Data Mining
Lecture 2: Recommender Systems

Jo Houghton

ECS Southampton

January 11, 2019

Recommender Systems - Introduction

Making recommendations: Big Money

35% of amazons income from recommendations

Netflix recommendation engine worth 1$ Billion per year

Recommender Systems - Introduction

Making recommendations: Big Money
35% of amazons income from recommendations
Netflix recommendation engine worth 1$ Billion per year

When you know very little
about that person,
suggesting:

- things to buy,
- films to watch,
- books to read,
- people to date,

..is hard.

Recommender Systems - Introduction

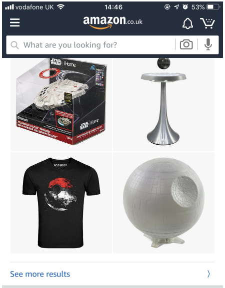
Making recommendations: Big Money
35% of amazons income from recommendations
Netflix recommendation engine worth 1$ Billion per year

And yet, Amazon seems to be able to recommend stuff I like.

When you know very little about that person, suggesting:

- things to buy,
- films to watch,
- books to read,
- people to date,

..is hard.



Recommender Systems - Problem Formulation

You have a set of films and a set of users. The users have rated some of the films.

Film	Alice	Bob	Carol	Dave
Love Really	4	1		4
Deadly Weapon		1	4	5
Fast and Cross	5		5	4
Star Battles	1	5		

How can you predict what should go in the blanks?

Recommender Systems - Algorithms

They use one of three different types of algorithm:

- Collaborative Filtering
- Content Based Filtering
- Hybrid Recommender Systems

Recommender Systems - Content based approach

Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Each film can be described by the vector $X = [1 \quad x_1 \quad x_2]$ (1 is for the bias term)

Learn 2D parameter vector θ, where $\theta^T X$ gives the number of stars for each user.
$\theta = [0 \quad 5 \quad 0]$ for someone who really likes romance films

Recommender Systems - Content based approach

Use Linear Regression to find user parameter vector θ where m is the number of films rated by that user

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \tag{1}$$

Can also use Bayesian classifiers, MLPs, etc.

Problems?

Recommender Systems - Content based approach

Use Linear Regression to find user parameter vector θ where m is the number of films rated by that user

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \tag{1}$$

Can also use Bayesian classifiers, MLPs, etc.

Problems?
requires hand coded knowledge of film
not easy to scale up
user may not have rated many films

Recommender Systems - Collaborative Filtering

Collaborative Filtering example:
Alice likes Dr Who, Star Wars and Star Trek
Bob likes Dr Who and Star Trek
A recommender system would correlate the likes, and suggest that Bob might like Star Wars too.
Personal preferences can be correlated.

Task: Discover patterns in observed behaviour across a community of users

- Purchase history
- Item ratings
- Click counts

Predict new preferences based on those patterns

Recommender Systems - Collaborative Filtering

Collaborative filtering uses a range of approaches to accomplish this task

- Neigbourhood based approach
- Model based approach
- Hybrid (Neighbourhood and model) based approach

This lecture will cover the Neighborhood based approach

Collaborative Filtering

Measure user preferences. Eg. Film recommendation
Users rate films between 0 and 5 stars

	Lady in the Water	Snakes on a Plane	Just my Luck	Superman Returns	The Night Listener	You, Me and Dupree
Lisa	2.5	3.5	3.0	3.5	3.0	2.5
Gene	3.0	3.5	1.5	5.0	3.0	3.5
Mike	2.5	3.0		3.5	4.0	
Jane		3.5	3.0	4.0	4.5	2.5
Mick	3.0	4.0	2.0	3.0	3.0	2.0
Jill	3.0	4.0		5.0	3.0	3.5
Toby		4.5		4.0		1.0

The data is sparse, there are missing values

Collaborative Filtering - Sparsity

Sparsity can be taken advantage of to speed up computations
Most libraries that do matrix algebra are based on LAPACK, written in Fortan90
Computation is done by calls to the Basic Linear Algebra Subprograms (BLAS).

This is how the Python `numpy` library does its linear algebra.

Collaborative Filtering - Sparsity

Compresssed Row Storage (CRS) [1]

Matrix specified by three arrays: *val*, *row_ind* and *col_ptr*

val stores the non zero values

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start arow

eg:

val = [1, 2, 9, 8, 2, -1, 4, 5, 2, 7]

col_ind = [1, 2, 4, 3, 4, 1, 2, 4, 2, 3]

row_ptr = [1, 4, 6, 9]

[1] Harwell-Boeing sparse matrix format, Duff et al, ACM Trans. Math. Soft., 15 (1989), pp. 1-14.

Collaborative Filtering - Sparsity

Compresssed Row Storage (CRS) [1]

Matrix specified by three arrays: *val*, *row_ind* and *col_ptr*

val stores the non zero values

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start arow

eg:

val = [1, 2, 9, 8, 2, -1, 4, 5, 2, 7]

col_ind = [1, 2, 4, 3, 4, 1, 2, 4, 2, 3]

row_ptr = [1, 4, 6, 9]

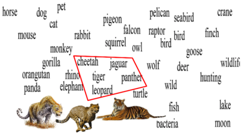
$$\begin{bmatrix} 1 & 2 & & 9 \\ & & 8 & 2 \\ -1 & 4 & & 5 \\ & 2 & 7 & \end{bmatrix}$$

[1] Harwell-Boeing sparse matrix format, Duff et al, ACM Trans. Math. Soft., 15 (1989), pp. 1-14.

Collaborative Filtering - Feature Extraction

Features are stored in a 'feature vector', a fixed length list of numbers.

- The length of this vector is the number of dimensions
- Each vector represents a point and a direction in the featurespace.
- Each vector must have the same dimensionality



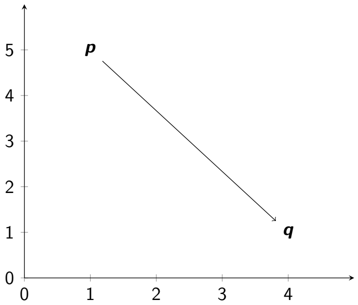
A t-SNE projection of encoded word vectors shows that similar words are close to each other in feature space.

We say two things are *similar* if they have similar feature vectors, i.e. are close to each other in featurespace.

Collaborative Filtering - Distance

There are three ways to measure distance in feature space:

- Euclidean Distance:



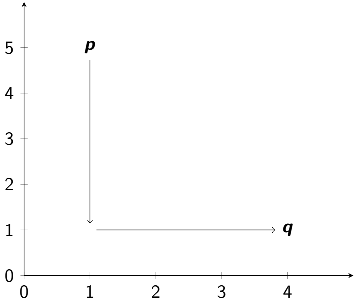
$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

Euclidean distance:

$$||\boldsymbol{p} - \boldsymbol{q}|| = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2} \tag{2}$$

Collaborative Filtering - Distance

- Manhattan Distance:



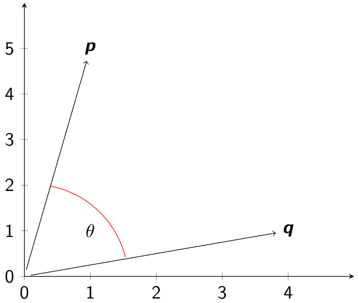
$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

Manhattan distance:

$$||\boldsymbol{p} - \boldsymbol{q}||_1 = \sum_{i=1}^{N}(q_i - p_i) \quad (3)$$

Collaborative Filtering - Distance

- Cosine Similarity



Only measures direction, not magnitude of vector.

$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

Cosine Similarity:

$$cos(\theta) = \frac{\boldsymbol{p}.\boldsymbol{q}}{|\boldsymbol{p}||\boldsymbol{q}|}$$

$$= \frac{\sum_{i=1}^{N} p_i q_i}{\sqrt{\sum_{i=1}^{N} p_i^2}\sqrt{\sum_{i=1}^{N} q_i^2}}$$

Collaborative Filtering - User Similarity

Need to define a similarity score, based on the idea that similar users have similar tastes, i.e. like the same movies.)

Needs to take in to account sparsity, not all users have seen all movies.

Typically between 0 and 1, where 1 is the same, and 0 is totally different

Can visualise the users in feature space, using two dimensions at a time

space_vis demo ipynb

Collaborative Filtering - User Similarity

There are many ways to compute similarity based on Euclidean distance

We *could* chose:

$$sim_{L2}(x, y) = \frac{1}{1 + \sqrt{\sum_{1 \in I_{xy}} (r_{x,1} - r_{y,i})^2}}$$

where $r_{x,i}$ is the rating from user x for item i

I_{xy} is set of items rated by both x and y

i.e. when the distance is 0, the similarity is 1, but when the distance is large, similarity $\rightarrow 0$

Collaborative Filtering - User Similarity

Alternatively, calculate correlation of users, based on ratings they share

Using Pearson's Correlation: standard measure of dependence between two related variables.

$$sim_{Pearson}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Where $\bar{r_x}$ is average rating user x gave for all items in I_{xy}

Demo: pearson in ipynb

Collaborative Filtering - User Similarity

Users are inconsistent. Some users always give out 5s to films they like, whereas some are more picky.
For example, look at Lisa and Jill, both rank the films in the same order, but give different ratings.
Pearson correlation corrects for this, but Euclidean similarity doesn't.
Data normalisation and mean centering can overcome this Data standardisation.

Collaborative Filtering - User Filtering

We now have a set of measures for computing the similarity between users

Produce a ranked list of best matches to a target user. Typically want the top-N users

May only want to consider a subset of users, i.e. those who rated a particular item.

Demo: Ranking users by similarity

Collaborative Filtering - Recommending

Now we have a list of similar users, how can we recommend items?
Predict rating $r_{u,i}$ of item i by user u as an aggregation of the ratings of item i by users similar to u

$$r_{u,i} = aggr_{\hat{u} \in U}(r_{\hat{u},i})$$

Where U is the set of *top* users most similar to u that rated item i
Multiply the score by the similarity of the user
Normalise by sum of similarities (otherwise items rated more often will dominate)

$$r_{u,i} = \frac{\sum_{\hat{u} \in U} sim(u, \hat{u}) r_{\hat{u},i}}{\sum_{\hat{u} \in U} |sim(u, \hat{u})|}$$

This is *User Based Filtering*
Demo:User based recommendation

Collaborative Filtering - User Based Filtering

Can also aggregate by computing average over similar users

$$r_{u,i} = \frac{1}{N} \sum_{\hat{U} \in U} r_{\hat{U},i}$$

Or by subtracting the average user rating score for all the items they scored, this is to compensate for people that judge generously or meanly.

$$r_{u,i} = \bar{r}_u + \frac{\sum_{\hat{u} \in U} sim(u, \hat{u})(r_{\hat{u},i} - \bar{r}_{\hat{u}})}{\sum_{\hat{u} \in U} |sim(u, \hat{u})|}$$

Collaborative Filtering - User Based Filtering

We can also compute similarity between items, using the same method.
this provides a *fuzzy* basis for recommending alternative items.
There are more structured ways of identifying what products people buy together using "Market Basket Analysis"
Demo: Item Item Similarity

Collaborative Filtering - User Based Filtering

Problems?

- Need to compute the similarity against every user.
- Doesn't scale up to millions of users.
- Computationally hard
- With many items, may be little overlap, making the similarity calculation hard

Collaborative Filtering - Item Based Filtering

The comparisons between items will not change as frequently as comparisons between users

So?

- Precompute and store the most similar items for each item

To make a recommendation for a user:

- Look at top rated items
- Aggregate similar items using precomputed similarities

These similarities will change with new ratings, but will change **slowly**

Demo: Precomputing Item Similarity

Collaborative Filtering - Item Based Filtering

To compute recommendations using this approach:

Estimate the rating for unrated item $\hat{i}$ that has a top-N similarity to a rated item i:

$$r_{u,\hat{i}} = \frac{\sum_{i \in I} sim(\hat{i}, i) r_{u,i}}{\sum_{i \in I} sim(\hat{i}, i)}$$

Where I is the subset of all N items similar to $\hat{i}$
Demo: Item based recommendation

Collaborative Filtering - Comparing Item and User based Filtering

User Based Filtering:

- Easier to implement
- No maintenance of comparisons
- Deals well with datasets that frequently change
- Deals well with small dense datasets

Item Based Filtering:

- Maintenance of comparison data necessary
- Deals well with small dense datasets
- Also deals well with larger sparse datasets
- Deals well with frequently changing users

Collaborative Filtering - Problems

Problems?

Collaborative Filtering - Problems

Problems?

The 'cold start' problem

Collaborative filtering will not work for a new user, or new item.

Collaborative Filtering - Solutions

For new items: Hybrid approach

- Use content based features to find similar items
- Bootstrap ratings for the new item by averaging the ratings users gave to similar items

Collaborative Filtering - Solutions

For new users: Harder

- Bootstrap user profile from 'cookies'
- Ask new users questions

Collaborative Filtering - Summary

Recommender systems are worth millions
Collaborative Filtering:

- Uses peoples behaviour to gather information
- Doesn't need content based features

Collaborative Filtering - Summary

Recommender systems are worth millions
Collaborative Filtering:

- Uses peoples behaviour to gather information
- Doesn't need content based features
  User based Neighbourhood approach:
  - Computes similarities between users
  - Predicts unseen item weights using ratings of similar users

Collaborative Filtering - Summary

Recommender systems are worth millions
Collaborative Filtering:

- Uses peoples behaviour to gather information
- Doesn't need content based features

  User based Neighbourhood approach:

  - Computes similarities between users
  - Predicts unseen item weights using ratings of similar users

  Item based Neighbourhood approach:

  - Precomputes similarities between items
  - Predicts unseen user ratings using ratings of similar items