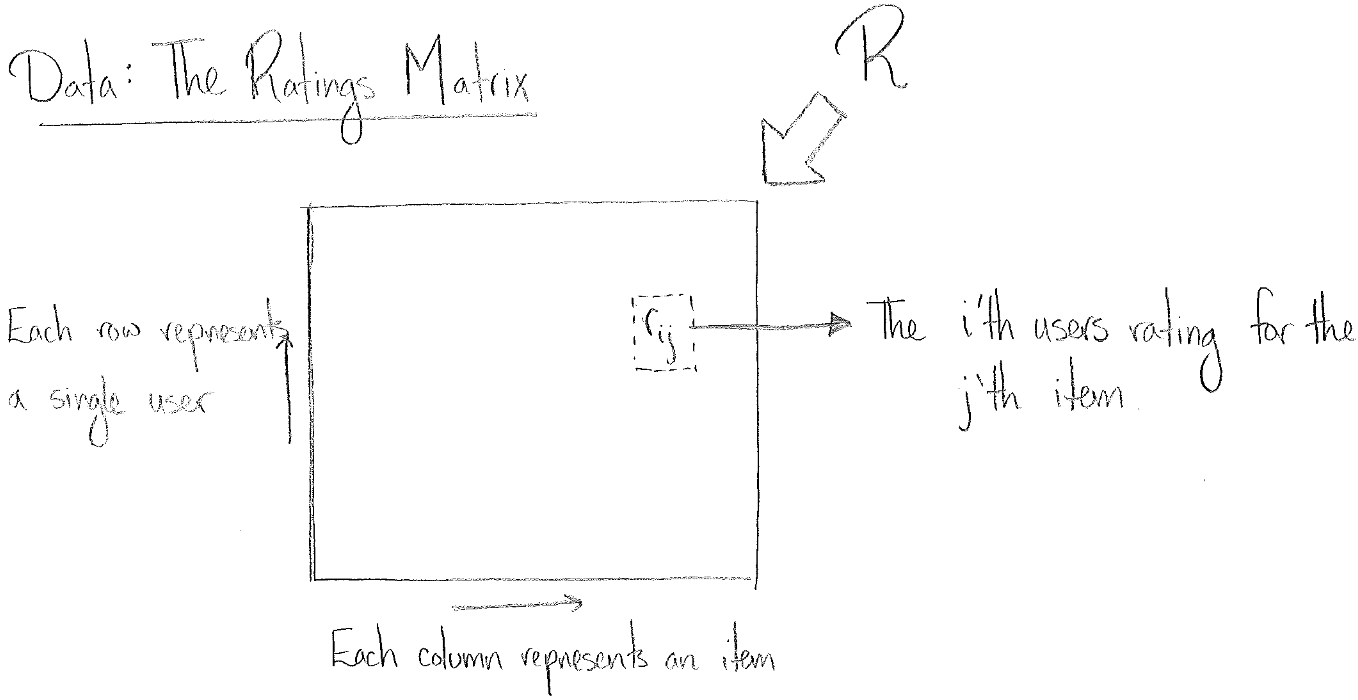
Reccomendation Systems

Morning: Similarity based / Collaborative Filtering

Afternoon: Matrix Factorization

Data: The Ratings Matrix



Usually, most entries are missing, since most users have not rated most items!

Ratings can be explicit or implicit

Explicit: User supplies ratings for items

Implicit: User consumes selected items, measure of consumption is taken as a rating.

Goal: Predict missing ratings!

r_{ij}: Actual rating of user i for item j

$\hat{r}_{ij}$: Predicted rating of user i for item j.

Similarity Based

User - User Hypothesis:

Similar users tend to give similar ratings to a single product.

Item - Item Hypothesis:

A single user will tend to give similar ratings to similar products.

Need: A way to measure similarity between users and/or items.

Then: We can predict ratings as a weighted average of the actual ratings

- of similar users (for a fixed item)
- of similar products (for a fixed user)

User - User:

$$\hat{r}_{ij} = \frac{\displaystyle\sum_{\substack{\text{users } u \text{ that have} \\ \text{rated item } j}} \overbrace{\text{sim}(i,u)}^{\text{user - user similarity}} \, r_{uj}}{\displaystyle\sum_{u} \text{sim}(i,u)}$$

↑ rating for fixed item j.

Item - Item:

$$\hat{r}_{ij} = \frac{\displaystyle\sum_{\substack{\text{items } t \text{ that are} \\ \text{rated by user } i}} \overbrace{\text{sim}(j,t)}^{\text{item - item similarity}}}{\displaystyle\sum_{t} \text{sim}(j,t)}$$

Discussions:

1. What are the possible benefits / drawbacks of both approaches.
2. How may we make thes calculations more efficient?

Similarity Measures :

Requirements :

1. $sim(a, b)$ is between (inclusive) zero and one.
2. $sim(a,b) = 0$ means "a and b are not at all similar".
3. $sim(a,b) = 1$ means "a and b are as similar as possible".

Similarity between users / items is based off the rows (user-user) or columns (item-item) of the rating matrix.

Cosine Similarity :

$$\cos(\theta_{\vec{a},\vec{b}}) = \frac{\vec{a}}{|\vec{a}|} \cdot \frac{\vec{b}}{|\vec{b}|}$$

Notation

$\vec{a}, \vec{b}$: rows (user-user) or columns (item-item) of the rating matrix R.

$$\text{cos-sim}(\vec{a}, \vec{b}) = \frac{1}{2} + \frac{1}{2}\cos(\theta_{\vec{a},\vec{b}})$$

Pearson - Correlation Similarity

$$\text{corr}(\vec{a}, \vec{b}) = \frac{\text{cov}(\vec{a}, \vec{b})}{\text{sd}(\vec{a})\,\text{sd}(\vec{b})}$$

$$\text{corr-sim}(\vec{a}, \vec{b}) = \frac{1}{2} + \frac{1}{2}\text{corr}(\vec{a}, \vec{b})$$

Discussion :

When are these large (≈ 1)

When are these small (≈ 0)

Jaccard - Similarity

R must be binary : user consumed item or not ?

$$\text{jacc-sim}(\vec{a}, \vec{b}) = \frac{\#\text{ of items consumed by both a and b}}{\#\text{ of items consumed by either a or b}}$$

Matrix Factorization Methods

Inspiration: Content Based Preference

User content preferences:

	Action	Strategy	Story	Exploration	Collection
Matt	3	3	1	4	2
Caitlyn	1	4	4	2	5

Call this matrix U

Item content attributes

	Action	Strategy	Story	Exploration	Collection
Zelda	4	3	2	5	3
Mario	5	2	1	3	3
Animal Crossing	1	2	3	2	5

Call this matrix V^t

Overall preference of user for items is a dot-product

$$\text{pref}(\text{Matt}, \text{Zelda}) = 3\times4 + 3\times3 + 1\times2 + 4\times5 + 2\times3 = 49$$

$$\text{pref}(\text{Caitlyn}, \text{Animal Crossing}) = 1\times1 + 4\times2 + 4\times3 + 2\times2 + 5\times5 = 50$$

$$\text{pref}(\text{Matt}, \text{Animal Crossing}) = 3\times1 + 3\times2 + 1\times3 + 4\times2 + 2\times5 = 33$$

Idea: Is it possible to learn U and V when we take ratings as an expression of preferences?

Matrix Factorization

If we take ratings as an expression of preferences, the our content based setup results in the matrix equation:

$$\hat{R} = UV^t$$

U is (# users) $\times K$

V is (# items) $\times K$

K is a hyperparameter

So each predicted rating is a dot product

$$\hat{r}_{ij} = \sum_{K} u_{iK} v_{jK}$$

To learn U and V, we want this to accurately reproduce the ratings we know:

$$R \approx UV^t$$ ← Remember, a lot of R is missing, so this equation only applies to the non-missing values.

The next step is familiar, we need to measure the quality of our predictions, and we use least squares:

$$\hat{U}, \hat{V} = \underset{U,V}{\operatorname{argmin}} \left\{ \sum_{r_{ij} \text{ ratings in } R} \left(r_{ij} - \sum_{K} u_{iK} v_{jK} \right)^2 \right\}$$

This problem is easily solved with gradient descent, which has very simple update rules:

$$\left.\begin{aligned} \frac{\partial L}{\partial u_{iK}} &= 2\sum_{r_{ij}} (r_{ij} - \hat{r}_{ij})\, u_{iK} \\ \frac{\partial L}{\partial v_{iK}} &= 2\sum_{r_{ij}} (r_{ij} - \hat{r}_{ij})\, v_{jK} \end{aligned}\right\}$$ These are the components of the gradient of L.

Comments

① We are estimating (# users + # items) × K parameters, which is a lot. So, regularization is useful:

$$\hat{U}, \hat{V} = \underset{U, V}{\operatorname{argmin}} \left\{ \sum_{r_{ij}} \left(r_{ij} - \vec{u}_i \cdot \vec{v}_j \right)^2 + \lambda \left(\sum_{i,k} u_{ik}^2 + \sum_{j,k} v_{jk}^2 \right) \right\}$$

This doesn't affect the difficulty of fitting the model with gradient descent.

② K is a hyper parameter, it must be tuned with cross validation. But, be careful about removing <u>all</u> ratings for a user or item!

③ Similarly, matrix factorization cannot provide ratings for new users or items. You need a fallback methodology for these cases!

④ Some users/items have different ranges for ratings

- Some users rate everything 4 or 5 stars.
- Some products are garbage, and are always rated 1 or 2 stars.

You can account for this with user and item level parameters:

$$\hat{r}_{ij} = \vec{u}_i \cdot \vec{v}_j + b_i^u + b_j^v + \mu$$

overall average rating

item j's deviation from the average

user i's deviation from the average