) and (q_i))$ to reflect the most current user behaviors and content trends.
- **Advantages:** Keeps the recommendation system adaptive to changing trends and user interests.
- **Implementation:** Automate the retraining process to occur at set intervals (e.g., nightly or weekly), ensuring the system remains current.

**Idea 2: Weighting Recent Data More Heavily**

- **Weighted Formulation:** Assign higher weights to more recent interactions in the model to emphasize current user preferences.

$$(R_{ui} = w_{ui} \cdot (p_u \cdot q_i + b_i))$$

Where $(w_{ui})$ represents the weight assigned to the rating $(R_{ui})$, prioritizing recent activity.

**Operationalizing the Ideas:**

1. **Recency-Based Similarity Recommendations:**
    - **K-Nearest Neighbours (KNN):** For a video being watched $((v_i))$, recommend videos that are similar based on recent views, leveraging KNN for fast, real-time recommendations.

2. **Historical Taste-Based Recommendations:**
    - **Matrix Factorization (MF):** Perform MF regularly (e.g., nightly) to compute user and item vectors $((p_u)$ and $(q_i))$ and biases $((b_i))$, focusing on recent data to capture evolving tastes.
    - **Weight Matrix** $((W))$**:** Construct $(W)$ to give more importance to recent interactions, ensuring that the recommendations reflect the latest trends and user interests.

**Building the System:**

- **Batch Processing:** Utilize overnight batch processes to update the recommendation model, incorporating the latest user interactions and content uploads.
- **Top Recommendations:** From the nightly MF results, select the top 100 videos that align with a user's recent viewing history for recommendations.
- **Balanced Approach:** Combine recency-based KNN recommendations with MF-derived suggestions to offer a mix of fresh and historically aligned content.

## MF: Feature Engineering for Netflix Use Case

**Overview**

Matrix Factorization (MF) not only aids in recommendations but also serves as an implicit form of feature engineering, creating sensible d-dimensional vectors for users $((p_u))$ and movies $((q_i))$. These vectors, derived algorithmically, offer a rich, albeit non-interpretable, representation of users and movies, which can be utilized across various machine learning applications, including classification, regression, and clustering.

**Feature Engineering through MF**

- **Implicit Process:** MF generates $(p_u)$ and $(q_i)$ vectors, encapsulating user and movie features without explicit feature selection or creation.
- **Similarity Representation:** Users with similar preferences have similar $(p_u)$ vectors, and movies of similar genres or appeal have similar $(q_i)$ vectors.
- **Non-Interpretable Features:** Unlike traditional features (e.g., country, pin code), $(p_u)$ and $(q_i)$ lack direct interpretability but encapsulate user and item characteristics.

**Application Examples**

- **E-Commerce:** Recommender systems in ecommerce can utilize purchase data to create $(p_u)$ and $(q_i)$ vectors, stored in a feature store for immediate retrieval and application in various tasks like improving search results.
- **Feature Store:** A repository for key-value data pairs, allowing quick access to user and item vectors based on IDs, facilitating applications beyond recommendations.
- **Improving Search Results:** The search team can leverage item vectors to enhance search relevance, using techniques like K Nearest Neighbour to find and suggest similar items.

**Beyond Recommendations**

- **Ad Probability Models:** In scenarios requiring ad display decisions,$(p_u)$ and $(q_i)$ vectors can provide valuable user and item features to model the likelihood of ad clicks, demonstrating the versatility of MF-derived features across different contexts.

## MF Feature Engineering for Text Data

**Overview**

Matrix Factorization (MF) applied to text data, such as a Wikipedia dataset, aims to derive d-dimensional representations for words. This ensures these vectors group similar words, offering a non-interpretable yet sensible feature set.

**Representing Text Data**

- **Matrix $(A)$:** Documents as rows, words as columns $((n \times m))$, where $(A_{ij})$ denotes the frequency of word $(j)$ in document $(i)$.
- **Goal:** Decompose $(A)$ to obtain d-dimensional representations for words $((q_j))$ and documents $((p_i))$.

**Applications**

- **Document Similarity:** Utilizing document vectors to identify similar documents, beneficial for content platforms like Kindle or Google News.
- **Word Similarity:** Finding similar words based on their vectors, useful in search engines and grammar tools.

**Co-occurrence Matrix Approach**

- **Co-occurrence Matrix $(X)$:** An $(m \times m)$ matrix where $(X_{ij})$ represents the number of times word $(i)$ occurs in the context of word $(j)$.
- **Neighborhood Definition:** Words are considered in context if they appear within a predefined proximity, affecting the overall meaning when combined.
- **Decomposition:** Decomposing $(X)$ through MF yields two matrices $((B)$ and $(C))$, each offering a d-dimensional representation of words.

**Truncated SVD**

- **Top $(k)$ Singular Values**: Focusing on the top $(k)$ singular values in $(S)$ to reduce computational complexity without significant data loss.
- **Dimension Adjustment:** Adjusting dimensions of $(U)$, $(S)$, and $(V)$ to accommodate the top $(k)$ singular values, leading to the truncated SVD formulation $(X \approx U_k S_k V_k^\top)$.
- **Word Vector Extraction:** The word vector for word $(j)$ is derived from the $(j)th$ row of $(U_k)$.

**Considerations**

- **Computationally Intensive:** Even with Truncated SVD, the computation can be extensive due to the large vocabulary size. A practical workaround is to focus on the top 'n' words using IDF scores from TF-IDF.

## MF Feature Engineering for Images: Eigen Faces

**Overview**

Utilizing Matrix Factorization (MF) for face recognition involves transforming images into a compact, d-dimensional representation, historically known as Eigen Faces, a precursor to modern Convolutional Neural Networks (CNNs).

**Image Representation**

- **Format:** Images are represented as 2D matrices $((n \times n))$, flattened into vectors of dimension $(d = n^2)$ using row-major order.
- **Data Matrix $(X)$:** Composed of $(m)$ images, each represented as a row, leading to a matrix of dimension $(m \times d)$, where $(d)$ is the number of pixels.

**Applying PCA for Eigen Faces**

- **Covariance Matrix:** PCA is applied to the covariance matrix $(S)$ derived from $(X)$, as PCA requires a square matrix.
- **Dimensionality Reduction**: The dimensionality is reduced from $(d)$ to $(k)$ by selecting the top $(k)$ eigenvectors and eigenvalues, resulting in a reduced matrix $(V_k)$ of dimensions $(d \times k)$.

**Generating Eigen Faces**

- **k-Dimensional Representation:** The k-dimensional representation of images is obtained by multiplying $(X)$ with $(V_k)$, where $(XV_k)$ yields a k-dimensional vector for each image.

- **Visualization:** The k-dimensional vectors can be reshaped back into $(n \times n)$ format to visualize the Eigen Faces, which capture the most significant variations across the set of images.

**Applications and Limitations**

- **Face Recognition:** The Eigen Faces technique was an early method for face recognition, leveraging the principal components of image data to identify and distinguish faces.
- **Limitations:** Despite its innovativeness, Eigen Faces has been largely superseded by more advanced techniques like CNNs, which offer superior performance in capturing and recognizing facial features due to their ability to learn hierarchical representations.