Collaborative Filtering

SC4/SM4 Data Mining and Machine Learning, Hilary Term 2017

Dino Sejdinovic

4.1 Ratings and Recommendations

Collaborative Filtering (CF) is a collective name for a range of techniques that tackle the problem of making predictions about the preferences of a set of *users* on a set of *items*, based on the user's ratings on other items and based on the ratings of other users. Typical example concerns predicting movie preferences based on the ratings of previously watched movies – popularized by the 2006 Netflix competition.

movie \ user	Alice	Bob	Chuck	Dan	Eve
Happy Gilmore	?	2	5	1	4
Click	1	?	4	?	?
Ex Machina	?	4	?	?	2
Blade Runner	5	?	1	?	?
The Matrix	5	5	?	?	4

In a typical setup, we have a *partially observed* matrix $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_2}$ where $y_{i,j}$ is the rating (e.g. between 1 and 5) of movie i by user j, assuming we have n_1 movies and n_2 users[1]. Most entries will be *missing/unknown* since most users will not have rated most movies. We will also introduce a matrix of *exposure indicators* $\mathbf{E}$ where $e_{i,j} = 1$ if the user j has rated movie i and $e_{i,j} = 0$ otherwise.

4.2 Content-Based Recommendations and Alternating Linear Regressions

In the case where additional attributes about users or about the movies *are observed*, the problem can be treated in a supervised learning fashion. Assume that for each movie i we also have access to a *feature vector* $\phi_i = [\phi_{i1}, \ldots, \phi_{ik}]^\top \in \mathbb{R}^k$ (for example, ϕ_{i1} may indicate whether a movie i is a romantic comedy, ϕ_{i2} whether it is based on a comic book etc). Then we could simply formulate the problem as n_2 separate linear models[2] for each user j:

$$\min_{\psi_j} \sum_{i\,:\,e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2 + \lambda_\psi \|\psi_j\|_2^2, \quad j = 1, \ldots, n_2. \tag{4.1}$$

Here, ψ_j is the corresponding vector of coefficients in the linear model, and we have included the L_2-regularization term (which becomes important if some users have rated only a small number of movies). This model is called *content-based recommendation system* since it depends on specific

[1] note the departure from our usual $n \times p$ convention – indeed, we will consider that both users and movies have some underlying set of variables – but that these are not necessarily observed

[2] in the typical case of integer ratings, a generalised linear model is more appropriate, as linear model can make predictions outside of the range of valid rating values, but we are keeping things simple

features of the movies. Note that content-based recommendations are not "collaborative" in the sense that recommendations made to a user do not make use of the information across the entire user-base.

We often do not have appropriate features for movies and even if we do, it is not clear if those specific features are relevant for ratings prediction. Notice that $\psi_j = [\psi_{j1}, \ldots, \psi_{jk}]^\top \in \mathbb{R}^k$ in (4.1) can be treated as a preference vector for each user j (e.g. ψ_{j1} tells us whether the user j likes romantic comedies, ψ_{j2} whether the user j likes movies based on comic books etc), so let us assume for the moment that we have access to these user preferences but not to the actual feature vectors ϕ_i for the movies. Because of the symmetry in the model, we can now infer those feature vectors, based on the preferences:

$$\min_{\phi_i} \sum_{j\,:\,e_{i,j}=1} (y_{i,j} - \phi_i^\top \psi_j)^2 + \lambda_\phi \|\phi_i\|_2^2, \quad i = 1, \ldots, n_1. \tag{4.2}$$

Thus, we see that it is possible to formulate the predictions not based on features of the movies nor based on the preferences of the users (either of which may or may not be observed), but merely on the ratings matrix: we *randomly initialise* movie feature vectors ϕ_i, and then perform an iterative minimization alternating between the updates (4.1) and (4.2). This may result in features/preferences that do not have an easily understandable meaning, but are capturing salient movie/user properties that result in the ratings we observe. Moreover, by optimizing over both movie features and user preferences, predictions for each user can potentially depend on ratings of all other users (i.e. they are "collaborative").

While alternating linear regressions can be solved in closed form, due to very large numbers n_1 and n_2 of movies and users, in practice one often uses stochastic gradient descent (SGD) updates, where when we observe a new rating $y_{i,j}$, we only update the feature vector ϕ_i of movie i and the preference vector ψ_j of user j:

$$\phi_i \leftarrow (1 - \epsilon_t \lambda_\phi)\phi_i + \epsilon_t \psi_j (y_{ij} - \phi_i^\top \psi_j), \tag{4.3}$$

$$\psi_j \leftarrow (1 - \epsilon_t \lambda_\psi)\psi_j + \epsilon_t \phi_i (y_{ij} - \phi_i^\top \psi_j). \tag{4.4}$$

4.3 Low-Rank Matrix Factorization

The method of alternating linear regressions can be understood as low-rank matrix factorization of the ratings matrix $\mathbf{Y}$. Indeed, the ratings matrix is being approximated as a product of two low-rank matrices, $\Phi \in \mathbb{R}^{n_1 \times k}$, $\Psi \in \mathbb{R}^{n_2 \times k}$, where typically $k \ll \min(n_1, n_2)$, such that $\mathbf{Y} \approx \Phi\Psi^\top$. The columns $\phi^{(1)}, \ldots, \phi^{(k)}$ of Φ can be interpreted as *learned attributes* of movies, whereas columns $\psi^{(1)}, \ldots, \psi^{(k)}$ of Ψ can be interpreted as *learned attributes* of users.

If $\mathbf{Y}$ was fully observed then the best low-rank approximation is given by SVD, i.e. from SVD $\mathbf{Y} = UDV^\top$ we can set $\Phi = U_{1:n_1,1:k} D^{1/2}_{1:k,1:k}$ and $\Psi = V_{1:n_2,1:k} D^{1/2}_{1:k,1:k}$ and this is a solution[3] of

[3] not unique as D can be distributed arbitrarily between Φ and Ψ - compare to the discussion of different versions of scaled biplots

$$\min_{\Phi,\Psi} \underbrace{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}(y_{i,j}-\phi_i^\top\psi_j)^2}_{=\|\mathbf{Y}-\Phi\Psi^\top\|_F^2}. \tag{4.5}$$

However, as most entries in $\mathbf{Y}$ are missing, we have the optimization problem given by

$$\min_{\Phi,\Psi} \sum_{e_{i,j}=1} (y_{i,j}-\phi_i^\top\psi_j)^2. \tag{4.6}$$

This seemingly minor modification results in a difficult optimization problem which cannot be solved using standard SVD techniques. Moreover, it is typically needed to add regularization terms due to a large number of missing entries in $\mathbf{Y}$, which results exactly in the objective of the alternating linear regressions:

$$\min_{\Phi,\Psi} \sum_{e_{i,j}=1} (y_{i,j}-\phi_i^\top\psi_j)^2 + \lambda_\phi\|\Phi\|_F^2 + \lambda_\psi\|\Psi\|_F^2. \tag{4.7}$$

4.4 Probabilistic Matrix Factorization

Introduced in [2], the generative model corresponding to CF can be described as follows:

- For each movie $i = 1,\dots,n_1$, sample independently the latent vector of features $\phi_i \sim \mathcal{N}(0,\sigma_\phi^2 I_k)$ from a k-dimensional normal distribution,
- For each user $j = 1,\dots,n_2$, sample independently the latent vector of preferences $\psi_j \sim \mathcal{N}(0,\sigma_\psi^2 I_k)$ from a k-dimensional normal distribution,
- For each movie-user pair (i,j), sample $e_{i,j} \sim \mathit{Bernoulli}(p)$ independently and if $e_{i,j}=1$, sample $y_{i,j}|\phi_i,\psi_j \sim \mathcal{N}(\phi_i^\top\psi_j,\sigma_y^2)$.

The (hyper)parameter vector here is given by $\theta = (\sigma_\phi^2,\sigma_\psi^2,\sigma_y^2)$. We can write the joint probability density of the observations and the latent variables as

$$
\begin{aligned}
p(\mathbf{Y}, \Phi, \Psi|\theta) &= \prod_{i=1}^{n_1} \frac{1}{(2\pi\sigma_\phi^2)^{k/2}} \exp\left(-\frac{1}{2\sigma_\phi^2}\|\phi_i\|_2^2\right) \\
&\quad \cdot \prod_{j=1}^{n_2} \frac{1}{(2\pi\sigma_\psi^2)^{k/2}} \exp\left(-\frac{1}{2\sigma_\psi^2}\|\psi_j\|_2^2\right) \\
&\quad \cdot \prod_{e_{i,j}=1} \frac{1}{(2\pi\sigma_y^2)^{1/2}} \exp\left(\frac{1}{2\sigma_y^2}(y_{i,j} - \phi_i^\top \psi_j)^2\right) \\
&\propto \exp\left(-\frac{1}{2\sigma_\phi^2}\|\Phi\|_F^2 - \frac{1}{2\sigma_\psi^2}\|\Psi\|_F^2 - \frac{1}{2\sigma_y^2}\sum_{e_{i,j}=1}(y_{i,j} - \phi_i^\top \psi_j)^2\right). \qquad (4.8)
\end{aligned}
$$

Maximizing $\log p(\Phi, \Psi|\mathbf{Y}, \theta)$ in this model thus corresponds exactly to (4.7) with regularization parameters given by $\lambda_\phi = \sigma_y^2/\sigma_\phi^2$, $\lambda_\psi = \sigma_y^2/\sigma_\psi^2$. Since we now have a full probabilistic model, however, it is possible to consider joint inference of Φ, Ψ and θ as well as to consider more sophisticated model construction.

4.5 User-based and Item-based Collaborative Filtering

There is also a range of model-free (also called *memory-based*) methods for collaborative filtering, which are typically based on some notion of *user-user similarity* or *item-item similarity*.

User-based CF (UBCF) starts with a notion of *user-user similarity* computed based on the ratings, and then predicts ratings by *aggregating those of similar users*. Generally, it proceeds in the following three steps:

1. Assign a weight to all users with respect to similarity with the current user.
2. Select k users that have the highest similarity with the current user – commonly called the *neighbourhood*.
3. Compute a prediction using a weighted combination of the neighbours' ratings.

An example similarity κ is given by

$$
\kappa_{j,j'} = \frac{\sum_{i \in I_{jj'}}(y_{ij} - \bar{y}^j)(y_{ij'} - \bar{y}^{j'})}{\sqrt{\sum_{i \in I_{jj'}}(y_{ij} - \bar{y}^j)^2}\sqrt{\sum_{i \in I_{jj'}}(y_{ij'} - \bar{y}^{j'})^2}}, \qquad (4.9)
$$

where $\bar{y}^j = \text{avg}_{i\,:\,e_{ij}=1}\{y_{ij}\}$ denotes the average rating given by user j and $I_{jj'} = \sum_{i=1}^{n_1} e_{ij}e_{ij'}$ is the set of movies rated by both users j and j'. Thus, this similarity is simply the Pearson correlation between the ratings columns restricted to movies rated by both users. Now to make a prediction for

a new movie-user pair (i, j), we can simply aggregate predictions over the *neighbourhood* $U_k(i, j)$ of user j: the k users most similar to j who rated movie i, for example:

$$\hat{y}_{i,j} = \bar{y}^j + \frac{\sum_{j' \in U_k(i,j)} \kappa_{j,j'}(y_{i,j'} - \bar{y}^{j'})}{\sum_{j' \in U_k(i,j)} |\kappa_{j,j'}|}. \tag{4.10}$$

Item-based CF (IBCF) works analogously by aggregating predictions the user has made on similar movies. There is a large number of different variants of these algorithms, which consider different similarity measures and different aggregation strategies. See [1] for further details and references.

4.6 Biclustering

Also known as *coclustering*, this is a method for clustering both rows (items) and columns (users) in the observed data matrix. This can be a ratings matrix in CF, but can also correspond to a more general data matrix – for example, biclustering is often used in analysing gene expression data.

The intuition behind biclustering is as follows: even if the two users have a similar movie taste, it is extremely unlikely that the two have watched (and rated) the same movies (the overlap could in fact be very small). Thus, identifying similar users solely based on the similarity of their ratings may not be sufficient. Moreover, a group of users may have similar ratings across a certain group of movies, i.e. Alice and Bob both like science fiction, but have very different ratings across another type of movies, i.e. Alice also likes horrors but Bob hates them. Instead, we wish to *simultaneously* find groups of similar users and groups of similar movies.

In the biclustering method, we associate to each row i a latent indicator $r_i \in \{1, \dots, K^r\}$ and to each column j a latent indicator $c_j \in \{1, \dots, K^c\}$. Based on the cluster membership, matrix $\mathbf{Y}$ is partitioned into blocks, with y_{ij} belonging to the same block as $y_{i'j'}$ iff $(r_i, c_j) = (r'_i, c'_j)$. Further, we assume that the matrix entries are i.i.d. within each block (r_i, c_j), i.e.

$$p(\mathbf{Y} \mid \mathbf{r}, \mathbf{c}, \theta) = \prod_{e_{ij}=1} p(y_{ij}|r_i, c_j, \theta) = \prod_{e_{ij}=1} p(y_{ij}|\theta_{r_i,c_j}),$$

for some parametric probability distribution $p(y_{ij}|\theta_{r_i,c_j})$. For example, we can obtain a model similar to matrix factorization by letting $\theta_{r_i,c_j} = (\phi_{r_i}, \psi_{c_j})$ for movie-group-level feature vectors $\phi_{r_i} \in \mathbb{R}^k$ and user-group-level preference vectors $\psi_{c_j} \in \mathbb{R}^k$ and $p(y_{ij}|\theta_{r_i,c_j}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_{ij} - \phi_{r_i}^\top \psi_{c_j})^2\right)$. Inference can then proceed similarly as in the EM algorithm.

References

[1] Prem Melville and Vikas Sindhwani. Recommender systems. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 829–838. Springer US, Boston, MA, 2010.

[2] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.