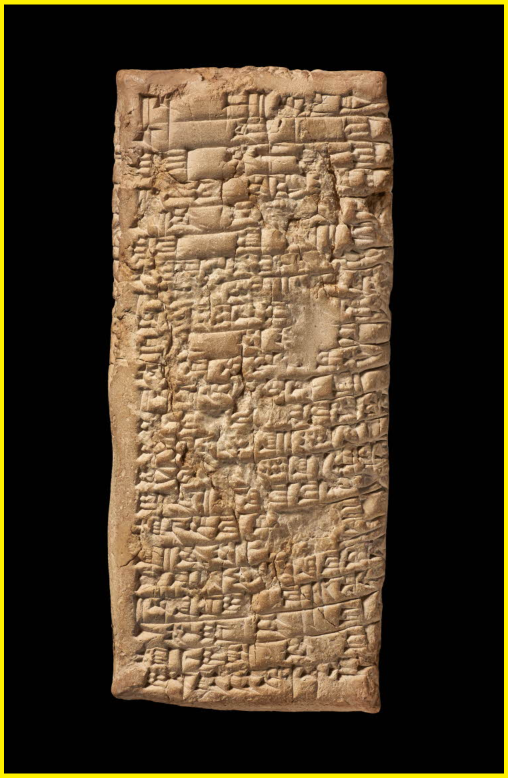
A Tour of Recommender Systems

Norm Matloff

University of California, Davis



Ancient "Yelp"

About This Book

The cover is from the British Museum onine site,

`https://www.britishmuseum.org/research/collection_online/collection_object_details/collection_image_gallery.aspx?partid=1&assetid=1613004116&objectid=277770`

with descripition:

> Clay tablet; letter from Nanni to Ea-nasir complaining that the wrong grade of copper ore has been delivered after a gulf voyage and about misdirection and delay of a further delivery...

Author's Biographical Sketch

Dr. Norm Matloff is a professor of computer science at the University of California at Davis, and was formerly a professor of statistics at that university. He is a former database software developer in Silicon Valley, and has been a statistical consultant for firms such as the Kaiser Permanente Health Plan.

Prof. Matloff's recently published books include'*Parallel Computation for Data Science* (CRC, 2015) and *Statistical Regression and Classification: from Linear Models to Machine Learning* (CRC 2017). The latter book was selected for the Ziegal Award in 2017.

Dr. Matloff was born in Los Angeles, and grew up in East Los Angeles and the San Gabriel Valley. He has a PhD in pure mathematics from UCLA, specializing in probability theory and statistics. He has published numerous papers in computer science and statistics, with current research interests in parallel processing, statistical computing, and regression methodology.

Prof. Matloff is a former appointed member of IFIP Working Group 11.3, an international committee concerned with database software security, established under UNESCO. He was a founding member of the UC Davis Department of Statistics, and participated in the formation of the UCD Computer Science Department as well. He is a recipient of the campuswide Distinguished Teaching Award and Distinguished Public Service Award at UC Davis.

Contents

Chapter 1

Setting the Stage

Let's first get an overview of the topic and the nature of this book. We'll first describe *collaborative filtering*.

1.1 What Are Recommender Systems?

What is a recommender system (RS)? We're all familiar with the obvious ones — Amazon suggesting books for us to buy, Twitter suggesting whom we may wish to follow, even OK Cupid suggesting potential dates.

But many applications are less obvious. The University of Minnesota, for instance, has developed an RS to aid its students in selection of courses. The tool not only predicts whether a student would like a certain course, but also even predicts the grade she would get!

In discussing RS systems, we use the terms *users* and *items*, with the numerical outcome being termed the *rating*. In the famous MovieLens dataset, which we'll use a lot, users provide their ratings of films.

Systems that combine user and item data as above are said to perform *collaborative filtering*. The first part of this book will focus on this type of RS. *Content-based* RS systems work by learning a user's tastes, say by text analysis.

Ratings can be on an ordinal scale, e.g. 1-5 in the movie case. Or it can be binary, such as a user clicking a Like symbol in Twitter, 1 for a click, 0 for no click.

But ratings in RSs are much more than just the question, "How much do you like it?" The Minnesota grade prediction example above is an instance of this.

In another example, we may wish to try to predict bad reactions to presciption drugs among patients in a medical organization. Here the user is a patient, the item is a drug, and the rating may be 1 for reaction, 0 if not.

More generally, and setting suitable for what in statistics is called a *crossed heirachical model* fits into RS. The word *crossed* here means that each user is paired with multiple items, and vice versa. The hierarchy refers to the fact that we can group users within items or vice versa. There wpould be two levels of hierarchy here, but there could be more.

Say we are looking at elementary school students rating story books. We could add more levels to the analysis, e.g. kids within schools within school districts. It could be, for instance, that kids in different schools like different books, and we should take that into account in our analysis. The results may help a school select textbooks that are especially motivational for their students.

Note that in RS data, most users have not rated most items. If we form a matrix of ratings, with rows representing users and columns indicating items, most of the elements of the matrix will be unknown. We are trying to predict the missing values.

1.2 How Is It Done?

Putting aside possible privacy issues that arise in some of the above RS applications,[1] we ask here, How do they do this? In this prologue, discuss a few of the major methods.

1.2.1 Nearest-Neighbor Methods

This is probably the oldest class of RS methodology, still popular today. It can be explained very simply.

Say there is a movie spoofing superheroes called *Batman Goes Batty* (BGB). Maria hasn't seen it, and wonders whether she would like it. To form a predicted rating for her, we could search in our dataset for the k users most similar to Maria in movie ratings and who have rated BGB. We would then average their ratings in order to derive a predicted rating of BGB for Maria. We'll treat the issues of choosing the value of k and defining "similar" later, but this is the overview.

[1] I used to be mildy troubled by Amazon's suggestions, but with the general demise of browsable bricks-and-mortar bookstores, I now tend to view it as "a feature rather than a bug."

1.2.2 Latent Factor Approach: Matrix Factorization

Let A denote the matrix of ratings described earlier, with A_{ij} denoting the rating user i gives to item j. Keep in mind, as noted, that most of the entries in A are unknown; we'll refer to them as NA, the R-language notation for missing values. Matrix factorization methods then estimate all of A as follows.

Let r and s denote the numbers of rows and colums of A, respectively. In the smallest version of the MovieLens data, for example, $r = 943$ and $s = 1682$. Recall that the *rank* of a matrix is its maximal number of linearly independent row and colums. (It can be shown that the row wank and column rank are the same.) So here the rank of A will be at most 943. If it is 943, we say the matrix is of *full rank*.

The idea is to find a *low-rank approximation* to A: We find matrices W and H, of dimensions $r \times k$ and $k \times s$, each of rank k, such that

$$A \approx WH \tag{1.1}$$

Typically $k << \min(r, s)$.

1.2.3 Latent Factor Approach: Statistical Models

As noted, collaborative-filtering RS applications form a special case of crossed random-effects models, a statistical methodology.

1.3 Covariates/Side Information

In predicting the rating for a given (user,item) pair, we may for example have demographic information on the user, such as age and gender. Incorporating such information — called *covariates* in statistics and *side information* in machine learning — may enhance our predictive ability, especially if this user has not rated many items to date.

1.4 Some Quick Infrastructure

There are some issues that will come up frequently. We'll treat them in detail later, but give overviews here.

1.4.1 Data as a Sample

In statistics, the data are usually considered to be a sample from a population. For instance, during an election campaign pollsters will take a sample, typically of 1200 people, from the population of all voters. Say they are interested in p, the population proportion of voters favoring Candidate Jones. They calculate Their estimate of p, denoted $\widehat{p}$, to be the proportion of voters in the sample who like Jones.

Sometimes the notion of sampling is merely conceptual. Say for instance we are studying hypertension, on data involving 1000 patients. We think of them as a sample from the population of all sufferers of hypertension, even though we did not go through an actual sampling process.

In machine learning circles, it is not customary to think explicitly in terms of populations, samples and estimates. Nevertheless, it's implied, as ML people do talk about predicting new data from the model fitted on the original data. For the model to be valid, the new data needs to come from the same source as the original — what statisticians call a population.

We will usually think in terms of sample data here.

1.4.2 Regression Models

Suppose we are predicting a variable Y from a vector X of variables, say predicting human weight from the vector (height,age). The *regression function* at $t = (t_1, t_2)$of Y on X is defined to be the

1.4.3 Bias, Variance and Overfitting

1.5 Prerequisites

1.6 What You Should Gain from This Book

Chapter 2

Nearest-Neighbor Methods

SHOW THE CODE FOR USING IT IN OUR 3 METHODS

GIVE A SIMPLE MATH DERIVATION OF VALUE OF IT, E.G. FOR SMALL N-I

Chapter 3

Matrix Factorization Methods

SHOW THE CODE FOR USING IT IN OUR 3 METHODS

GIVE A SIMPLE MATH DERIVATION OF VALUE OF IT, E.G. FOR SMALL N-I

Chapter 4

Statistical Models

SHOW THE CODE FOR USING IT IN OUR 3 METHODS

GIVE A SIMPLE MATH DERIVATION OF VALUE OF IT, E.G. FOR SMALL N-I

Chapter 5

Use of Side Information

SHOW THE CODE FOR USING IT IN OUR 3 METHODS

GIVE A SIMPLE MATH DERIVATION OF VALUE OF IT, E.G. FOR SMALL N-I

Chapter 6

Bias, Variance and Overfitting

GIVE EX FROM 132.TEX ON DENSITY ESTIMATION AT A POINT OF RISING DENSITY, MAYBE AT ENDPOINT OF SUPPORT, SO HAVE A CLEAR BIAS; MATH DERIVATION

APPLY TO K IN KNN, RANK OF MATRIX, FIXED- VS. RANDOM-EFFECTS IN STAT MODEL, VALUE OR NOT TO USING COVARS