Commerce on the web: Automated recommender systems

Based on Baldi et al.[*], Chapter 8

Juha Raitio
juha.raitio@iki.fi

26th November 2003

[*]P. Baldi, P. Frasconi and P. Smyth. Modeling the Internet and the Web. Wiley, 2003.

Outline

1. Challenging world of e-commerce
2. Customer data on the Web
3. Recommender problem
4. Evaluation of automated recommender systems
5. Recommenders based on collaborative filtering
6. Model-based collaborative filtering
7. Collaborative filtering utilising content
8. Summary

Commerce on the Web

- Rapid commercialisation of the Web presents new challenges
- Examples of challenges in e-commerce
  - Recommendations of new products to site-visitors based on their browsing behaviour
  - Modelling factors that influence how customers make purchases
  - Real time prediction of purchases based on navigation behaviour
- Vast amounts of raw data is available

Customer Data on the Web

- Data can be obtained at server or client side or points in between (proxies)
- Application specific logs have more complete information on user access
- Logs can be based on cookies and used for personalisation of web content
- Client-side recording with special software by a sample of web users can give full *panel data*

Market basket or vote data

- Assume large numbers n of individuals and m of items

- A vote matrix $\boldsymbol{V}$ $(n \times m)$ stores the votes for items per user,

  $v_{i,j} = 1$ indicates that individual i *purchased* or *voted* for item j, otherwise entry is left missing

- Missing values are indicated by zero,
  - $\boldsymbol{V}$ is typically very sparse
  - Missing values contain *negative selection bias*

- Vote matrix can also ratings that are non-binary
  - Rating can be achieved explicitly (e.g. scores given to movies) or implicitly (e.g. viewing time of a document)

Recommender problem

- Automatically recommend or rank a list of new items to a user
- A recommendation should consider items the user has already voted for, and past voting pattern of all users
- For the active user a, ℓ_a is the subset of total items ℓ user has already voted for
- Typically the number of remaining items needing a rank, those in $\ell \setminus \ell_a$, is quite large
- Ranking in real time is computationally challenging

Evaluating recommender systems

- Improvement achieved by a recommender can be measured against e.g. recommending a random or the most popular item
- Evaluation can be based on historical data or controlled experiments with users
- Any error function or statistic will do as a measure
- True value of a recommender system can only be found in controlled experiments with actual users

Recommenders based on nearest-neighbour collaborative filtering

- Basic algorithm
  1. Find the most similar users to a (neighbours)
  2. Combine their voting histories ℓ_i
  3. Predict votes for items $\ell \setminus \ell_a$
- Specific implementation for prediction of votes of user a

$$\hat{v}_{a,j} = v_a + \frac{1}{C}\sum_{i=1}^{n} w_{a,i} v^*_{i,j}, \qquad j \in \ell \setminus \ell_a, \qquad \text{where}$$

$v_i = \frac{1}{|\ell_i|}\sum_{j \in \ell_i} v_{i,j}$ is the mean vote for user i,
$v^*_{i,j} = v_{i,j} - v_i$ results in a mean adjusted vote matrix,
weights $w_{a,i}$ are based on how similar user i is to user a,
$C = \sum_{i=1}^{n} |w_{a,i}|$ is a normalisation constant

Defining weights in NN collaborative filtering

- Using correlation

$$w_{a,i} = \frac{1}{\sqrt{\sum_j (v_{a,j} - v_a)^2 \sum_j (v_{i,j} - v_i)^2}} \sum_j (v_{a,j} - v_a)(v_{i,j} - v_i)$$

- Normalised dot product

$$w_{a,i} = \sum_j \frac{v_{a,k}}{\sqrt{\sum_{k \in \ell_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{j \in \ell_i} v_{i,k}^2}}$$

- Weights may be thresholded to avoid summed effect of distant users

Reducing dimensionality of data

- Use correlation structure in items to find *meta-items*
  - e.g. latent semantic indexing (LSI)
- Use clustering to group users
  - e.g. K-means

Model-based collaborative filtering

- Build an explicit model of relationships among items based on historical voting patterns
- Prediction scales well (independent of n) as opposed to memory-based methods
- Basic approaches
  - Model full joint probability distribution with random variables v_j for items in ℓ i.e. $P(v_1, \ldots, v_m)$
  - Build m conditional models conditioned on $m - 1$ items i.e. $P(v_j | \{v_1, \ldots, v_m\} \setminus \{v_j\})$
  - Use models that produce directly ranked lists, rather than score items

Joint density modelling using finite mixtures

- For K mixtures the joint distribution

$$P(v_1, \ldots, v_m) = \sum_{k=1}^{K} P(v_1, \ldots, v_m | c = k) P(c = k), \quad \text{where}$$

$P(c = k)$ is the probability that a user belongs to component k,
$P(v_1, \ldots, v_m | c = k)$ is a probability of specific voting pattern in component k

- Assuming conditional independence (Naive Bayes model)

$$P(v_1, \ldots, v_m | c = k) = \prod_{j=1}^{K} P(v_j | c = k) = \prod_{j=1}^{K} \theta_{jk}^{v_j} (1 - \theta_{jk})^{1 - v_j}, \quad \text{where}$$

$\theta_{jk} = P(v_j | c = k)$

Properties of joint density models using finite mixtures

- Number of parameters in Naive Bayes model is Km compared to 2^m of full joint distribution model
- Model can be estimated using *EM-algorithm*
- Model ignores any dependencies items within a component
  - Alternative to Naive Bayes model for component model are e.g. Bayesian networks
- A limitation of mixture model is that it assumes each user was 'generated' by a single component model

Conditional distribution models

- m conditional models for $P(v_j|\{v_1,\ldots,v_m\}\setminus\{v_j\})$
- Each model can be implemented as a *probabilistic decision tree*
  - Binary trees are estimated from vote data recursively, always adding a node with item-value $(v_j = k)$ pair that maximises the probability of tree structure T given data V, $P(T|V)$
  - Performs almost as well as full Bayesian network models for joint distribution, but is faster in prediction

Recommending based on similarity of items

- Basic approach
  1. Model how similar items are to each other
  2. Recommend similar items
  - Generalises to new items
  - Misses purchase patterns
- Combining with vote data gives better results
  - An example model for joint distribution of users u, documents d and words w with topic z as a latent variable

$$P(u, d, w) \approx \sum_{z} P(u|z)P(d|z)P(w|z)P(z)$$

  - Estimated with *EM-algorithm*

Summary

- Real time recommendation based on large data sets is a typical challenge in e-commerce
- Client side recording can give panel type data on user preferences, application log files may give complete information regarding use of a single application
- Vote matrix stores votes or rating for items per user
- A recommendation should consider items the user has already voted for, and past voting pattern of all users
- True value of a recommender system can only be found in controlled experiments with actual users
- Recommenders can be based on nearest neighbour collaborative filtering

Summary (cont.)

- Proper reduction of dimensionality results in effient meta-items and users groups
- Building an explicit model of relationships among items scales well
- Joint density modelling using finite mixtures is economic in number of parameters used
- Probabilistic decision trees are an efficent way of building conditional models on items
- Extending recommendation to base on similarity of items generalises to new items

Problems

1. Construct an automated recommender system for movies.

   Use a reasonable size sample of the EachMovie data set to base your recommendations on. Utilise methods given in the presentation or any of your own ideas.

   Choose at random 50 movies you know and rank them from 0 to 5 stars (awful to brilliant). Test your recommender by five time cross validation leaving 10 out.

   Compute statistics on how well the recommender predicted your taste on movies.

   A sample of EachMovie data set will be e-mailed to the course participants. Refer to *http://www.research.compaq.com/SRC/eachmovie/* for more information on the data set and the terms of usage.