Model-based Collaborative Filtering

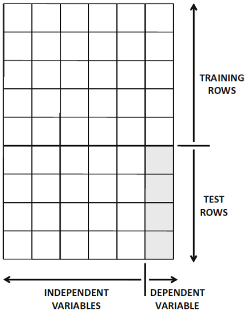
Neighborhood-based approaches for collaborative filtering are specific to the user and item that is being predicted. In contrast, **model-based collaborative filtering** relies on a summarized model of the data that is created beforehand during a training phase. This model is then used in the later prediction phase to efficiently calculate the predictions.

Advantages of model-based approaches[1]:

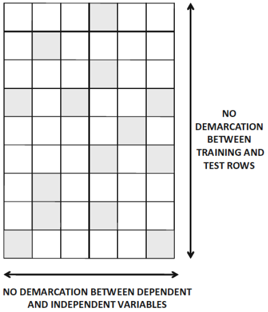
- Space-efficiency: Typically, the size of the learned model is much smaller than the original ratings matrix.
- Training speed and prediction speed: One problem with neighborhood-based methods is that the pre-processing stage is quadratic in either the number of users or the number of items. Model-based systems are usually much faster in the preprocessing phase of constructing the trained model.
- Avoiding overfitting: Overfitting is a serious problem in many machine learning algorithms, in which the prediction is overly influenced by random artifacts in the data. The summarization approach of model-based methods can often help in avoiding overfitting.

[1] Adopted from: Recommender Systems The Textbook, C. C. Aggarwal, Springer, 2016

Model-based Collaborative Filtering



(a) Classification



(b) Collaborative filtering

Comparing the traditional classification problem with collaborative filtering. Shaded entries are missing and need to be predicted[1]

Model-based collaborative filtering methods include, for example, rule-based methods, decision trees, regression models, Bayes classifiers, support vector machines, or neural networks.

[1]Source: Recommender Systems The Textbook, C. C. Aggarwal, Springer, 2016

Association Rule Mining

Association Rule Mining is a technique to identify rulelike relationship patterns in large-scale transactions.

- An example rule could be: "If a customer purchases baby food then he or she also buys diapers in 70 percent of the cases".

Definition by Sarwar et al.:

- A (sales) transaction T is a subset of the set of available products $P = \{p_1, \ldots, p_m\}$ and describes a set of products that have been purchased together.
- Association rules are often written in the form $X \rightarrow Y$, with X and Y being both subsets of P and $X \cap Y = \emptyset$
- An association rule $X \rightarrow Y$ (e.g. baby food $\rightarrow$ diapers) expresses that whenever the elements of X (the rule body) are contained in transaction T, it is very likely that the elements in Y (the rule head) are elements of the same transaction.

In collaborative recommender systems, the goal of association rule mining is to detect rules such as *"If user Alice liked both Item* 5 *and Item* 7*, then Alice will most probably also like Item* 23*"*.

Association Rule Mining

The goal of rule mining algorithms such as *Apriori* (by Agrawal and Srikant, 1994) is to automatically detect such rules and *calculate a measure of quality for those rules*[1].

- The **support** of a rule $X \rightarrow Y$ is calculated as the percentage of transactions that contain all items of $X \cup Y$ with respect to the number of overall transactions (i.e., the probability of co-occurrence of X and Y in a transaction):

$$\text{Support}(X \rightarrow Y) = \frac{\text{Number of transactions containing } X \cup Y}{\text{Number of transactions}}$$

- The **confidence** is defined as the ratio of transactions that contain all items of $X \cup Y$ to the number of transactions that contain only X:

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Number of transactions containing } X \cup Y}{\text{Number of transactions containing } X}$$

[1] Adapted from: D. Jannach et al.: Recommender Systems An Introduction, Cambridge University Press, 2011

Association Rule Mining: Example

Assume a ratings matrix with a binary scale ("like" / "dislike"):

	Bread	Butter	Milk	Fish	Beef	Ham
User 1	1	1	1	0	0	0
User 2	0	1	1	0	1	0
User 3	1	1	0	0	0	0
User 4	1	1	1	1	1	1
User 5	0	0	0	1	0	1
User 6	0	0	0	1	1	1
User 7	0	1	0	1	1	0

It is evident that the columns of the table can be partitioned into two sets of closely related items. One of these sets is $\{Bread, Butter, Milk\}$, and the other set is $\{Fish, Beef, Ham\}$. These are the only itemsets with at least 3 items, which also have a support of at least 0.2.

Association Rule Mining

Finding association rules involves two steps:

1. Find all itemsets that satisfy a minimum support threshold s.
2. From each of these itemsets Z, all possible 2-way partitions $(X, Z - X)$ are used to create a potential rule $X \rightarrow Z - X$. Those rules satisfying the minimum confidence are retained.

The calculation of interesting association rules can be performed offline.

- At runtime, the following scheme can be used to compute recommendations for a user Alice:
  - Determine the set of $X \rightarrow Y$ association rules that are relevant for Alice that is, where Alice has bought (or liked) all elements from X.
  - Compute the union of items appearing in the consequent Y of those association rules that have not been purchased by Alice.
  - Sort the products according to the confidence of the rule that predicted them.
  - Return the first N elements of this ordered list as a recommendation.

Matrix factorization / Singular Value Decomposition

- Simply put, matrix factorization methods can be used in recommender systems to derive a set of **latent** (hidden) factors from the rating patterns and characterize both users and items by such vectors of factors.
- **Singular Value Decomposition (SVD)** was proposed by Deerwester et al. in 1990 as a method to discover latent factors in documents.
- The SVD theorem by Golub and Kahan (1965) states that a given matrix $\mathbf{M}$ can be decomposed into a product of three matrices as follows:

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}$$

where $\mathbf{U}$ and $\mathbf{V}$ are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*.

Singular Value Decomposition: Example

Consider the following rating matrix:

	Item 1	Item 2	Item 3	Item 4
User 1	3	1	2	3
User 2	4	3	4	3
User 3	3	2	1	5
User 4	1	6	5	2

Because in this example the 4x4 matrix $\mathbf{M}$ is quadratic, $\mathbf{U}, \Sigma$ and $\mathbf{V}$ are also quadratic.

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}$$

$$\mathbf{U} = \begin{pmatrix} -0.35 & -0.36 & 0.29 & -0.80 \\ -0.56 & -0.08 & 0.62 & 0.52 \\ -0.44 & -0.56 & -0.65 & 0.21 \\ -0.59 & -0.73 & -0.28 & -0.17 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 12.22 & 0 & 0 & 0 \\ 0 & 4.92 & 0 & 0 \\ 0 & 0 & 2.06 & 0 \\ 0 & 0 & 0 & 0.29 \end{pmatrix}$$

$$\mathbf{V}^T = \begin{pmatrix} -0.43 & -0.53 & -0.52 & -0.50 \\ -0.49 & 0.53 & 0.40 & -0.55 \\ 0.55 & -0.41 & 0.48 & -0.53 \\ 0.51 & 0.50 & -0.56 & -0.38 \end{pmatrix}$$

Singular Value Decomposition: Example

The main point of the SVD is that we can approximate the full matrix by observing only the most important features, i.e., those with the largest singular values.

- For example, consider the projection of $\mathbf{U}$, $\mathbf{V^T}$ and Σ in the two-dimensional space:

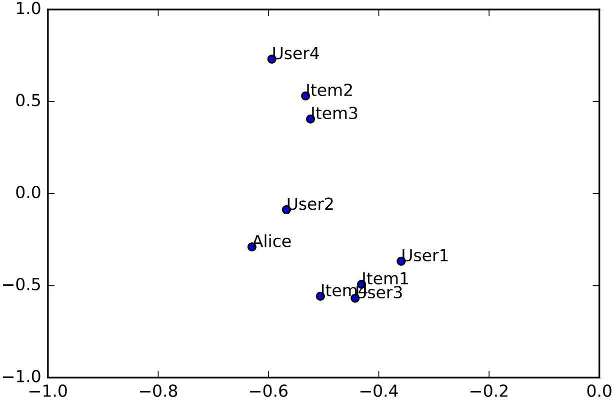
$$\mathbf{U}_2 = \begin{pmatrix} -0.35 & -0.36 \\ -0.56 & -0.08 \\ -0.44 & -0.56 \\ -0.59 & -0.73 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 12.22 & 0 \\ 0 & 4.92 \end{pmatrix} \qquad \mathbf{V}_2^T = \begin{pmatrix} -0.43 & -0.53 & -0.52 & -0.50 \\ -0.49 & 0.53 & 0.40 & -0.55 \end{pmatrix}$$

- Matrix $\mathbf{U}$ and $\mathbf{V}$ correspond to the **latent user and item factors**.
- Although in this example we cannot observe any clusters of users, we can see that the items from $\mathbf{V}$ build two groups.

Singular Value Decomposition: Example

Consider a target user *Alice* with a rating vector $r_{Alice} = [5, 3, 4, 4]$ for the four items. We can find out where Alice would be positioned in this two-dimensional space by calculating

$$r_{Alice2D} = r_{Alice} \times \mathbf{V}_2 \times \Sigma_2^{-1} \approx [-0.63, 0.29]$$



Naive Bayes Collaborative Filtering

Naive Bayes is a generative model, which is commonly used for classification.

- We treat items as features and users as instances.
- The main challenge is that any feature (item) can be the target class, and we have to deal with incomplete feature variables.

Assume a small number of l distinct ratings $v_1, \ldots, v_l$ such as *like, neutral, dislike*, and an $m \times n$ rating matrix where the (u, j)th element is denoted by r_{uj}. Furthermore, let I_u be the set of items that have been rated by user u.
The goal of the Bayes classifier is to predict the unobserved rating r_{uj}.

- r_{uj} can take any one of the discrete possibilities in $\{v_1, \ldots, v_n\}$.
- We would like to determine the **probability** that r_{uj} takes on any of these values **conditional on the observed ratings in I_u**.

In other words, we want to determine the **probability** $P(r_{uj} = v_s | \text{Observed ratings in } I_u)$ for each value of s in $\{1, \ldots, l\}$.

Naive Bayes Collaborative Filtering

We can simplify this expression by using the well-known **Bayes theorem**:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Therefore for each value of s in $\{1, \ldots, l\}$ we have

$$P(r_{uj} = v_s|\text{Observed ratings in } I_u) = \frac{P(r_{uj} = v_s) * P(\text{Observed ratings in } I_u|r_{uj} = v_s)}{P(\text{Observed ratings in } I_u)}$$

- We would like to determine the value of s for which the value of $P(r_{uj} = v_s|\text{Observed ratings in } I_u)$ is as large as possible.
- Because the denominator of the right hand side is independent of s, we can express the equation in terms of a constant of proportionality:

$$P(r_{uj} = v_s|\text{Observed ratings in } I_u) \propto P(r_{uj} = v_s) * P(\text{Observed ratings in } I_u|r_{uj} = v_s)$$

Naive Bayes Collaborative Filtering

- The value of $P(r_{uj} = v_s)$, also called the **prior probability**, is estimated to the fraction of users that have specified the rating v_s for item j (users who haven't rated j are ignored here).
- $P(\text{Observed ratings in } I_u | r_{uj} = v_s)$ is estimated with the use of the **naive assumption** (i.e., conditional independence between ratings). Conditional independence says that the ratings of a user for various items I_u are independent from each other, *conditional* of the fact that the value of r_{uj} was observed to be v_s. We can express this mathematically as follows:

$$P(\text{Observed ratings in } I_u | r_{uj} = v_s) = \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)$$

- $P(r_{uk} | r_{uj} = v_s)$ is estimated as the fraction of users that have specified the rating of r_{uk} for the kth item, *given that they have specified the rating of their jth item to v_s*.

Naive Bayes Collaborative Filtering

By plugging in the estimation of the prior probability $P(r_{uj} = v_s)$ and the previous equation, it is possible to obtain an estimate of the **posterior probability** of the rating of item j for user u as follows:

$$P(r_{uj} = v_s | \text{Observed ratings in } I_u) \propto P(r_{uj} = v_s) * \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)$$

By computing each of the expressions on the right-hand side for each $s \in \{1, \ldots, l\}$, and determining the value of s at which it is the largest, we can determine the most likely value $\hat{r}_{uj}$ of the missing rating r_{uj}:

$$\hat{r}_{uj} = \text{argmax}_{v_s} P(r_{uj} = v_s) * \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)$$