# Recommendation systems

Tool: Clustering, collaborative and content filtering.

The Analytics Edge: Recommendation systems build models around user preferences to help personalize the user experience. Examples include Netflix, which provides recommendation on movies to users, Amazon, which provides recommendations on items of almost any type, and eHarmony, which provides recommendations on compatible matches for single men and women. The large number of choices available and the massive amount of data available through online transactions help companies use tools of analytics—such as clustering and collaborative and context filtering—to find the right items to recommend to users. The challenge in recommendation systems pertains to obtain real-time recommendations, make good recommendations (poor ones can make customers unhappy), and deal with users on which very little information is available.

## 1 The Netflix story

Netflix is a company that provides DVD by mail service and on-demand streaming of films and TV series. As of October 2018, Netflix has more than 130 million subscribers globally (about 190 countries), with an annual revenue of roughly $ 11 billion (in 2017). Netflix is available in 190 countries, including Singapore. The company started when Reed Hastings (one of the co-founders) was forced to pay $ 46 in fines for returning the Apollo 14 DVD well past its due date. For this reason, Reed Hastings and Marc Randolph (the other co-founder) created a new business model based on the monthly subscription concept still in used today: for a flat monthly fee, a subscriber rents DVDs without due dates, late fees, and shipping/handling fees. To rent a new film, the subscriber must mail back the current DVD, upon the receipt of which Netflix will ship the next DVD in the rental queue. A similar business model is applied to the on-demand streaming.

### 1.1 Netflix Prize

Netflix introduced its recommendation system—called *Cinematch*—in 2000. According to Netflix, Cinematch was accurate within half a star 75 % of the time (Netflix users are asked to give a rating, from one to five stars, for any movie they watch). Yet, a more accurate recommendation system was an attractive option.

In October 2006, Neflix announced that it would award a 1-million dollar prize to the first developer of a recommendation system that could beat Cinematch at predicting customer ratings by more than 10 % on a test dataset. Netflix provided a training dataset of 100,480,507 ratings that 480,189 users gave to thousands of movies ($\approx$ 18,000 movies). In addition, Netflix withheld the ratings of 2,817,131 data points from the same subscribers over the same set of movies as a ”qualifying” dataset. Any participating team had to predict the ratings on the entire qualifying dataset but was informed on the accuracy only for roughly half of the data (a quiz set of 1,408,342 ratings). This was used to calculate the leaderboard. The other half (a test of 1,408,789) of ratings was used to determine the ultimate winner. Only the judges knew the quiz and test set partition in the qualifying dataset. In the training dataset, the average user rated over 200 movies and the average movie was rated by over 5,000 users. However, some movies had only 3 ratings in the training set, while one user had rated over 17,000 movies (wide variance in the dataset).

The predictions—which could be any number, not necessarily in the set $\{1, 2, 3, 4, 5\}$—were scored against the true ratings in terms of Root Mean Squared Error (RMSE, to be minimized). The trivial algorithm that used the average rating for each movie from the training set produced a RMSE of 1.0540 on the quiz set. Cinematch scored an RMSE of 0.9514 on the quiz data using the training set to build the model (roughly, a 10% improvement). On the test set, Cinematch had an RMSE of 0.9525 (close to the quiz set). To win the prize, a team had to beat Cinematch by another 10 %—that is, 0.8572 of the test set. Once the RMSE on the quiz set was 0.8572 or lesser, 30 days were provided to submit additional candidate predictions. As long as no team won the grand prize, $ 50,000 were awarded for each year they improved on the previous winner by at least 1%. Each team could submit one attempt per day.

The story of the Netflix Prize can be summarized as follows:

**Oct 2, 2006.** The competition was launched.

**Oct 8, 2006.** WXYZ Consulting beats Cinematch.

**June 2007.** Over 20,000 teams registered.

**Nov 13, 2007.** BellKor (a team of 3 AT&T lab researchers) won the $ 50,000 progress prize with 8.43% improvement over Cinematch.

**2008.** Team BellKor and BigChaos joined together to win another $ 50,000 prize with 9.4% improvement.

**June 26, 2009.** Team BellKor Pragmatic Chaos achieved 10.05% improvement (quiz test RMSE of 0.8558).

**Last call (30 days).** Netflix entered last call for the Grand Prize.

**July 25, 2009.** Team Ensemble achieved 10.09%.

**July 26, 2009.** Netflix stopped gathering submissions.

**Sep 18, 2009.** On the test set, both Team BellKor Pragmatic Chaos and Team Ensemble had a RMSE of 0.8657, but Team BellKor Pragmatic Chaos submitted 20 min earlier, making them the winners of the $ 1-million Netflix Prize.

The Team BellKor Pragmatic Chaos consisted of two researchers from AT&T Labs, two researchers form Austin at the Commando research & Consulting, one researcher from Yahoo, and two researchers from Pragmatic Theory. The team published a description of the algorithm, as required by the competition. A variety of method was used in the final winning predictive algorithm. Indeed, successful analytics often need a combination of a variety of approaches. These approaches included collaborative filtering, regression models, and LASSO—all combined into an ensemble.

## 1.2 MovieLens

MovieLens is a recommendation system setup by GroupLens, a research lab based at the University of Minnesota (`https://grouplens.org`). GroupLens collects data on movie ratings and makes them available for research.

## 1.3 Summary

Data: The data contain information on the ratings of movies provided by multiple customers.

Model: clustering can be used to categorize movies and customers, while collaborative and content filtering can be used to make recommendations on which movie to watch.

Value and Decision: an effective recommendation system provides an edge over competitors by increasing sales and improving the customer's satisfaction.

# 2 Clustering

In supervised learning, one has typically access to a set of $p$ features (or predictor variables) $\{x_1, \ldots, x_p\}$ and an output, or response variable, $y$. The goal is to predict $y$ from $x_1, \ldots, x_p$. We have seen different supervised learning algorithms, such as linear regression, logistic regression, discrete choice models, CARTs, and Random Forests. In unsupervised learning, one has only access to a set of $p$ features $\{x_1, \ldots, x_p\}$, and the goal is to identify patterns within the data. One of the main challenges with unsupervised learning is that it is far more subjective, since there is no clear way to perform cross-validation or validate the results on a test set. However, these techniques are very important in various domains, such as recommendation systems. Clustering, for example, can be used to identify groups of shoppers based on their browsing and purchasing histories on, say, Amazon; then, an individual can be preferentially shown items based on the purchase histories of other shoppers in the same cluster.

The goal of clustering is to organize observations into distinct groups such that:

1. Observations within a group are as similar as possible;
2. Observations in different groups are different from each other.

In the followings, we will see two well-established clustering techniques, namely K-means and Hierarchical clustering.

## 2.1 K-means clustering

Given $n$ observations $x_i = \{x_{i1}, \ldots, x_{ip}\}$ (with $i = 1, \ldots, n$), we want to create $K$ clusters $C_1, \ldots, C_K$ (with $K \leq p$), where $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$ and $C_k \cap C_{k'} = \{\}$ (for $k \neq k'$). This can be achieved by minimizing the within-cluster sum of squares:

$$\min_{\mu_1,\ldots,\mu_K} \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2,$$

where $||x_i - \mu_k|| = \sqrt{(x_{i1} - \mu_k)^2 + \ldots + (x_{ip} - \mu_k)^2}$ and $\mu_1, \ldots, \mu_K$ is the mean of the observations in clusters $C_1, \ldots, C_K$. By solving this problem, the observations are partitioned into clusters in which each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster. An illustration for $K = 3$ is given in Figure 1.

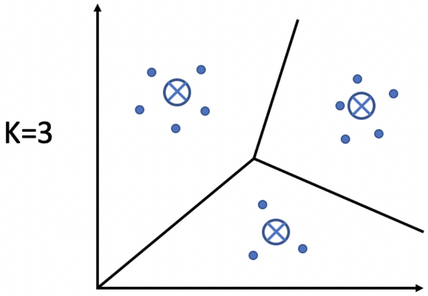

Figure 1: Partitioning of the data space into three clusters (regions where all the points in that cell are closer to the symbol $\otimes$ than any other).

Given $K$ clusters and $n$ observations, there are $K^n$ ways to partition the space. Solving the K-means problem exactly is computationally demanding, but there are efficient heuristics that quickly converge to the local optimum. One of these is Lloyd's algorithm. Given an initial (and random) set of $K$ means, the algorithm proceeds by alternating two steps:

1. Assignment step. Assign each observation to the cluster whose mean (or centroid) has the least squared Euclidean distance;
2. Update step. Calculate the new means of the observations in the new clusters.

The process continues until the assignment no longer changes. An example for $K = 3$ is given in Figure 2.

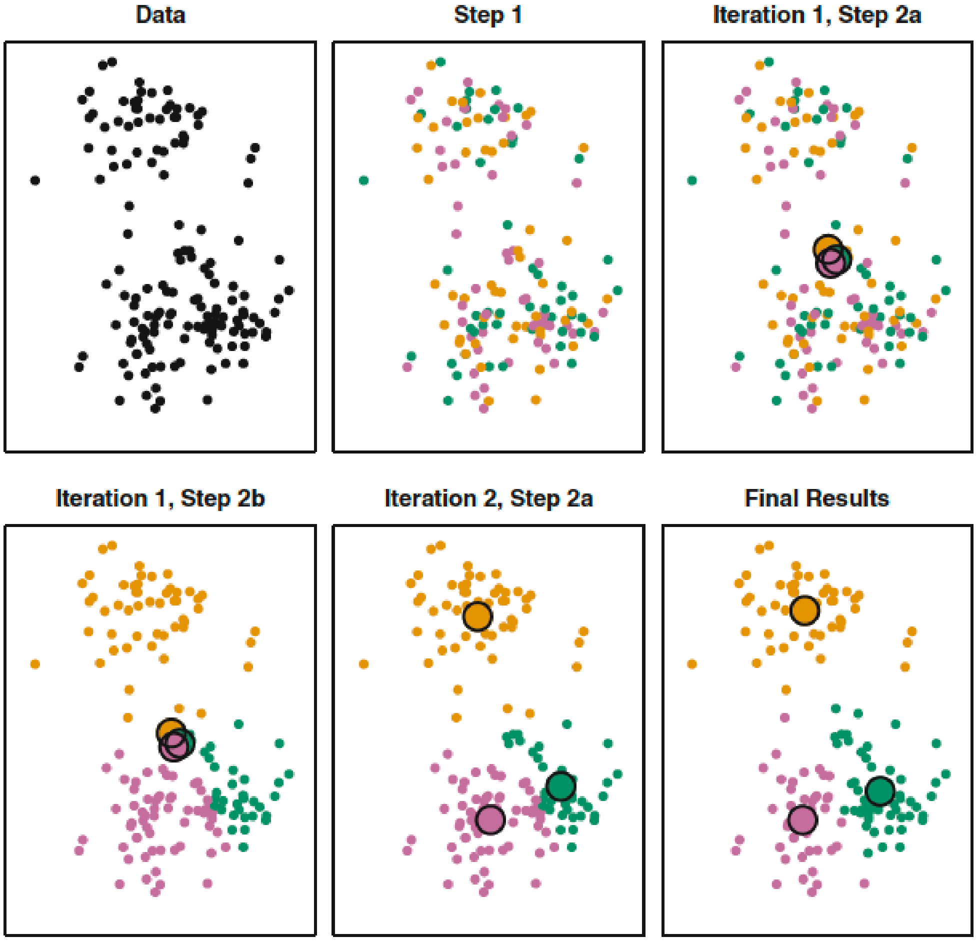

Figure 2: Illustration of the K-means algorithm for $K = 3$. Source: James et al. (2014)

The main advantage of the K-means algorithm is that it works well for both small and large datasets. However, the number of clusters ($K$) has to be decided before running the algorithm, so it is advisable to run the algorithm a few times with different choices of $K$ and to check if the clusters are reasonable—for example by looking at statistics of each cluster or looking at an outcome variable not used during the clustering process. Since randomization is involved, one can also try running the algorithm multiple times (for a fixed value of $K$) and then choose the best clusters.

## 2.2 Hierarchical clustering

This method does not require $K$ to be pre-specified; instead, it provides a tree-based representation (called *dendrogram*) of the observations (illustrated in Figure 3). Each leaf of the dendrogram corresponds to an observation, so the fusion of two leaves into a branch corresponds to similar observations. Observations that fuse at the bottom of the tree are very similar, while fuses at the top tend to be different. The height of fusion indicates how different the observations are. Note that similarity of observations should be based on the location on the vertical axis, where branches containing the observations are fused first. The similarity of two observations cannot be based on their proximity along horizontal axis. In Figure 3, for example, 9 is not closer to 2 anymore than it is to 8, 5, or 7. To identify clusters across a dendrogram, make a horizontal cut across the dendrogram: the number of vertical lines that it crosses determines the number of clusters. The furthest this horizontal line can move up and down without touching one of the horizontal lines the better it is. Also depending on the application, you might have some sense of how many clusters you want.

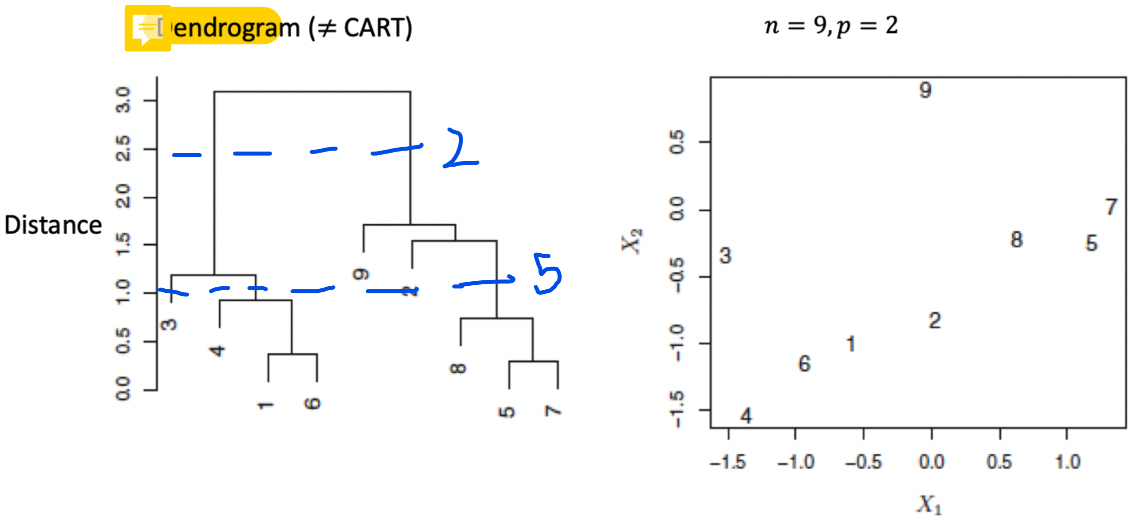

Figure 3: Graphical illustration of the Hierarchical clustering for a problem with 9 observations in a two-dimensional space. Source: James et al. (2014)

The term hierarchical refers to the observation that the clusters obtained by cutting the dendrogram at a particular height are nested within clusters obtained by cutting it at a greater height. The algorithm underpinning Hierarchical clustering can be summarized as follows:

1. Begin with $n$ observations and a measure of pairwise dissimilarities (a total of $\frac{n(n-1)}{2}$ measures). Treat each of the observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$, (a) examine all pairwise inter-cluster dissimilarities among the $i$ clusters and identify pairs of clusters that are most similar. Fuse these clusters. The dissimilarity between clusters helps determine the height of the dendrogram where fusion should be placed; and (b) compute the pairwise inter-cluster dissimilarity among the remaining $i - 1$ clusters.

Note that there are different measures for determining the distance (dissimilarity) between clusters. Suppose that the Euclidean distance is used: then, one can define the linkage (dissimilarity between two groups of observations) as complete (maximal distance between two points in clusters A and B) or average (average distance between any two observations, one in A and the other in B). Ward's criterion is another common measures (it minimizes the total within-cluster variance).

# 3 Recommendation systems

There are three main types of recommendation systems:

1. **Collaborative filtering.** Recommendations are based on attributes of users. Each user is represented by a vector of items where the $i$-th entry gives the customer's rating of the $i$-th item. This vector will typically have many empty entries (only a small fraction of the items is ranked or purchased).
2. **Content filtering.** Recommendations are made based on attributes of items. Each item is represented by a set of attributes (e.g., genre of movie, keywords, or webpage). For example, Pandora uses the attributes of a song (e.g., style and artist) to seed the station with other songs with similar attributes.
3. **Hybrid recommendation systems.** This is a combination of both collaborative and content filtering. Netflix, for example, makes recommendations by comparing the watching and searching habits of similar users (collaborative filtering) as well as recommending movies that share similar attributes (content filtering).

Content filtering often works better than collaborative filtering if the user has not rated or purchased many items. As soon as one item is liked, content filtering can be used to make recommendations. However, if a user has rated many items, it is hard for content filtering to make recommendations, since there might be many items with similar recommendations. Furthermore in collaborative filtering, it is possible to recommend items not previously considered in user's history. This is possible to get them purchase items that may otherwise not be considered.

## 3.1 Collaborative filtering

In this approach, the data of ratings of users for items are used to predict missing ratings or create top-$N$ recommendation lists for an active user. Note that collaborative filtering will suffer from a cold start problem in comparison with content filtering, since it will be unable to address new products and users. On the other hand, it is domain free (it does not depend on the item attributes). A graphical illustration of the collaborative filtering approach is given in Figure 4.

- Person A likes movies 1, 2 & 3
- Suppose persons B, C, D & E also likes movies 1, 2 & 3
- All of them (B, C, D & E ) also like movie 4, followed by movie 6.
- Thus we can recommend them to A in order.

### 3.1.1 Techniques for collaborative filtering

Let $r_{ui}$ be the rating of user $u$ for item $i$:

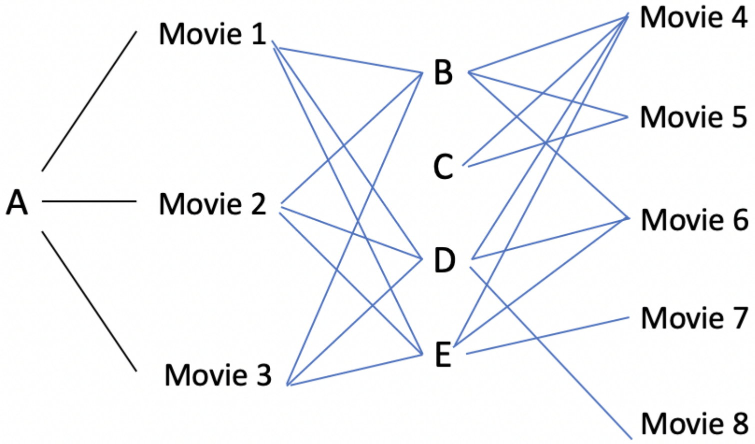

Figure 4: Graphical illustration of the collaborative filtering approach.

**Baseline model.** A simple baseline model is to predict the average rating based on the item average popularity:

$$b_{ui} = \bar{r}_i,$$

where $b_{ui}$ is the baseline prediction for user $u$ and item $i$, and $\bar{r}_i$ is the average rating for item $i$ across all users who rated it.

**User-based collaborative filtering.** This model is based on identifying other users (whose ratings are similar to those of the active user) and use their ratings on other items to predict what the active (current) user will like. To measure the similarity between users, we can use the Pearson correlation, i.e.,

$$S_{uv} = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_u \cap I_v} (r_{vi} - \bar{r}_v)^2}},$$

where $S_{uv}$ is the similarity between users $u$ and $v$, $I_u$ and $I_v$ the items rated by users $u$ and $v$, $\bar{r}_u$ and $\bar{r}_v$ the average ratings of users $u$ and $v$. This similarity metrics was adopted by Netflix. An alternative similarity measure is the cosine similarity:

$$S_{uv} = \frac{\sum_{i \in I_u \cap I_v} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{ui}^2} \sqrt{\sum_{i \in I_u \cap I_v} r_{vi}^2}}.$$

To predict the rating, the simplest method is to choose a set of neighbours of user $u$, denoted by $N_u$ (say the $k$ nearest neighbours with a certain level of similarity),

$$p_{ui} = \frac{\sum_{v \in N_u} r_{vi}}{|N_u|},$$

where $p_{ui}$ is the predicted rating for user $u$ and item $i$. We can also use the observation that some users are more similar to $u$ using the similarity metric:

$$p_{ui} = \frac{\sum_{v \in N_u} S_{uv} r_{vi}}{\sum_{v \in N_u} |S_{uv}|}.$$

While all users could be used in the set $N_u$, it helps strict it to a smaller number of neighbours (in the range of 20 to 200, typically).

The performance of recommendation systems can be improved by normalizing the ratings so as to compensate for user's differences on rating scales. For example, there could be bias caused by users who consistently rate higher than (or lower than) other users. Let $\hat{r}_{ui} = r_{ui} - \bar{r}_u$: this centers the scores. After applying collaborative filtering and normalizing back to the original scale, we get:

$$p_{ui} = \bar{r}_u + \frac{\sum_{v \in N_u} S_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in N_u} S_{uv}}.$$

Other transformations include taking the rating variance into account.

To measure the quality of predicted ratings, we can use:

$$\text{RMSE} = \sqrt{\sum_i \frac{(p_{ui} - r_{ui})^2}{n}},$$

where $p_{ui}$ is the predicted rating for user $u$ and item $i$, $r_{ui}$ the real rating, and $n$ the total number of ratings that are predicted.