Machine Learning

By Ghazal Lalooha

Recommender Systems

Table of Contents

Introduction

Introduction

Group refinement

Group refinement

- The most prominent approach used in recommender systems:
  - Used by very large commercial sites
  - Contains different types of algorithms
  - Can be used in many domains (books, movies, music, etc.)
- Approach:
  - Using "collective wisdom" to recommend products
- Ideas:
  - Users rate the purchased goods. (usually between 1 and 5)
  - Users who have similar tastes in the past will probably have similar tastes in the future.

Group refinement

- Input: A matrix containing points given by users to various products.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

- Output types:
- A (numerical) prediction about the user's interest in a particular product
- A suggested list of N top items

User-based group filtering

- Basic method:
  - Given an active user such as Alice and an item i that has not been previously seen by Alice:
    - Find a set of users (nearest neighbors) who have similar tastes to Alice and have previously rated item i.
    - Calculate the average score given by these users to product i.
    - Use the calculated mean as an estimate of Alice's interest in item i.
    - Repeat this for all items that Alice has not rated.
    - Offer more points to Alice.
- Ideas:
  - Users who have similar tastes in the past will probably have similar tastes in the future.

User-based group filtering

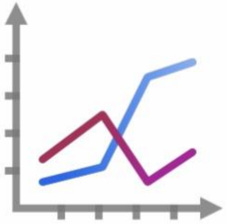
Example: input:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Objective: predicting Alice's interest in item number 5

User-based group filtering

- A few basic questions:
  - How to calculate the similarity between different users?
  - How many neighbors should be considered?
  - How can you make a prediction based on the score of the neighbors?



	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Criteria for measuring users' similarity

- Pearson's correlation coefficient: a widely used measure.
- a and b: users
- $r_{a,p}$: rating given by user a to product p
- P: A set of products rated by both users a and b.

$$\boxed{sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}}$$

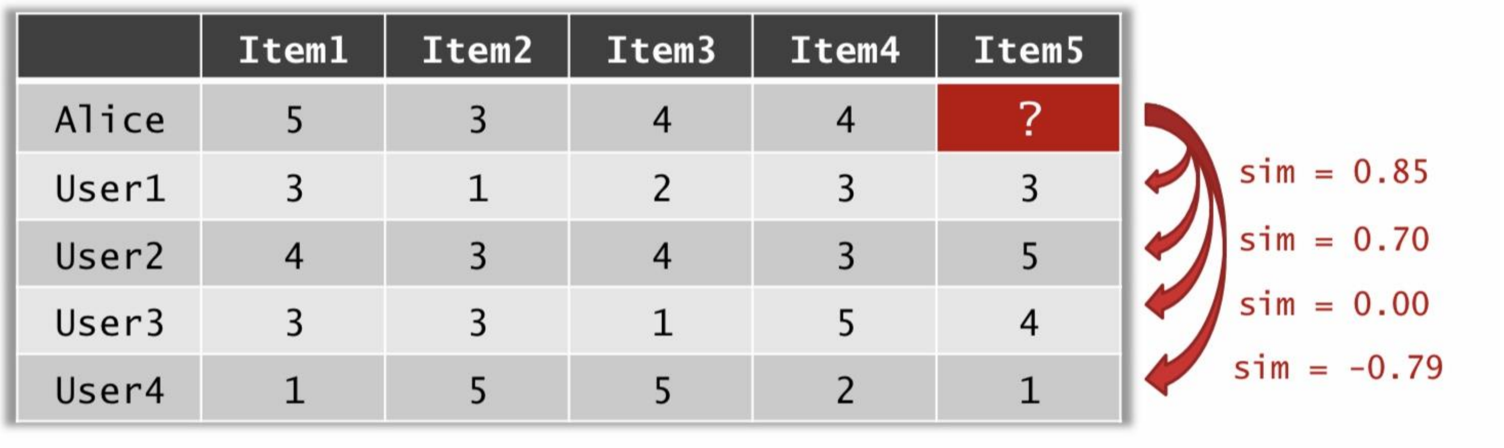
$$\frac{cov(a,b)}{std(a) \cdot std(b)}$$

$$\frac{1}{2} + \frac{1}{2} sim(a,b)$$

نرمال‌سازی

Criteria for measuring users' similarity

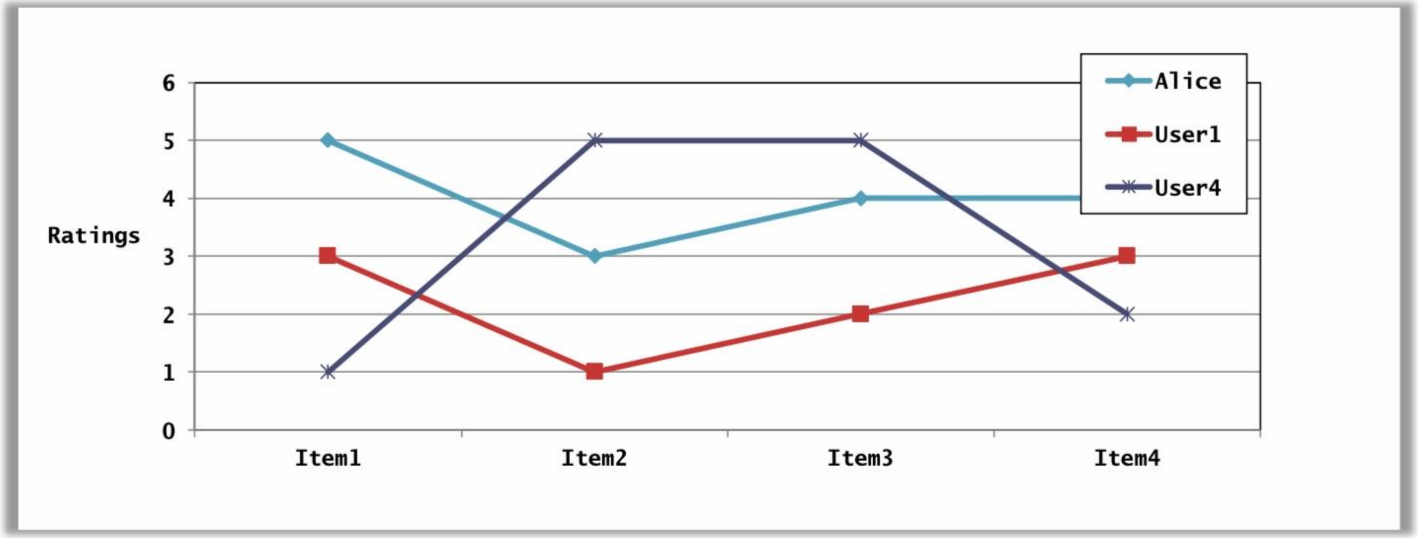
- Pearson's correlation coefficient: a widely used measure.
- a and b: users
- $r_{a,p}$: rating given by user a to product p
- P: A set of products rated by both users a and b.



	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Pearson

- Advantage: taking into account differences in scoring habits.
- Pearson's correlation coefficient performs better than other measures in many domains.
- (Euclidean criterion, cosine criterion)



Prediction

- A common function for predicting:

$$pred(a,p) = \bar{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r_b})}{\sum_{b \in N} sim(a,b)}$$

  - Calculate whether the neighbor's score for item p is lower or higher than their average.
  - Combining Differences - Using Similarity Criteria for Weighting
  - Add the average score of user a to the calculated value

Memory-based and model-based approaches:

- User-based batch filtering is a "memory-based" method.
  - Direct use of score matrix to find nearest neighbors and prediction
  - In many real-world applications, this approach is not applicable!
    - Due to tens of millions of users and millions of products
- Model-based approaches:
  - based on learning the model offline (offline training)
  - At runtime, only the learned model is used for prediction.
  - Models are used intermittently.
  - Creating a model and updating it can be computationally expensive.

Item-based group refinement

- Ideas:
  - Using similarity between products for prediction (not similarity between users)
- Example: Search for products similar to product 5
  - Using scores given by Alice to similar products to predict product scores 5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Item-based group refinement

- Cosine similarity criterion:
  - Produce better results in product-to-product comparisons
  - The scores given to each item are considered as a vector in n-dimensional space.
  - The similarity between two goods is measured by calculating the cosine of the angle corresponding to the vector of these two goods:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- Adjusted cosine similarity measure:
  - Considering the average score of each user
  - U: The set of users who rated both products a and b.

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r_u})(r_{u,b} - \bar{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \bar{r_u})^2}}$$

Prediction

- A common function for predicting:

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)}$$

- Usually, the size of the neighborhood is limited.
- That is, not all neighbors are used for prediction.
- A rule of thumb: In many real-world applications, the number of neighbors is considered to be between 20 and 50. (Herlocker, 2002

The problem of low data population

- Cold start issue:
  - How can you recommend new products?
  - How to advise new users?
- Simple solutions:
  - Ask the user to rate a set of data.
  - In the early stages, use other methods such as content-based filtering.
  - Default Values: Use default values for items rated by only one of the two users to be compared.

A variety of model-based approaches

- Decomposition of matrices:
  - Single value analysis, principal component analysis
- Exploring communication rules:
  - Compare: Shopping Cart Analysis
- Probabilistic models:
  - Clustering, Bayesian networks and...
- Preprocessing cost (model learning):
  - It is not usually talked about.
  - Is a gradual update possible?

Single value analysis

- Motivation:
  - Data simplification
  - Removal of noise and redundancy
  - Improve algorithm results
- Example applications:
  - Information search and retrieval (latent semantic indexing)
  - Recommender systems

Single value analysis

- Single value analysis

$$Data_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^{T}_{n\times n}$$

Single value matrix

- Single value matrix:
  - A diagonal matrix in which the individual values are ordered in descending order.
  - Single values from an index such as r onwards have zero value.
  - The singular values are the square roots of the Data × Data ^{T} matrix.

Content-based recommendations

Content-based recommendations

- Training: Learn the real vector $\theta^{(j)}$ for user j.
- Prediction: rating of movie i for user j

$$(\theta^{(j)})^T x^{(i)}$$

	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1	x_2
Titanic	5	5	0	0	0.90	0.00
Sound and Music	5	?	?	0	1.00	0.01
Casablanca	?	4	0	?	0.99	0.00
Fast and Furious	0	0	5	4	0.10	1.00
Desperado	0	0	5	?	0.00	0.90

Formal statement of the problem

- $r(i,j) = 1$ if user j rated movie i, zero otherwise
- $y^{(i,j)}$ rating given by user i to movie j
- $\theta^{(j)}$ vector of parameters for user j;
- $x^{(i)}$ is the feature vector for movie i
- Predicting the rating of movie i for user j:

$$(\theta^{(j)})^T x^{(i)}$$

- $m^{(j)}$ number of movies rated by user j

The goal of optimization

- Learning vector $\theta^{(j)}$ -- parameters for user j

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2$$

- Learning vectors $\theta^{(1)}$, $\theta^{(2)}$, ..., $\theta^{(nu)}$

$$\min_{\theta^{(1)},\ldots\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2$$

Optimization Algorithm

Objective function:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2$$

Gradient Descent:

$$\theta_k^{(j)} = \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} = \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} + \lambda\theta_k^{(j)}\right) \quad (\text{for } k \neq 0)$$

Group refinement

Group refinement

	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1	x_2
Titanic	5	5	0	0	0.90	0.00
Sound and Music	5	?	?	0	1.00	0.01
Casablanca	?	4	0	?	0.99	0.00
Fast and Furious	0	0	5	4	0.10	1.00
Desperado	0	0	5	?	0.00	0.90

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \quad \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

$$\left(\theta^{(1)}\right)^T x^{(1)} \approx 5$$
$$\left(\theta^{(2)}\right)^T x^{(1)} \approx 5$$
$$\left(\theta^{(3)}\right)^T x^{(1)} \approx 0$$
$$\left(\theta^{(3)}\right)^T x^{(1)} \approx 0$$

The goal of optimization

Learning x $^{(i)}$ by having $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$ parameters:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

Learning $x^{(1)}, x^{(2)}, \ldots, x^{(nu)}$
by having $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$
parameters:

$$\min_{x^{(1)},\ldots x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

Group refinement

- Idea : by having score matrix and $x^{(1)}, x^{(2)}, \ldots, x^{(nu)}$ vectors, we can estimate $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$ vectors.

- Idea: by having score matrix and $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$ vectors, we can estimate $x^{(1)}, x^{(2)}, \ldots, x^{(nu)}$ vectors.

- Algorithm:

Random initialization

$$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \cdots$$

Group refinement Algorithm

Group refinement

estimating $\theta^{(1)}$, $\theta^{(2)}$, ..., $\theta^{(nu)}$ vectors, by $x^{(1)}$, $x^{(2)}$, ..., $x^{(nu)}$ vectors:

$$\min_{\theta^{(1)},\dots\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2$$

estimating $x^{(1)}$, $x^{(2)}$, ..., $x^{(nu)}$ vectors, by $\theta^{(1)}$, $\theta^{(2)}$, ..., $\theta^{(nu)}$ vectors:

$$\min_{x^{(1)},\dots x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Optimization objective in group refinement

- Idea: simultaneous learning of feature vectors x $^{(i)}$ and vectors $\theta^{(j)}$

- The objective function:

$$J\left(x^{(1)},\cdots,x^{(n)},\theta^{(1)},\cdots,\theta^{(n)}\right) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_u}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2$$

- Goal:

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)}\\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J\left(x^{(1)},\cdots,x^{(n)},\theta^{(1)},\cdots,\theta^{(n)}\right)$$

Group refinement Algorithm

Education:
Initialize x and theta vectors with small random values
Cost function minimization using gradient descent (or advanced optimization methods)

$$x_k^{(i)} = x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right) \theta_k^j + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} = \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

Prediction:
for user j with parameter vector $\theta^{(j)}$ and movie i with feature vector $x^{(i)}$

$$\left(\theta^{(j)}\right)^T x^{(i)}$$

Mean normalization

New Users

	Alice(1)	Bob(2)	Carol(3)	Dave(4)	eve(5)	
Titanic	5	5	0	0	?	0
Sound and Music	5	?	?	0	?	0
Casablanca	?	4	0	?	?	0
Fast and Furious	0	0	5	4	?	0
Desperado	0	0	5	?	?	0

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_u}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2$$

$$\theta^{(5)} = \begin{bmatrix}0\\0\end{bmatrix} \rightarrow \left(\theta^{(5)}\right)^T x^{(i)} = 0$$

Mean normalization

Mean normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \qquad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow \qquad Y_{norm} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2.0 & -2.0 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

Prediction: the level of interest of user j in movie i

$$\hat{y}(i,j) = \left(\theta^{(j)}\right)^T x^{(i)} + \mu^{(i)}$$