

Notes for Recommender Systems Handbook



1. Introduction

1.2. Recommender Systems functions

1. Increase the number of item sold: increase the conversion rate.
2. Sell more diverse items: suggest or advertises unpopular items to users.
3. Increase the user satisfaction: improve the user experience.
4. Increase user fidelity.
5. Better understand what user wants: can be leveraged to many other applications, is the description of the user's preferences, either collected explicitly or predicted by system.

1.3. Data and Knowledge source.

1. Items.

- complexity and values

low

news, Web pages
books, CDs, movies

high

digital cameras,
mobile phones,
PCs

insurance policy
financial investment
travels, jobs

- can be represented as a single id code, set of attributes, ontological representation.

2. Users.

- profiles users. e.g., demographic attributes, behavior pattern data.

3. Transactions

- interaction between user and RS

- ratings are most popular form of transaction data that RS collect.

1. numerical ratings ex. 1~5 star

2. ordinal ratings ex. strongly agree, agree, neutral, disagree, ...

3. binary ratings ex. good / bad

4. unary ratings ex. user has observed or purchased an item, or rated the item positively.

5. tags ex. in MovieLens users feel about a movie "too long", "acting"

1.4. Recommendation Techniques.

- a simple, non-personalized recommendation algorithm is recommend the most popular item.
- model the degree of utility of user u for the item i as a function $R(u,i)$, predict the value of R over pairs of users and items $\hat{R}(u,i)$, having computed prediction for user u on a set of items $\hat{R}(u,i_1), \dots, \hat{R}(u,i_N)$, the system will recommend the items i_1, \dots, i_K with largest \hat{R}_i , and K is typically much smaller than the number of the item.
- some recommender systems do not fully estimate the utility, e.g. knowledge-based system.
- taxonomy of recommendation approaches.

1. Content-based

- recommend items that are similar to the ones that user liked in the past.

2. Collaborative filtering

- most popular and widely implemented.
- recommend to user the items that other users with similar tastes liked in the past.
- similarity in taste: based on the similarity in the rating history.
- refers to people-to-people correlation.

3. Demographic

- recommends items based on the demographic profile of the user.
- relatively little research into demographic systems.

4. Knowledge-based

- recommends items based on specific domain knowledge.
- case-based: similarity function estimates how much the user needs match the recommendation (utility)
- constraint-based: predominately exploit predefined knowledge base that contain explicit rules about how to relate users and items.
- tends to work better than others at the beginning, but may be surpassed by other methods that can exploit the logs of human/computer interaction.

5. community-based. (social recommender systems)

- recommends items based on the preferences of the users friends.
- "Tell me who your friends are, and I will tell you who you are."

6. Hybrid.

- combination of the above techniques, use the advantages of A to fix the disadvantage of B.
- paradigms for incorporating contextual information into the recommendation process.

1. reduction-based (pre-filtering): only the information that matches the current usage context are used to compute the recommendation.
2. contextual post filtering: recommendation algorithm ignores the context information. the output the algorithm is filtered / adjusted to include only the recommendations that are relevant.
3. contextual modeling: context data is explicitly used in the prediction model.

1.5. Application and Evaluation.

- common applications:
 - Entertainment - movies, music, IPTV.
 - Content - news, documents, Web pages, e-learning applications, e-mail.
 - E-commerce - books, cameras, PCs
 - Services - travel services, experts for consultation, house to rent, matchmaking services.
- analysis of the available sources of knowledge → decide algorithm.
- evaluation is required at different stages of the system life cycle.
 - At design time: verify the selection of the appropriate recommender approach.
 - In the design phase: off-line evaluation consists of running several algorithms and comparing their performance. off-line evaluation should follow experiment design.

- After system launched : on-line evaluation. If on-line evaluation is not feasible or too risky, a controlled experiment can be planned.

1.b. Recommender Systems and Human Computer Interaction.

- 7 roles that can be played by explanation in RSs.
 1. transparency - explaining how the system works.
 2. scrutability - allowing users to tell the system it is wrong.
 3. trust - increasing user confidence in the system.
 4. effectiveness - helping users make good decisions.
 5. persuasiveness - convincing users to try or buy.
 6. efficiency - helping users make decision faster.
 7. satisfaction - increasing the ease of use and enjoyment.
- Conversational RSs.
 - critiquing-based interface (dialogues model) : given an initial set of user preferences, have attracted great interest in domain where there is a need for more sophisticated and interactive decision/recommendation.
 - preference-based.

Part I Basic Techniques

2. Data Mining Methods for Recommender Systems.

2.2. Data Preprocessing.

- similarity measures.

- Euclidean distance: $d(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$

- Minkowski distance: $d(x, y) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{\frac{1}{r}}$

r : degree of distance

$r=1$, city block / Manhattan / taxicab / L_1 -norm

$r=2$, euclidean distance

$r \rightarrow \infty$, supremum / L_{\max} -norm / L_∞ -norm

- Mahalanobis distance: $d(x, y) = \sqrt{(x-y)\Sigma^{-1}(x-y)^T}$

Σ : covariance matrix of data

- cosine similarity: $\cos(x, y) = \frac{(x \cdot y)}{\|x\| \|y\|}$

- Pearson correlation: $\text{Pearson}(x, y) = \frac{\Sigma(x, y)}{\sigma_x \sigma_y}$, Σ : covariance, σ : s.d.

- similarity measures (binary)

- simple matching coefficient

$$SMC = \frac{\# \text{ of matches}}{\# \text{ of attributes}} = \frac{M_{00} + M_{11}}{M_{01} + M_{10} + M_{00} + M_{11}}$$

- Jaccard coefficient $J_C = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$

- Extended Jaccard (Tanimoto) coefficient

$$d = \frac{x \cdot y}{\|x\|^2 + \|y\|^2 - x \cdot y}$$

- Reducing Dimensionality.

It is common in RS to have a dataset with features that defines a high dimensional space with very sparse information.

- Principal Component Analysis (PCA)

PCA allows to obtain an ordered list of components that account

	$y=0$	$y=1$
$x=0$	M_{00}	M_{01}
$x=1$	M_{10}	M_{11}

for the largest amount of the variance.

PCA allows us to retrieve the original data matrix by projecting the data onto the new coordinate system: $X'_{n \times m'} = X_{n \times m} W_{m \times m'}$.

The new data matrix X' contains most of the information of the original X with a dimensionality reduction of $m-m'$.

limitations: 1. empirical dataset should be a linear combination of a certain basis.

2. original dataset has been drawn from a Gaussian distribution.

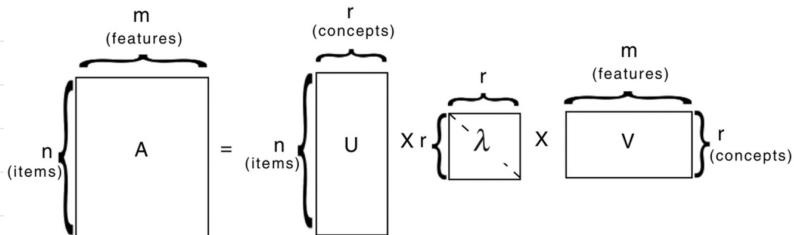
- **Singular Value Decomposition (SVD)**

SVD is to find a lower dimensional feature space where the new features represent "concept" and each concept is computable.

It can be used as the basis of latent-semantic analysis.

It is always possible to decompose a given matrix A into

$$A = U \lambda V^T$$



- U : item-to-concept.

- λ : strength of each concept.

a diagonal matrix contains the singular values, which will always be positive and sorted in decreasing order.

- V : term-to-concept.

The original matrix can be approximated by simply truncating the eigenvalue at a given k . $A_k = U_k \lambda_k V_k^T$. A_k is the closest rank- k matrix to A , "closest" means that A_k minimizes the sum of the squares of A_k and A .

- **Denoising**

1. natural noise: unvoluntarily introduced by users.

2. malicious noise: deliberately introduced in a system in order to bias the results.

2.3. Classification.

• Nearest Neighbors. (kNN)

kNN classifier finds the **k closest points** (nearest neighbors) from the training records, it then assigns the class label according to the class labels of its nearest neighbors.

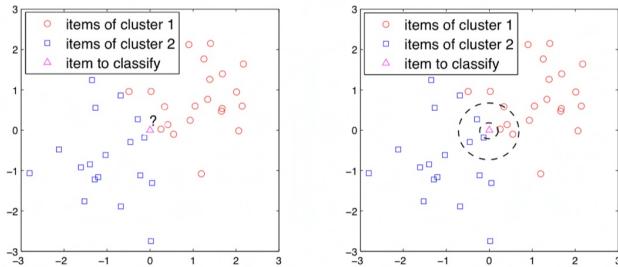


Fig. 2.4: Example of k-Nearest Neighbors. The left subfigure shows the training points with two class labels (circles and squares) and the query point (as a triangle). The right sub-figure illustrates closest neighborhood for $k = 1$ and $k = 7$. The query point would be classified as square for $k = 1$, and as a circle for $k = 5$ according to the simple majority vote rule. Note that the query points was just on the boundary between the two clusters.

challenging: How to choose the value of k ?

- k too small: classifier will be sensitive to noise point.
- k too large: neighborhood might include too many points from other classes.

• Decision Trees.

advantages:

- inexpensive to construct.
- extremely fast.
- can be used to produce a set of rules that are easy to interpret.

• usage in RS:

1. used as a filter to select which users should be targeted with recommendations.
2. used as a tool for item ranking.

• Rule-based Classifiers.

- a collection of "if...then..." rules.

definition

- **cover**: if the attributes of the instance satisfy the rule condition.
- **coverage**: fraction of records that satisfy its antecedent.

- **accuracy**: fraction of records that satisfy both the antecedent and the consequent.
 - **mutually exclusive rules**: rules are independent of each other.
every record is covered by at most one rule.
 - **exhaustive rules**: each record is covered by at least one rule.
- advantages**:
- easy to interpret.
 - easy to generate.
 - classify new instances efficiently.

• Bayesian Classifier

- based on **conditional probability** and **Bayes theorem**.
 - **posterior** \propto **likelihood** \times **prior**
- likelihood: effect of data
prior: belief in the model before the data was observed.
posterior: probability of a model given the data.
- Given N attributes (A_1, A_2, \dots, A_N), the goal is to predict class C_k by finding the value of C_k that maximizes the posterior of the class given the data $P(C_k | A_1, A_2, \dots, A_N)$. Applying Bayes' theorem,
 $P(C_k | A_1, A_2, \dots, A_N) \propto P(A_1, A_2, \dots, A_N | C_k) P(C_k)$

• Naive Bayes Classifier

- assumes the **probabilistic independence** of the attributes.

$$\Rightarrow P(A_1, A_2, \dots, A_N | C_k) = P(A_1 | C_k) P(A_2 | C_k) \dots P(A_N | C_k)$$

- advantage**:
- robust to isolated noise points and irrelevant attributes.
 - can handle missing values by ignoring the instance during probability estimation.

disadvantage: independence assumption is too strong.

- often used for **content-based RS**.
- **Artificial Neural Network**.

advantage: can perform non-linear classification task.

disadvantage: ANN belongs to sub-symbolic classifier, they provide no semantics for inferring knowledge.

- Support Vector Machines (SVM)

- find a linear hyperplane (decision boundary) that separates the data in such way that the margin is maximized.

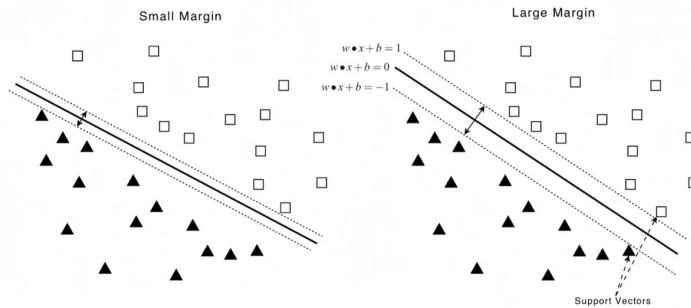


Fig. 2.6: Different boundary decisions are possible to separate two classes in two dimensions. Each boundary has an associated margin.

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b \geq 1 \\ -1, & \text{if } w \cdot x + b < -1 \end{cases} \Rightarrow \text{Constrained optimization problem.}$$

$$\text{margin} = \frac{2}{\|w\|^2}$$

- If not linear separable, introduce slack variable for soft margin, or use kernel trick.
- soft margin

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b \geq 1 - \varepsilon \\ -1, & \text{if } w \cdot x + b \leq -1 + \varepsilon \end{cases}$$

$$\text{minimize } L(w) = \frac{\|w\|^2}{2} + C \sum_{i=1}^N \varepsilon_i \in \text{slack variable.}$$

- Ensembles of classifiers.

- construct a set of classifiers \rightarrow aggregating their predictions.
- Bagging
 - perform sampling with replacement, build classifier on each bootstrap sample.
 - each sample has probability $(1 - \frac{1}{N})^N$ of being selected.
if $N \rightarrow \infty$, $(1 - \frac{1}{N})^N \rightarrow 0.623$.
- Boosting
 - iterative procedure

- adaptively change distribution of training data
 - focusing more on previously misclassified records.
- Evaluating Classifiers.
- focus only on classification performance.

		Actual	
		Neg	Pos
pred	Neg	TN	FN
	Pos	FP	TP

- Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$

- Precision: $\frac{TP}{TP+FP}$

- Recall: $\frac{TP}{TP+FN}$

- F1-score: $\frac{2RP}{R+P} = \frac{2TP}{2TP+FN+FP}$

- Receiver Operating Characteristic Curve (ROC curve)

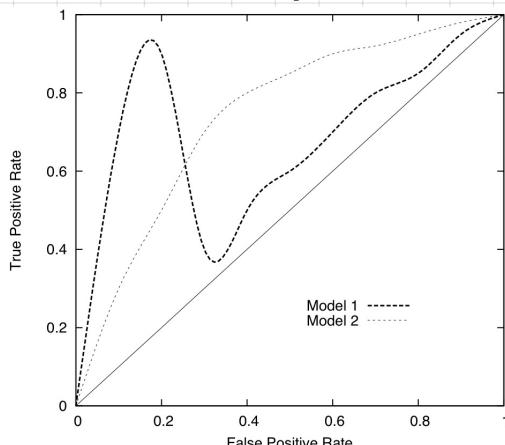


Fig. 2.7: Example of ROC Curve. Model 1 performs better for low False Positive Rates while Model 2 is fairly consistent throughout and outperforms Model 1 for False Positive Rates higher than 0.25

2.4 Cluster Analysis.

- assigning items into groups, items in the same groups are more similar than items in different groups.
- minimize intra-cluster distances.
- maximize inter-cluster distances.
- { partitional : divide data into non-overlapping clusters.
hierarchical: producing nested cluster organized as a hierarchical tree.
- **k-Means**
 - belongs to partitioning method.

- iterative process to minimize $E = \sum_{j=1}^k \sum_{n \in S_j} d(x_n, \lambda_j)$

x_n : vector representation of n-th item.

λ_j : centroid of the item in S_j

d : distance measure.

step 1: randomly selecting k centroids.

step 2: all items are assigned to the cluster whose centroid is the closest to them.

step 3: update new cluster centroids.

step 4: repeat 2,3 until there are no further items that change cluster.
stopping condition is often change to "until relatively few points change clusters" to improve convergence speed.

advantage: extremely simple and efficient.

disadvantage: (1) how to choose appropriate k.

(2) sensitive to the initial centroids.

(3) may produce empty cluster.

(4) can not handle clusters with differing size, density
and non-globular shape.

(5) not robust to outlier.

- Density-based clustering.

DBSCAN

density: number of points within a specified radius

(1) core point: more than a specified number of neighbors within a given distance.

(2) border point: fewer than a specified number of neighbors but belong to core points' neighbor.

(3) noise point: neither core or border.

iteratively removes noise points and perform clustering.

- Message-passing clustering.

- graph-based clustering methods.

- Affinity Propagation

- Hierarchical Clustering.
 - does not have to assume number of cluster in advance.
 - agglomerative: start with individual point as per cluster, then merge the closest pair of cluster.
 - divisive: start with one, all-inclusive cluster, then split cluster.

2.5 Association Rule Mining.

- means co-occurrence but not causality.

definition:

- (1) **itemset**: collection of one or more items. e.g. (Milk, Beer, Diaper)
- (2) **k-itemset**: itemset that contains k items.
- (3) **support count**: frequency of an itemset.
e.g. (Milk, Beer, Diaper) = 131.
- (4) **support**: the fraction of transactions that contain a itemset
e.g. (Milk, Beer, Diaper) = 0.12
- (5) **minsup**: minimum support, a threshold.
- (6) **frequent itemset**: itemset with support $\geq \text{minsup}$.
- (7) **rule**: expression of the form $X \Rightarrow Y$, X, Y are itemsets.
- (8) **support of rule**: the fraction of transactions that both have X and Y.
- (9) **confidence**: how often items in Y appear in transaction that contain X.

- Given a set of transaction T, finding all rules that:

- (1) **support $\geq \text{minsup}$**
- (2) **confidence $\geq \text{minconf}$**

- brute-force approach: list all possible association rules, and prune rules that do not satisfy conditions.
 ⇒ **computationally expensive !!**

- take two-step approach:

- (1) **Frequent Itemset Generation**

generate all itemsets whose support $\geq \text{minsup}$

- (2) **Rule Generation**

generate high confidence rules from frequent itemsets.

- Apriori Principle.

- use to reduce the number of candidate.
- if an itemset is frequent \Rightarrow all subsets are frequent
- the support of an itemset never exceeds that of its subset.

3. Content-based Recommender System:

State of the Art and Trends.

3.1. Introduction

content-based recommendation: recommend items similar to those a given user has liked in the past.

3.2. Basics of Content-based Recommender System.

- High-level Architecture of Content-based Recommender Systems.

Content Analyzer

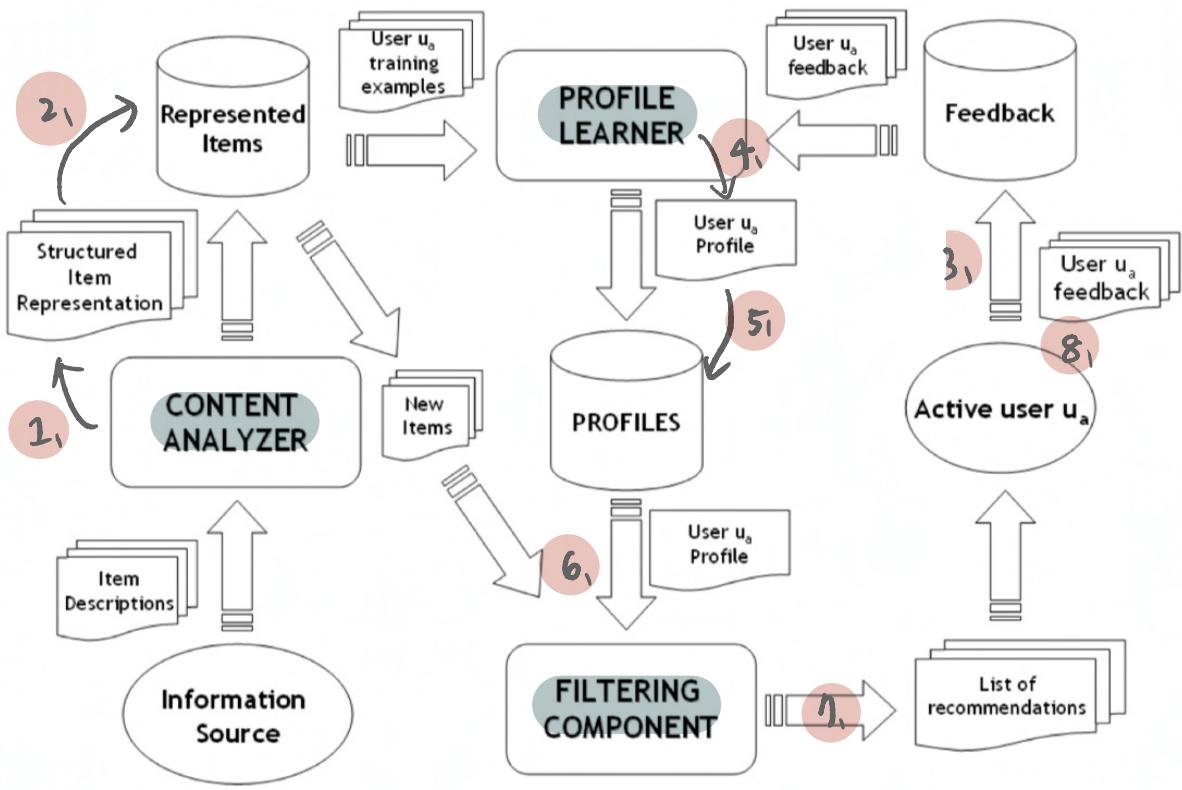
- extract structured relevant information
- represent the content of items.
- representation is the input to the Profile Learner and Filtering Component.

Profile Learner

- represent the user preferences.
- infer a model of user interests starting from items liked or disliked in the past.

Filtering Component

- suggest relevant items by matching the profile representation against that of items to be recommended.
- the result is a binary or continuous relevance judgment based on some similarity metrics.



1. Content Analyzer extracts features from unstructured item information to structured item representation.
2. structured item representation stored in Represented Item.
3. store active users' reaction into Feedback.
4. Profile Learner generate supervised model: user profile by using training set $TR_a = \langle I_k, r_k \rangle$ for user u_a , where r_k is the rating on item representation I_k by user u_a . Given item representation I , user profile can predict responding rating r .
5. user profile is stored in profile repository.
6. Given a new item, Filtering Component predicts the interest for the active user,
7. Top-ranked items are included in a list of recommendations.
8. users' feedback on the list of recommendations are gathered. and are used to update user profile.

Advantages:

- (1) User Independence: build user profile for each active user exploit solely their own ratings.

- (2) **Transparancy**: explanations can be provided by listing content features or descriptions that caused an item to occur in the list.
- (3) **New Item**: capable of recommending items not yet rated by any user, contrast to collaborative filtering relies on users' preference.

Disadvantage

- (1) **Limited Content Analysis**: domain knowledge often needed to extract useful features to discriminate the user like a item or not. useful feature might be insufficient.
- (2) **Over-specialization**: content-based system tends to produce recommendations with a limited degree of novelty, this drawback also called **serendipity problem**.
- (3) **New user**: enough ratings have to be collected to train user profile.

3.3. State of the Art of the Content-based Recommender Systems.

- Item Representation.
 - Keyword-based Vector Space Model

Vector Space Model (VSM)

VSM is a spatial representation of text document, each document is represented by a vector in a n-dimension space, where each dimension corresponds to a term.

Formally, every document is represented as a vector of term weights, where each weight indicates the degree of association between document and term.

$D = \{d_1, d_2, \dots, d_N\}$: set of documents or corpus.

$T = \{t_1, t_2, \dots, t_n\}$: set of words in the corpus.

$d_j = \{w_{1j}, w_{2j}, \dots, w_{nj}\}$: vector representation of document d_j .
 where w_{kj} is the weight for term t_k in document d_j .

2 issues in VSM: (1) weighting the term.

(2) measuring the feature vector similarity.

most commonly used term weighting scheme:

TF-IDF (Term Frequency - Inverse Document Frequency)

based on empirical observations:

Terms that occur frequently in one document (TF), but rarely in the rest of the corpus (IDF), are more likely to be relevant to the topic of the document.

In addition, normalizing the resulting weight prevent longer document from having a better chance of retrieval.

$$\text{TF-IDF}(t_k, d_j) = \underbrace{\text{TF}(t_k, d_j)}_{\text{TF}} \cdot \underbrace{\log \frac{N}{n_k}}_{\text{IDF}}, \quad \text{TF}(t_k, d_j) = \frac{f_{kj}}{\max_z f_{zj}}$$

N : number of documents

n_k : number of documents the the term t_k occurs at least once.

In order for the weights to fall in $[0, 1]$ and for the documents to be represented by vectors of equal length, weights are usually normalized by cosine normalization:

$$w_{kj} = \frac{\text{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^S \text{TF-IDF}(t_s, d_j)^2}}$$

and cosine similarity is the most widely used to determine the closeness between two documents:

$$\text{sim}(d_i, d_j) = \frac{\sum_k w_{ki} w_{kj}}{\sqrt{\sum_k w_{ki}^2} \sqrt{\sum_k w_{kj}^2}}$$

- Methods for Learning User Profiles.

The problem of learning user profiles can be cast as a **binary classification task**, each document has to be classified as

interesting or not.

1. Probabilistic Method and Naive Bayes.

The model estimates the posterior probability of document d belonging to class c : $P(c|d)$

By Bayes Theorem:

$$P(c|d) = \frac{P(c) P(d|c)}{P(d)}$$

To classify document d , the class with highest posterior probability is chosen:

$$c = \operatorname{argmax}_{c_j} \frac{P(c_j) P(d|c_j)}{P(d)}$$

→ estimated by training data.
→ generally removed as it is equal for all c_j .

but estimating $P(d|c)$ is problematic, as it is very unlikely to see the same document more than once

⇒ Naive Bayes classifier overcomes this problem by simplifying the model through conditional independent assumption.

$$P(d_i, c_j) = \prod_{t_k \in V_{d_i}} P(t_k|c_j)^{N(d_i, t_k)}$$

$N(d_i, t_k)$: number of times token t_k appeared in document d_i .

V_{d_i} : subset of tokens that appear in the document d_i .

To estimate $P(t_k|c_j)$ more robust to infrequent words, smoothing methods such as Laplace estimate (adding one to all the word counts for a class) and Witten-Bell are applied instead of simple event count.

2. Relevance Feedback and Rocchio's Algorithm.

4. A Comprehensive Survey of Neighborhood-based Recommendation Methods

4.1. Introduction.

The recommendation problem can be defined as estimating the response of a user for new items.

The type of user-item response might be:

- (1) scalar: ratings, are numerical (e.g., 1~5 stars) or ordinal
- (2) binary: e.g., like or dislike / interested or not interested.
- (3) unary: e.g., purchase / online access.

• Formal Definition

- (1) U : set of users
- (2) I : set of items
- (3) R : set of ratings
- (4) S : set of possible values for a rating
(e.g., $S = [1, 5]$ or $S = \{\text{like}, \text{dislike}\}$)
- (5) r_{ui} : rating of user u to item i
(only one rating can be made by any user for particular item)
- (6) U_i : subset of users that have rated an item i
- (7) I_u : subset of items that have been rated by user u .
- (8) I_{uv} : items that have been rated by two users u and v , i.e., $I_u \cap I_v$
- (9) U_{ij} : set of users that have rated both item i and j .

• best item recommendation problem

for a particular user u , finding the new item $i \in I \setminus I_u$ for which u is most likely to be interested in.

When ratings are available \Rightarrow regression or multi-class classification.

The goal is learn a function $f: U \times I \rightarrow S$ that predicts the rating $f(u, i)$ of a user u for a new item i .

recommend to the active user u_a an item i^* for which the estimated rating has the highest value: $i^* = \operatorname{argmax}_{j \in I \setminus I_u} f(u_a, j)$

The ratings R are divided into a training set R_{train} and a test set R_{test} .

Two popular measures of accuracy are:

(1) MAE (Mean Absolute Error)

$$\text{MAE}(f) = \frac{1}{|R_{\text{test}}|} \sum_{r_{ui} \in R_{\text{test}}} |f(u_i) - r_{ui}|$$

(2) RMSE (Root Mean Square Error)

$$\text{RMSE}(f) = \sqrt{\frac{1}{|R_{\text{test}}|} \sum_{r_{ui} \in R_{\text{test}}} (f(u_i) - r_{ui})^2}$$

When ratings are not available

→ recommend to an active user u a list $L(u)$ containing N items.

The item set I are divided into I_{train} and I_{test} .

Let $T(u) \subset I_u \cap I_{\text{test}}$ be the subset of test items that a user u found relevant.
Performance can be evaluated by:

(1) Precision

$$\text{Precision}(L) = \frac{1}{|L|} \sum_{u \in U} \frac{|L(u) \cap T(u)|}{|L(u)|}$$

(2) Recall

$$\text{Recall}(L) = \frac{1}{|U|} \sum_{u \in U} \frac{|L(u) \cap T(u)|}{|T(u)|}$$

This task consider all items in $L(u)$ equally interesting to user u ,
an alternative setting consists in learning $L(u)$ where items are
ordered by their interestingness to u .

For each user u , a single item i_u of I_u , the performance of L can
be evaluated by Average Reciprocal Hit-Rank (ARHR):

$$\text{ARHR}(L) = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\text{rank}(i_u, L(u))}, \text{ where } \text{rank}(i_u, L(u)) = \begin{cases} \infty, & \text{if } i_u \notin L(u) \\ \text{rank of } i_u \text{ in } L(u) & \end{cases}$$

- Overview of Recommendation Approaches.

Content-based Approaches

- general principle: identify the common characteristic of items that have received a favorable rating from a user, and then recommend the new item that share these characteristics to that user.

- content-based generally suffer from:

1, **limited content analysis**: system may have only a limited amount of information on its users or the content of its items. Some reasons for lack of information might be:

- (1) **privacy issue**: user might refuse to provide personal information.
- (2) **precise content of item** may be difficult or costly to obtain.

2, **over-specialization**: the predicted rating of a user for an item is high if this item is similar to the ones liked by this user.

Collaborative filtering approaches

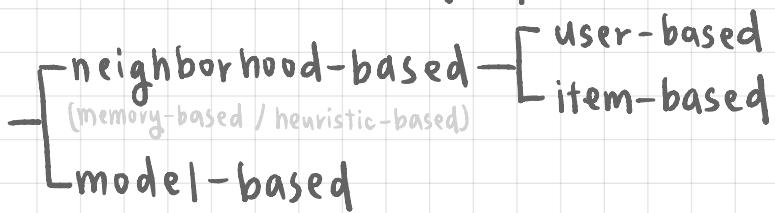
• key idea: rating of u for a new item i is likely to be similar to that of another user v , if u and v have rated other items in a similar way.

• overcome some limitations of content-based ones:

(1) **limited content analysis**: items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users.

(2) **over-specialization**: CF can recommend items with very different content, as long as other users have already shown interest for these different items.

• CF methods can be grouped in:



User-based

evaluate the interest of a user u for an item i using the ratings for this item by u 's neighbors that have similar rating pattern to u .

Item-based

predict the rating of a user u for an item i based on the ratings of u for items similar to i , two items are similar if several

users have rated these items in a similar fashion.

Model-based

model the user-item interactions with latent characteristic of the users and the items. model-based approaches including Bayesian Clustering, Latent Semantic Analysis, Latent Dirichlet Allocation, Maximum Entropy, Boltzmann Machine, SVM and SVD.

- Advantages of Neighborhood Approaches.

- (1) Simplicity: intuitive and relatively simple to implement.
- (2) Justifiability: provide a concise and intuitive justification for the recommendation.
- (3) Efficiency: no costly training phase unlike model-based approaches.
the nearest-neighbors can be pre-computed offline.
- (4) Stability: little affected by the constant addition of users, items, and ratings