Recommendation System

(k-NN Based Collaborative Filtering)

CED18I042
Reuben Skariah Mathew

Introduction

A **recommendation system**, or recommender system, is a subclass of information filtering system that seeks to **predict the "rating" or "preference" a user would give to an item**.

Recommender systems are used in a variety of areas, with commonly recognised examples taking the form of **playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms** and open web content recommenders.

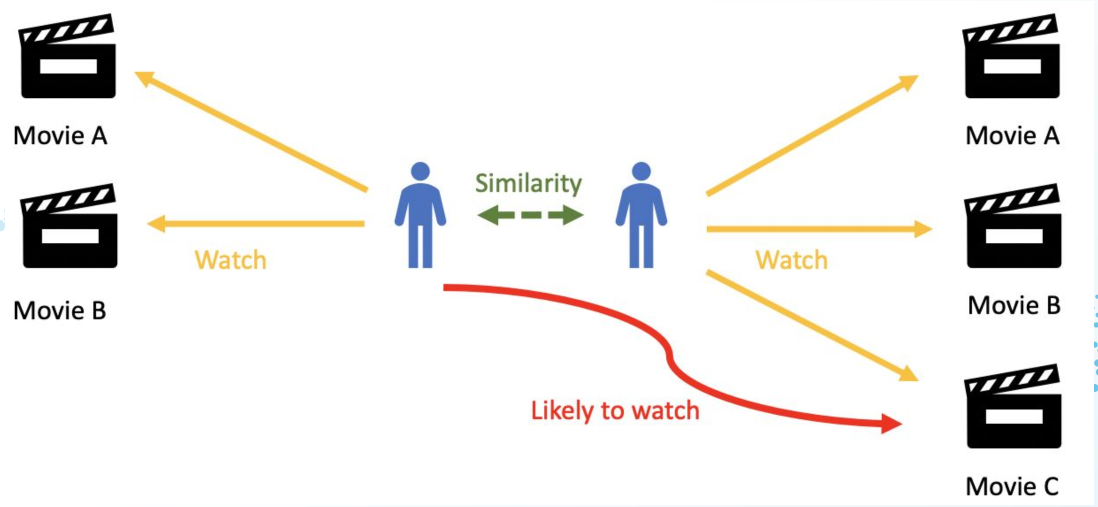
Most internet products we use today are powered by recommender systems. Youtube, Netflix, Amazon, Pinterest, and a long list of other internet products all rely on recommender systems to **filter millions of contents and make personalized recommendations to their users**.

These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries.

Collaborative Filtering

Collaborative filtering approach builds a model from a **user's past behaviors** (items previously purchased or selected and/or numerical ratings given to those items) as well as **similar decisions made by other users**. This model is then used to predict items (or ratings for items) that the user may have an interest in.

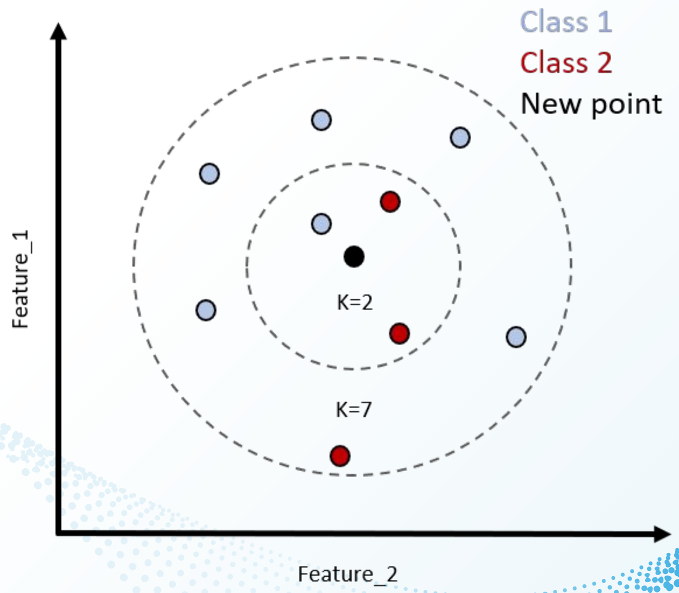
Collaborative filtering systems use the actions of users to recommend other movies. In general, they can either be user-based or item-based. **Item based** approach is usually preferred over user-based approach. To implement an item based collaborative filtering, **KNN** is a perfect go-to model and also a very good baseline for recommender system development.



k-Nearest Neighbours (KNN)

The **k-nearest neighbors (k-NN)** algorithm is a **supervised machine learning algorithm** that can be used to solve both classification and regression problems. This algorithm is **non-parametric** in nature, hence it does not make any underlying assumptions about the distribution of data.

The k-NN algorithm is considered as one of the simplest machine learning algorithms. However, it is **computationally expensive** especially when the size of the training set becomes large which would cause the classification task to become very slow.



Advantages

1. No Training Period: KNN is called Lazy Learner (Instance based learning). It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g. SVM, Linear Regression etc.

2. Since the KNN algorithm requires no training before making predictions, **new data can be added seamlessly** which will not impact the accuracy of the algorithm.

3. KNN is easy to implement: There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Disadvantages

1. Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.

2. Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.

3. Need feature scaling: We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.

Working

K-nearest neighbors (k-NN) algorithm uses **‘feature similarity’** to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

Firstly datasets are loaded. This will contain the list of movies (movie ID), list of users (user ID) and their ratings for particular movies.

For an input movie (for which recommendations will be provided):

1. The data is converted to a matrix (users and their ratings for each movie) that can be given as input to the algorithm.
2. A distance value (Cosine Similarity) between each matrix entry to every other item is calculated and stored in a prediction matrix.
3. The k matrix entries for a particular user (row), with highest similarity will be given as recommendations.

Code Balance

The number of floating point operations (FLOPs) can be found through line based profiling. In the output file we can see the number of times each line is executed/called. By adding the number of times lines with FLOPs is called we can estimate the number of FLOPs as well as number of words.A large number of FLOPs are found in the functions such as: norm, dotProduct, adjCosineSimilarity and colabFilter.

Total Flops: 2843799997 = 2.843799997 gigaflops
Total Words: 4539698421
*Calculation can be found in the report.

Code Balance = Words / FLOPs
4539698421 / 2843799997

Code Balance = 1.59634940073

Recommendation System
(k-NN Based Collaborative Filtering)

HPC Report

Roll No: CED18I042
Name: Reuben Skariah Mathew

Date: 8th September, 2021

Introduction

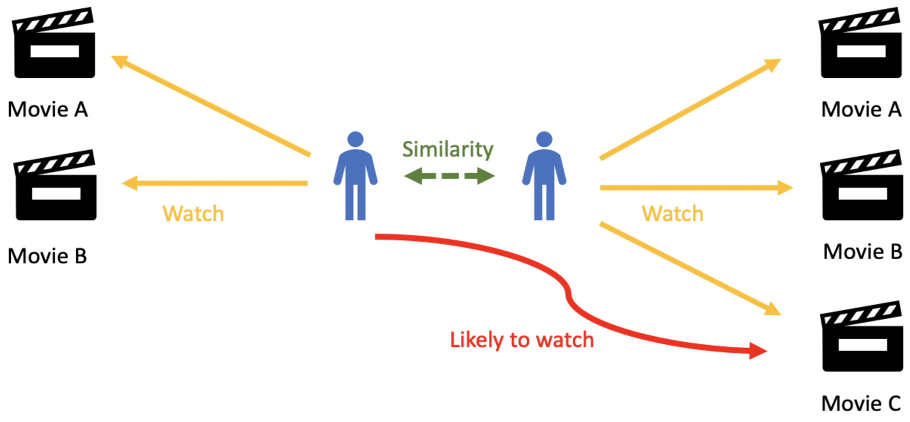
A **recommendation system**, or recommender system, is a subclass of information filtering system that seeks to **predict the "rating" or "preference" a user would give to an item**.

Recommender systems are used in a variety of areas, with commonly recognised examples taking the form of **playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms** and open web content recommenders.

Most internet products we use today are powered by recommender systems. Youtube, Netflix, Amazon, Pinterest, and a long list of other internet products all rely on recommender systems to **filter millions of contents and make personalized recommendations to their users**.

These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries.

Collaborative Filtering



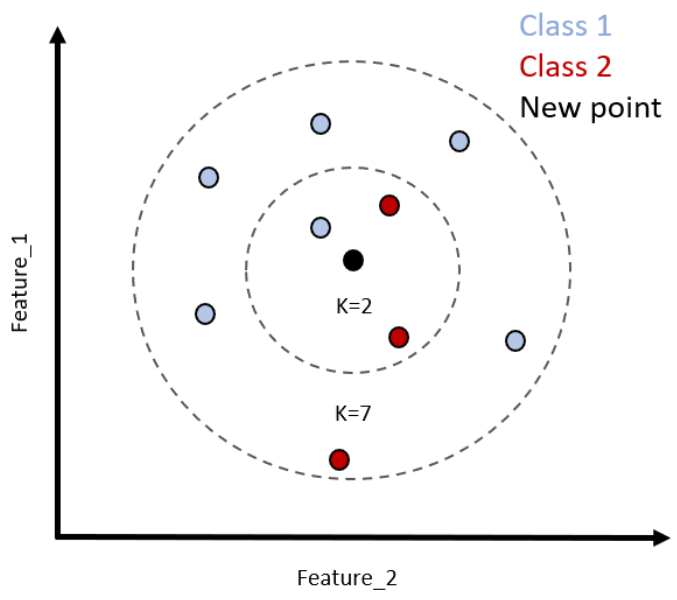
Collaborative filtering approach builds a model from a **user's past behaviors** (items previously purchased or selected and/or numerical ratings given to those items) as well as **similar decisions made by other users**. This model is then used to predict items (or ratings for items) that the user may have an interest in.

Collaborative filtering systems use the actions of users to recommend other movies. In general, they can either be user-based or item-based. **Item based** approach is usually preferred over user-based approach. To implement an item based collaborative filtering, **KNN** is a perfect go-to model and also a very good baseline for recommender system development.

k-Nearest Neighbours (KNN)

The **k-nearest neighbors (k-NN)** algorithm is a **supervised machine learning algorithm** that can be used to solve both classification and regression problems. This algorithm is **non-parametric** in nature, hence it does not make any underlying assumptions about the distribution of data.

The k-NN algorithm is considered as one of the simplest machine learning algorithms. However, it is **computationally expensive** especially when the size of the training set becomes large which would cause the classification task to become very slow.



Working

K-nearest neighbors (k-NN) algorithm uses '**feature similarity**' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

Firstly, datasets are loaded. This will contain the list of movies (movie ID), list of users (user ID) and their ratings for particular movies.

For an input movie (for which recommendations will be provided):

1. The data is converted to a matrix (users and their ratings for each movie) that can be given as input to the algorithm.
2. A distance value (Cosine Similarity) between each matrix entry to every other item is calculated and stored in a prediction matrix.
3. The k matrix entries for a particular user (row), with highest similarity will be given as recommendations.

Code Balance

The number of floating point operations (FLOPs) can be found through line based profiling. In the output file we can see the number of times each line is executed/called. By adding the number of times lines with FLOPs is called we can estimate the number of FLOPs as well as number of words.

A large number of FLOPs are found in the functions such as: norm, dotProduct, adjCosineSimilarity and colabFilter.

FLOPs Calculation:

norm() - (529112946 * 2 = 1058225892) + (11993022 * 6 = 71958132)

dotProduct() - (270552984 * 2 = 541105968) + (11993022 * 5 = 59965110)

adjCosineSimilarity() - (264556473 * 4 = 1058225892) + (5996511 * 8 = 47972088) = 1106197980

colabFilter() - (5996511 * 1) + (87601 * 4 = 350404) = 6346915

Total Flops: 2843799997 = 2.843799997 gigaflops

Word Count Calculation:

norm() - 529112946 * 2 = 1058225892

dotProduct() - 270552984 * 3 = 811658952

adjCosineSimilarity() - (264556473 * 10 = 2645564730) + (5996511 * 2 = 11993022) = 2657557752

colabFilter() - (5996511 * 2) + (87601 * 3) = 12255825

Total Words: 4539698421

Code Balance = Words / FLOPs

4539698421 / 2843799997

Code Balance = 1.59634940073