07-380 Notes
Pre-reading: Recommender Systems

Carnegie Mellon University
School of Computer Science

Contents

1 Recommender Systems **1**
1.1 Examples . . . 1
1.2 Various Approaches . . . 2
1.3 Collaborative Filtering: Latent Factor Model . . . 3
1.3.1 Optimization . . . 3

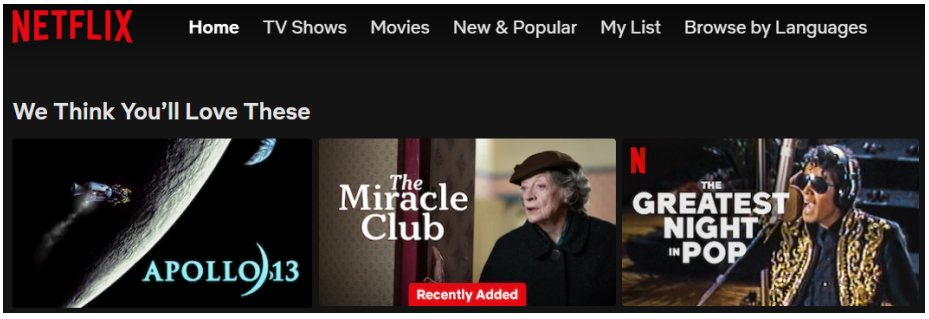
2 Nuclear Norm: Linear Algebra Background **4**
2.1 Vector Norms . . . 4
2.2 Trace . . . 4
2.3 Singular Values . . . 4
2.4 Rank . . . 4
2.5 Matrix Norms . . . 5

1 Recommender Systems

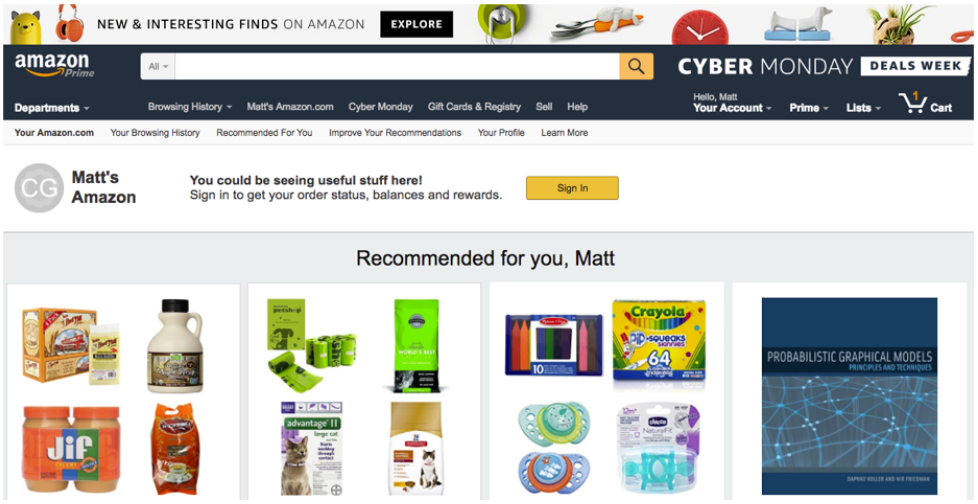
Recommender systems are a class of algorithms that predict which items (products) a user is most likely to be interested in. They are widely used in e-commerce, social media, and other online platforms to help users discover new products and services.

1.1 Examples

Example: Netflix movie recommendations Netflix uses a recommender system to recommend movies to users to help users quickly find movies they are likely to enjoy (and of course, keep them happy using the product).



Example: Amazon product recommendations Amazon uses a recommender system to recommend products to users based on their browsing and purchasing history. This helps users discover new products and increases the likelihood of a purchase.



1.2 Various Approaches

How do we decide which items to recommend to a user? That depends on what information we have. We might have features describing each item, features describing each user, or only a record of how users have rated items so far. Each of these leads to a different approach.

Item-based (content filtering). If we have features about each item, we can recommend items that are "close" to an item the user already likes. An online store knows a lot about every product it sells: the specifications, the manufacturer, the pictures, and so on. Pandora's music genome project described every song by hundreds of musical attributes (genre, tempo, instrumentation, vocals, ...); given one song as a starting point, it played other songs with similar attributes, so a listener could wander through the space of songs. Notice that this approach does not need to know anything about the user beyond the starting item.

User-based. If we have features about each user, we can look for other users who are "close" to a given user and recommend what those users liked. If a store knows a user's age, profession, and location, and those match another customer's, it may recommend the products that customer bought. This is the idea behind **market segmentation**: users with similar features are assumed to have similar preferences.

Learning the user-item relationship (collaborative filtering). Even without any features about the users or the items, we still have the ratings themselves: which users liked which items. **Collaborative filtering** techniques learn the relationship between users and items from this data alone. The underlying insight is that personal tastes are correlated, so the pattern of ratings we have already collected carries information about both the users and the items, enough to predict the ratings we have not seen yet.

These approaches are not exclusive; a real system may combine item features, user features, and ratings. In the rest of these notes we focus on one collaborative filtering technique, the latent factor model.

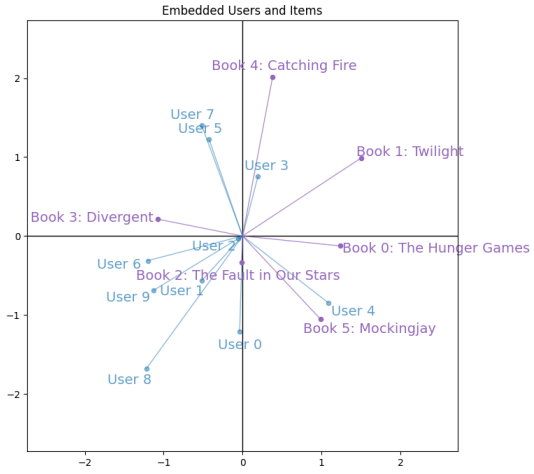
1.3 Collaborative Filtering: Latent Factor Model

One specific type of recommender system algorithm is the **latent factor model**. The idea behind latent factor models is to represent both users and items in a K-dimensional embedded space. Each user and item is represented by a K-dimensional vector, and the recommendation is made based on the similarity between the user and item vectors.

In the case where users have given ratings to items, the latent factor model tries to predict the ratings based on the learned user and item vectors. Specifically, we can try to predict the numerical rating of user i on item j as the dot product of the user and item vectors, $\hat{r} = \mathbf{u}_i^\top \mathbf{v}_j$.

This latent factor model can work even when we don't have any information or features about the users or the items other than a dataset of user-item ratings!

Here's a quick example of book items (purple) and users (blue) embedded in a $K = 2$ dimensional space. These are just the initial random parameters in U and V. As we train, it is likely that similar books will move closer to each other, as will readers with similar interests.



1.3.1 Optimization

The specific optimization problem is as follows:

- Input: N users, M items, and a dataset of user-item ratings $\mathcal{S} = \{(i, j, r)^{(1)}, (i, j, r)^{(2)}, \cdots\}$
- Hyperparameter: K, the size of the embedded space
- Parameters: $U \in \mathbb{R}^{N \times K}$ and $V \in \mathbb{R}^{M \times K}$, where the i^{th} row of U, $\mathbf{u}_i$, is the K-dimensional vector learned to represent user i and the j^{th} row of V, $\mathbf{v}_j$, is the K-dimensional vector learned to represent item j

We will use squared error as the loss function to measure the difference between an actual rating from user i on item j, r, and the predicted rating $\hat{r} = \mathbf{u}_i^\top \mathbf{v}_j$. The objective function is then the mean squared error over the dataset:

$$J(U, V) = \frac{1}{|\mathcal{S}|} \sum_{(i,j,r) \in \mathcal{S}} (r - \mathbf{u}_i^\top \mathbf{v}_j)^2$$

Each rating tuple contributes one term to this sum. We write that term as

$$J^{(i,j,r)}(U, V) = (r - \mathbf{u}_i^\top \mathbf{v}_j)^2, \qquad \text{so that} \qquad J(U, V) = \frac{1}{|\mathcal{S}|} \sum_{(i,j,r) \in \mathcal{S}} J^{(i,j,r)}(U, V).$$

Once we compute the gradients of a single term $J^{(i,j,r)}$ with respect to its user and item vectors, we can run stochastic gradient descent to optimize the user and item vectors, where each data point is a single user-item rating tuple (i, j, r) and each step updates only $\mathbf{u}_i$ and $\mathbf{v}_j$:

$$\mathbf{u}_i^{(t+1)} \leftarrow \mathbf{u}_i^{(t)} - \alpha \nabla_{\mathbf{u}_i} J^{(i,j,r)}(U^{(t)}, V^{(t)})$$
$$\mathbf{v}_j^{(t+1)} \leftarrow \mathbf{v}_j^{(t)} - \alpha \nabla_{\mathbf{v}_j} J^{(i,j,r)}(U^{(t)}, V^{(t)})$$

2 Nuclear Norm: Linear Algebra Background

A brief refresher of the linear algebra behind the nuclear norm. Throughout, $\mathbf{v} \in \mathbb{R}^N$ is a vector with entries v_i and $A \in \mathbb{R}^{N \times M}$ is a real matrix with entries a_{ij}.

2.1 Vector Norms

- ℓ_2 norm (review): $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^N v_i^2} = \sqrt{\mathbf{v}^\top \mathbf{v}}$
- ℓ_1 norm: $\|\mathbf{v}\|_1 = \sum_{i=1}^N |v_i|$

Example: $\mathbf{v} = [3, -4]^\top$ has $\|\mathbf{v}\|_2 = 5$ and $\|\mathbf{v}\|_1 = 7$.

2.2 Trace

For a square matrix $A \in \mathbb{R}^{N \times N}$ with eigenvalues $\lambda_1, \ldots, \lambda_N$ (counted with multiplicity), the **trace** is the sum of the diagonal entries, which equals the sum of the eigenvalues:

$$\operatorname{tr}(A) = \sum_{i=1}^N a_{ii} = \sum_{i=1}^N \lambda_i$$

For any $A \in \mathbb{R}^{N \times M}$, $\operatorname{tr}(A^\top A) = \sum_{i=1}^N \sum_{j=1}^M a_{ij}^2$.

Example: $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has eigenvalues 3 and 1, and $\operatorname{tr}(A) = 2 + 2 = 3 + 1 = 4$.

2.3 Singular Values

For $A \in \mathbb{R}^{N \times M}$, the $M \times M$ matrix $A^\top A$ is symmetric and its eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \geq 0$ are real and nonnegative. The **singular values** of A are their square roots,

$$\sigma_i = \sqrt{\lambda_i}, \qquad i = 1, \ldots, M, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_M \geq 0.$$

($AA^\top$ has the same nonzero eigenvalues as $A^\top A$, so the nonzero singular values can be computed from either product.)

Example: $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ gives $A^\top A = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ with eigenvalues 4 and 0, so the singular values are $\sigma_1 = 2$ and $\sigma_2 = 0$.

2.4 Rank

The **rank** of $A \in \mathbb{R}^{N \times M}$ is the number of linearly independent columns of A (equivalently, rows), so $\operatorname{rank}(A) \leq \min(N, M)$. It equals the number of nonzero singular values of A. A is **low rank** when

$\text{rank}(A)$ is much smaller than $\min(N, M)$.

Rank of a product: $\text{rank}(AB) \leq \min\big(\text{rank}(A), \text{rank}(B)\big)$. In particular, if $U \in \mathbb{R}^{N \times K}$ and $V \in \mathbb{R}^{M \times K}$, then $UV^\top \in \mathbb{R}^{N \times M}$ has rank at most K, however large N and M are.

Example: $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ has rank 1 (its two columns are equal), matching its single nonzero singular value, and it factors as $UV^\top$ with $U = V = [1, 1]^\top$, i.e. $K = 1$.

2.5 Matrix Norms

Frobenius norm (review): the ℓ_2 norm of the entries of A, which is also the ℓ_2 norm of its singular values:

$$\|A\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij}^2} = \sqrt{\text{tr}(A^\top A)} = \sqrt{\sum_i \sigma_i^2}$$

Nuclear norm (new): the ℓ_1 norm of the singular values of A:

$$\|A\|_* = \sum_i \sigma_i$$

Examples:

A	singular values	$\text{rank}(A)$	$\|A\|_F$	$\|A\|_*$
$\begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix}$	3, 2	2	$\sqrt{13} \approx 3.61$	5
$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	2, 0	1	2	2
$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1, 1	2	$\sqrt{2} \approx 1.41$	2