3.4 Naive Bayes Collaborative Filtering

In the following, we will assume that there are a small number of distinct ratings, each of which can be treated as a categorical value. Therefore, the orderings among the ratings will be ignored in the following discussion. For example, three ratings, such as *Like*, *Neutral*, and *Dislike*, will be treated as unordered discrete values. In the case where the number of distinct ratings is small, such an approximation can be reasonably used without significant loss in accuracy.

Assume that there are l distinct values of the ratings, which are denoted by $v_1 \ldots v_l$. As in the case of the other models discussed in this chapter, we assume that we have an $m \times n$ matrix R containing the ratings of m users for n items. The (u, j)th entry of the matrix is denoted by r_{uj}.

The naive Bayes model is a generative model, which is commonly used for classification. One can treat the items as features and users as instances in order to infer the missing entries with a classification model. The main challenge in using this approach for collaborative filtering is that any feature (item) can be the target class in collaborative filtering, and one also has to work with incomplete feature variables. These differences can be handled with minor modifications to the basic methodology of the naive Bayes model.

Consider the uth user, who has specified ratings for the set of items I_u. In other words, if the uth row has specified ratings for the first, third, and fifth columns, then we have $I_i = \{1, 3, 5\}$. Consider the case where the Bayes classifier needs to predict the unobserved rating r_{uj} of user u for item j. Note that r_{uj} can take on any one of the discrete possibilities in $\{v_1 \ldots v_l\}$. Therefore, we would like to determine the probability that r_{uj} takes on any of these values *conditional on the observed ratings in* I_u. Therefore, for each value of $s \in \{1 \ldots l\}$, we would like to determine the probability $P(r_{uj} = v_s | \textit{Observed ratings in } I_u)$. This expression appears in the form $P(A|B)$, where A and B are events corresponding to the value of r_{uj}, and the values of the observed ratings in I_u, respectively. The expression can be simplified using the well-known Bayes rule in probability theory:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)} \tag{3.3}$$

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

$$P(r_{uj} = v_s | \textit{Observed ratings in } I_u) = \frac{P(r_{uj} = v_s) \cdot P(\textit{Observed ratings in } I_u | r_{uj} = v_s)}{P(\textit{Observed ratings in } I_u)} \tag{3.4}$$

We need to determine the value of s in the aforementioned expression for which the value of $P(r_{uj} = v_s | \textit{Observed ratings in } I_u)$ on the left-hand side is as large as possible. It is noteworthy that the denominator on the right-hand side of Equation 3.4 is independent of the value of s. Therefore, in order to determine the value of s at which the right-hand side takes on the maximum value, one can ignore the denominator and express the aforementioned equation in terms of a constant of proportionality:

$$P(r_{uj} = v_s | \textit{Observed ratings in } I_u) \propto P(r_{uj} = v_s) \cdot P(\textit{Observed ratings in } I_u | r_{uj} = v_s) \tag{3.5}$$

If desired, the constant of proportionality can be derived by ensuring that all the resulting probability values $P(r_{uj} = v_s | \textit{Observed ratings in } I_u)$ for $s \in \{1 \ldots l\}$ sum to 1. A key observation is that all the expressions on the right-hand side of Equation 3.5 can be estimated easily in a data-driven manner. The value of $P(r_{uj} = v_s)$, which is also referred

to as the *prior probability* of rating r_{uj}, is estimated to the fraction of the users that have specified the rating v_s for the jth item. Note that the fraction is computed only out of those users that have rated item j, and the other users are ignored. The expression $P(\textit{Observed ratings in } I_u | r_{uj} = v_s)$ is estimated with the use of the *naive assumption*. The naive assumption is based on *conditional independence* between the ratings. The conditional independence assumption says that the ratings of user u for various items in I_u are independent of one another, *conditional* of the fact that the value of r_{uj} was observed to be v_s. This condition may be mathematically expressed as follows:

$$P(\textit{Observed ratings in } I_u | r_{uj} = v_s) = \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s) \tag{3.6}$$

The value of $P(r_{uk}|r_{uj} = v_s)$ 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. By plugging in the estimation of the prior probability $P(r_{uj} = v_s)$ and that of Equation 3.6 into Equation 3.5, it is possible to obtain an estimate of the posterior probability of the rating of item j for user u as follows:

$$P(r_{uj} = v_s | \textit{Observed ratings in } I_u) \propto P(r_{uj} = v_s) \cdot \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s) \tag{3.7}$$

This estimate of the posterior probability of the rating r_{uj} can be used to estimate its value in one of the following two ways:

1. By computing each of the expressions on the right-hand side of Equation 3.7 for each $s \in \{1 \ldots l\}$, and determining the value of s at which it is the largest, one can determine the most likely value $\hat{r}_{uj}$ of the missing rating r_{uj}. In other words, we have:

$$\begin{aligned} \hat{r}_{uj} &= \text{argmax}_{v_s} P(r_{uj} = v_s | \textit{Observed ratings in } I_u) \\ &= \text{argmax}_{v_s} P(r_{uj} = v_s) \cdot \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s) \end{aligned}$$

   Such an approach, however, treats a rating purely as a categorical value and ignores all ordering among the various ratings. When the number of possible ratings is small, this is a reasonable approach to use.

2. Rather than determining the rating that takes on the maximum probability, one can estimate the predicted value as the weighted average of all the ratings, where the weight of a rating is its probability. In other words, the weight of the rating v_s is proportional to the value of $P(r_{uj} = v_s | \textit{Observed ratings in } I_u)$, as computed in Equation 3.7. Note that the constant of proportionality in the equation is irrelevant for computing the weighted average. Therefore, the estimated value $\hat{r}_{uj}$ of the missing rating r_{uj} in the matrix R is as follows:

$$\begin{aligned} \hat{r}_{uj} &= \frac{\sum_{s=1}^{l} v_s \cdot P(r_{uj} = v_s | \textit{Observed ratings in } I_u)}{\sum_{s=1}^{l} P(r_{uj} = v_s | \textit{Observed ratings in } I_u)} \\ &= \frac{\sum_{s=1}^{l} v_s \cdot P(r_{uj} = v_s) \cdot P(\textit{Observed ratings in } I_u | r_{uj} = v_s)}{\sum_{s=1}^{l} P(r_{uj} = v_s) \cdot P(\textit{Observed ratings in } I_u | r_{uj} = v_s)} \\ &= \frac{\sum_{s=1}^{l} v_s \cdot P(r_{uj} = v_s) \cdot \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)}{\sum_{s=1}^{l} P(r_{uj} = v_s) \cdot \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)} \end{aligned}$$

   This approach is preferable when the granularity of the ratings distribution is greater.

For a given user u, all her unobserved ratings are estimated using this approach. The items with the top-k estimated values of the ratings are reported.

It is noteworthy that this approach computes the conditional probability of a rating, based on the ratings of the other *items* (or dimensions). Therefore, this approach is an *item*-based Bayes approach. This approach is a straightforward adaptation of traditional classification methods, except that the predicted (class) dimension is fixed in traditional classification, whereas the predicted dimension varies in collaborative filtering. This difference occurs because collaborative filtering is a generalization of classification (cf. Figure 3.1). In the particular case of collaborative filtering, it is also possible to compute the probability of a rating based on the ratings of the other *users* for the same item (see Exercise 4). Such an approach can be viewed as a *user*-based Bayes approach. It is even possible to combine the predictions from the user-based and item-based Bayes methods. In virtually all forms of collaborative filtering, such as neighborhood-based and rule-based methods, it is possible to provide a solution from the user-based perspective, the item-based perspective, or a combination of the two methods.

3.4.1 Handling Overfitting

A problem arises when the underlying ratings matrix is sparse and the number of observed ratings is small. In such cases, the data-driven estimations may not remain robust. For example, the estimation of the prior probability $P(r_{uj} = v_s)$ is unlikely to be robust if a small number of users have specified ratings for the jth item. For example, if no user has specified a rating for the jth item, the estimation is of the form $0/0$, which is indeterminate. Furthermore, the estimation of each value $P(r_{uk}|r_{uj} = v_s)$ on the right-hand side of Equation 3.6 is likely to be even less robust than the estimation of the prior probability. This is because only a small portion of the ratings matrix will be conditional on the event $r_{uj} = v_s$. In this case, the portion of the ratings matrix that needs to be analyzed is only those users that have specified the rating v_s for item j. If the number of such users is small, the estimation will be inaccurate and the multiplicative terms in Equation 3.6 will produce a large error. For example, for any value of $k \in I_u$, if no user has specified the rating r_{uk} in cases where the rating of the jth item is set to v_s, the entire expression of Equation 3.6 will be set to 0 because of its multiplicative nature. This is, of course, an erroneous and overfitting result, which is obtained because of the estimation of the model parameters from a small amount of data.

In order to handle this problem, the method of Laplacian smoothing is commonly used. For example, let $q_1 \ldots q_l$ be the number of users that have respectively specified the ratings $v_1 \ldots v_l$ for the jth item. Then, instead of estimating $P(r_{uj} = v_s)$ in a straightforward way to $q_s / \sum_{t=1}^{l} q_t$, it is smoothed with a Laplacian smoothing parameter α:

$$P(r_{uj} = v_s) = \frac{q_s + \alpha}{\sum_{t=1}^{l} q_t + l \cdot \alpha} \tag{3.8}$$

Note that if no ratings are specified for the jth item, then such an approach will set the prior probability of each possible rating to $1/l$. The value of α controls the level of smoothing. Larger values of α will lead to more smoothing, but the results will become insensitive to the underlying data. An exactly similar approach can be used to smooth the estimation of $P(r_{uk}|r_{uj} = v_s)$, by adding α and $l \cdot \alpha$ to the numerator and denominator, respectively.

Table 3.2: Illustration of the Bayes method with a binary ratings matrix

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6
1	1	-1	1	-1	1	-1
2	1	1	?	-1	-1	-1
3	?	1	1	-1	-1	?
4	-1	-1	-1	1	1	1
5	-1	?	-1	1	1	1

3.4.2 Example of the Bayes Method with Binary Ratings

In this section, we will illustrate the Bayes method with a binary ratings matrix on 5 users and 6 items. The ratings are drawn from $\{v_1, v_2\} = \{-1, 1\}$. This matrix is shown in Table 3.2. For ease in discussion, we will not use Laplacian smoothing although it is essential to do so in practice. Consider the case in which we wish to predict the ratings of the two unspecified items of user 3. Therefore, we need to compute the probabilities of the unspecified ratings r_{31} and r_{36} taking on each of the values from $\{-1, 1\}$, conditional on the observed values of the other ratings of user 3. By using Equation 3.7, we obtain the following posterior probability for the rating of item 1 by user 3:

$$P(r_{31} = 1|r_{32}, r_{33}, r_{34}, r_{35}) \propto P(r_{31} = 1) \cdot P(r_{32} = 1|r_{31} = 1) \cdot P(r_{33} = 1|r_{31} = 1) \cdot \\ \cdot P(r_{34} = -1|r_{31} = 1) \cdot P(r_{35} = -1|r_{31} = 1)$$

The values of the individual terms on the right-hand side of the aforementioned equation are estimated using the data in Table 3.2 as follows:

$$P(r_{31} = 1) = 2/4 = 0.5$$
$$P(r_{32} = 1|r_{31} = 1) = 1/2 = 0.5$$
$$P(r_{33} = 1|r_{31} = 1) = 1/1 = 1$$
$$P(r_{34} = -1|r_{31} = 1) = 2/2 = 1$$
$$P(r_{35} = -1|r_{31} = 1) = 1/2 = 0.5$$

Upon substituting these values in the aforementioned equation, we obtain the following:

$$P(r_{31} = 1|r_{32}, r_{33}, r_{34}, r_{35}) \propto (0.5)(0.5)(1)(1)(0.5) = 0.125$$

Upon performing the same steps for the probability of r_{31} taking on the value of -1, we obtain:

$$P(r_{31} = -1|r_{32}, r_{33}, r_{34}, r_{35}) \propto (0.5)\left(\frac{0}{1}\right)\left(\frac{0}{2}\right)\left(\frac{0}{2}\right)\left(\frac{0}{2}\right) = 0$$

Therefore, the rating r_{31} has a higher probability of taking on the value of 1, as compared to -1, and its predicted value is set to 1. One can use a similar argument to show that the predicted value of the rating r_{36} is -1. Therefore, in a top-1 recommendation scenario, item 1 should be prioritized over item 6 in a recommendation to user 3.

3.5 Using an Arbitrary Classification Model as a Black-Box

Many other classification (or regression modeling) methods can be extended to the case of collaborative filtering. The main challenge in these methods is the incomplete nature of the underlying data. In the case of some classifiers, it is more difficult to adjust the model to handle the case of missing attribute values. An exception is the case of *unary* data, in which missing values are often estimated to be 0, and the specified entries are set to 1. Therefore, the underlying matrix resembles sparse binary data of high dimensionality. In such cases, the data can be treated as a complete data set and any classifiers that are designed for sparse and high dimensional data can be used. Fortunately, many forms of data, including customer transaction data, Web click data, or other activity data, can be formulated as a unary matrix. It is noteworthy that text data is also sparse and high-dimensional; as a result, many of the classification algorithms used in text mining can be directly adapted to these data sets. In fact, it has been shown in [669] that one can directly leverage the success of support vector machines in text data to (unary) collaborative filtering, albeit with a squared form of the loss function. The squared form of the loss function makes the model more akin to regularized linear regression. It has also been suggested in [669] that the use of rare class learning methods can be effective in collaborative filtering due to the imbalanced nature of the class distribution. For example, one might use different loss functions for the majority and minority classes while adapting the support vector machine to the collaborative filtering scenario. Numerous ad hoc methods have also been proposed to extend various classification and regression methods to collaborative filtering. For example, smoothing support vector machines [638] have been used to estimate the missing values in the user-item matrix in an iterative way.

For cases in which the ratings matrix is not unary, it is no longer possible to fill in the missing entries of the matrix with 0s without causing significant bias. This issue is discussed in detail in section 2.5 of Chapter 2. Nevertheless, as discussed in the same section, several dimensionality reduction methods can be used to create a low-dimensional representation of the data, which is fully specified. In such cases, any known classification method can be used effectively by treating the low-dimensional representation as the feature variables of the training data. Any column that needs to be completed is treated as the class variable. The main problem with this approach is a loss of interpretability in the classification process. When the reduced representation represents a linear combination of the original columns, it is difficult to provide any type of explanation of the predictions.

In order to work in the original feature space, it is possible to use classification methods as meta-algorithms in conjunction with iterative methods. In other words, an off-the-shelf classification algorithm is used as a *black-box* to predict the ratings of one of the items with the ratings of other items. How does one overcome the problem that the training columns haven been incompletely specified? The trick is to iteratively fill in the missing values of the