1 Introduction to Data Mining

If we look at the change in processing speed and the speed of data collection, we can deduce that every 3 years the processing speed increases by a factor 4 and the collected data by a factor 8.

Definition 1.1 (Moore's Law)**.** Processing speed doubles every 18 months

Definition 1.2 (Kryder's Law)**.** Hard disk capacity doubles every 12 months

Definition 1.3 (Lyman & Varian)**.** The amount of collected data doubles every year

Meaning, every 3 years, the overflow of data (or flood) is doubling. Besides the total amount of data, the shape of the data has also changed over time. This new data is also called Big Data, generally characterized by the "Three Vs of Big Data" (IBM):

Definition 1.4 (Three Vs of Big Data (IBM))**.**

- **Volume** - Large size of data (terabytes, petabytes).
- **Velocity** - Speed of added data is fast, and needs to be processed fast.
- **Variety** - Structured and unstructured data, such as text, sensor data, audio, video, click streams, log files and more.

To be able to process such big data, we need to do fuzzy queries and we want to be able to run the processes in a streaming fashion. Traditionally these are easy to execute, but complex in this context. The trade-off therein is accuracy for speed (a rough answer in real-time is sufficient).

1.1 Examples

1.1.1 Recommender Systems

Given petabytes of sales data in the form *<user_id, item_id>* we want to do recommendations to the *current_user_id* that just clicked or bought *current_item_id* and we have only milliseconds to do this.

This is a fairly simplified scenario, as you generally have a lot more information on users to do these recommendations.

1.1.2 Data Stream Mining

In some situations, the incoming stream of data is too large or too fast to be stored, and needs to be analyzed on-the-fly with limited memory.

Examples of this are to detect fraud in electronic transactions, showing the correct advertisements, prognostic health monitoring of a fighter jet, detection of DoS attacks, or returning rankings from current interests or click streams from users.

1.2 Data Mining Paradigms

We can group data mining activities into two separate paradigms: supervised and unsupervised learning.

Definition 1.5 (Supervised Learning)**.** We have a set of defined inputs and defined outputs and try to build a model that converts input into these outputs

Definition 1.6 (Unsupervised Learning)**.** The outputs are characteristics and generally tries to learn similarity from the data.

1.3 Overfitting

If we would train our model on all the data that we have, we have the risk to overfit on the data. This means that it follows the data closely, but will not work with unseen data. Thus, we separate the training data from test and evaluation data, and use approaches to avoid overfitting on the training data.

For example, we can use k-Fold Cross-Validation where we split the training data (stratified) in to k folds and use $(k-1)$ folds for training and 1 for testing. Repeat this k times and average the results. A downside of this is that it is computationally expensive (but parallelizable).

2 Recommender Systems

There are different approaches to recommender system, the idea is basically that you have a mapping of users onto items and use this mapping to learn the value of an item to a user that is not in the mapping.

Generally, we have a matrix of the size users $\times$ items and it is generally filled with ratings, interest, history or some other value that represents the relation. Such matrices are very large and usually sparse.

2.1 Naive Approach

The baseline approach is to take the mean of the entire matrix, item or user. A bit more advanced we can do a linear regression modeled as:

Definition 2.1 (Naive Linear Recommender)**.**

$$R_{\text{user-item}}(u, i) = \alpha \times R_{\text{user}}(u, i) + \beta \times R_{\text{item}}(u, i)$$

The linear regression returns good results, is easy to calculate and update, and allows for simple interpretation of the model (good movie, harsh user, crowd follower).

2.2 Content-Based Approach

The intuition for Content-based Approaches is to construct a profile for every item and a profile of every user, and see which items are closest to the user profile. The profile generally contains labels such as budget, genre, origin, cast, et cetera. The baseline would be to take the mean of each dimension of the profile for a user.

2.2.1 Model-Based Approach

The content-based approach assumes that you can fill in these parameters for users yourself (or source it from the user themselves). However, we can also train a model per user that predicts rating from the item profiles. Pitfalls here are that it is expensive to build and maintain, the accuracy is low and it doesn't work with new users.

2.3 Collaborative Filtering

An alternative to item-specific profiles is to recommend items to users based on what similar users have liked. You can either do user-to-user collaborative filtering or item-to-item collaborative filtering.

User-User Recommendations Each user has a vector with ratings for every item that they have rated. Users with similar vectors are selected as neighbours. We then take the top L neighbours and aggregate their ratings to predict the rating.

Item-Item Recommendations Each item has a vector with ratings for every user that have rated it. Items with similar users are selected as neighbours. We then take the top L neighbours and aggregate their ratings to predict the rating.

The main differences here are that for either category insertion of new items or users creates a cold start problem respective of the type of collaborative filtering. Furthermore, user-to-user filtering has a higher personalization, suffers more from sparsity, is less scalable with larger user bases and may fluctuate with user behaviour. However, as it takes the user as the comparison base, the personalization is better.

2.4 Singular Value Decomposition

A large motivator for changes in recommendation algorithms was the Netflix Challenge. Before, as discussed in the introduction we can do recommendation by matching pairs of users and ratings to estimate ratings for new users (based on other ratings).

A possible approach to recommending movies by using the user ratings is the CINEMATCH system, which was state-of-the-art in 2006. While this approach worked well, Netflix decided to set up a challenge for scientists to improve the recommendation system. For this challenge you would predict ratings based on a training set and submit them. The improvement was measured using the RMSE.

Definition 2.2 (RMSE).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{predicted} - \text{true})^2}$$

2.5 Matrix Factorisation - UV Decomposition

One approach for recommendation is the use of matrix factorisation, where the users and ratings are combined into a utility matrix. This matrix is then split into a user matrix (with a fixed dimension) and a vector matrix (with a fixed dimension). Ratings can then be approximated by multiplying a specific user vector with the transpose of an item vector and summing the output. The dimension lengths are fixed before doing the decomposition but their semantic "meaning" and values are learned from the data.

If we encode a loss function to be a polynomial that measures the distance between the predicted values and the true values (like the MSE) we can do learn the correct values for U and V (where the MSE is minimal).

For example, we can do a simple line search, this initializes all variables randomly, chooses a random parameter (with the rest frozen) and then finds the optimal value for that parameter. It does that by calculating the derivative of that parameter and chooses the optimum.

The gradient descent algorithm is a better alternative, which takes the loss function and chooses an arbitrary point in the multi-dimensional space. Then, it finds a direction in which the loss function is decreasing the most rapidly using partial derivatives and makes a small step in that direction. It repeats this until a local minimum is reached.

Finally, an alternative is to use alternating least squares, which reduces the distance in an alternating fashion between the user and item matrices. First, freeze the user features and treat all item features as variables, solve the resulting least squares problem. Then, freeze the item features and treat the user features as variables and solve resulting least squares.

2.6 Boosting Accuracy - Blending Models

Better results were ultimately achieved by blending different algorithms together. For example, first you can train the initial values using a regression model instead of choosing random parameters.