15s1: COMP9417 Machine Learning and Data Mining

Recommender Systems

May 27, 2015

Acknowledgement: Material derived from Adomavicius & Tuzhulin (2005) IEEE Trans. on Knowledge and Data Engineering, 17(6), 734–749.

Aims

This lecture will enable you to describe and reproduce machine learning approaches within the framework of Recommender Systems. Following it you should be able to:

- define the problem of recommender systems
- describe content-based, collaborative and hybrid recommender systems
- reproduce key similarity-based approaches to recommender systems

Introduction

- Recommender systems – a form of *personalization*
  - "person who liked x may also like y"
- related to instance-based learning
  - *similarity function*
- other forms of learning may be used to model user choices

A Framework for Recommendation

	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	Ø	4	5	5
Cindy	2	2	4	Ø
David	3	Ø	5	2

Example movie rating matrix, where each entry has user c rating item s.

Given: utility $u : c \times s \mapsto \mathcal{R}$

Problem: $\forall c \in C$, choose $s'_c = \operatorname{argmax}_{s \in S} u(c, s)$

This is learning in the sense of requiring *extrapolation* to predict the unknown values of the utility funcion.

Content-based Recommendation

User c is recommended items s that are *similar* to past choices.

- idea comes from *information retrieval*
- requires a profile of the *content* or description of items

$$u(c, s) = score(ContentBasedProfile(c), Content(s))$$

E.g.,

$$u(c, s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\|\vec{w}_c\| \times \|\vec{w}_s\|}$$

where

$\vec{w}_c$ is a vector of summarising terms of c's past choices, and

$\vec{w}_s$ is a vector of most relevant terms describing s

Content-based Recommendation

Advantages

- well-understood techniques from Information Retrieval
- can extract latent features from text analysis

Disadvantages

- may not have content, or may be limited or sparse
- over-specialisation: recommendations given for known types only
- new user problem: must do some rating to get recommendations

Collaborative-based Recommendation

User c is recommended items that users with *similar taste* have chosen.

- a.k.a. *collaborative filtering* (CF)
- Amazon-style recommender systems

Two main methods: *memory-based*, and *model-based* CF.

Memory-based CF

Predict unknown rating $r_{c,s}$ of user c for item s by aggregating the ratings of N users c' most similar to c who have rated s:

$$r_{c,s} = \mathrm{aggr}_{c' \in C} \;\; r_{c',s}$$

What aggregation to use ? One commonly used is weighted sum

$$r_{c,s} = k \sum_{c' \in C} \mathrm{sim}(c, c') \times r_{c',s}$$

where

k is just a normalising factor, and the similarity function can be correlation, cosine distance, etc. on the vector of items rated (e.g., bought) by users.

Alternatively, can use *item-based* similarity (Amazon).

Model-based CF

Memory-based CF is like a nearest-neighbour method.

A big problem is *sparsity* — to address this, often try to find a low-rank approximation to the matrix (i.e., finding smaller "user-feature" and "movie-feature" matrices) using a form of stochastic gradient descent.

However, can use other machine learning methods to build a model to predict directly the unknown rating $r_{c,s}$ from examples in the database.

E.g., Naive Bayes-type approaches.

This is called *model-based* CF.

Collaborative-based Recommendation

Advantages

- works well in practice
- does not require content (descriptions)

Disadvantages

- new user problem: must do some rating to get recommendations
- new item problem: must be rated to be used in recommendations
- "grey sheep": insufficiently individual !
- "black sheep": too individual !!

Hybrid Recommender Systems

Key idea: combine model-based and memory-based approaches

- "cold-start" problem: use model to predict before user activity
- "sparsity" problem: use model to predict missing values

But: learning models may be difficult or expensive

Summary

- based on techniques from information retrieval and machine learning
- an application area growing rapidly
- simple systems can do surprisingly well
- many possible extensions, e.g., recommendation in social networks