Collaborative Filtering with Graphical Models

Ioana Bica [1]

Abstract

Online users are becoming increasingly reliant on recommender systems in to find movies, books, restaurant or other products that they might like. This report presents a model-based collaborative filtering probabilistic recommender system trained on the Yelp dataset for restaurant reviews. In this system, the restaurants and users are modelled as points in the trait space, where the higher the proximity between an user and a restaurant, the higher the chances the restaurants receives a positive rating from that user. Experiments performed on a subset of the Yelp dataset indicate that the proposed graphical model can learn distinctive features about the restaurants, which are encoded in the modelled traits. The results also show that increasing the number of traits used to model the users and restaurants does not necessarily have a positive impact on the evaluation metrics. Moreover, the advantages and disadvantages of using Gibbs sampling for estimating the posterior distributions of the latent traits for the users and restaurants are also explored.

1. Introduction

The wide range of available choices makes it difficult for users to find the right answer to what they need. Recommender systems have been developed to help users navigate through large quantities of information in order to identify products and places that would match their interests and needs. Recommender systems are becoming increasingly popular and are now used in E-commerce (Schafer et al., 1999), in personalizing search engines (Teevan et al., 2005) as well as in matchmaking (Herbrich et al., 2007), contextual advertising (Fain & Pedersen, 2006) and many other areas.

The two main approaches used to build recommender systems involve content based filtering and collaborative filtering, each making use of different types of data. Content based filtering takes into account domain-specific features of users (age, gender, occupation) and of items (producer, genre). Using this information, users receive recommendations of items that match their features. On the other hand, collaborative filtering systems take into account ratings given by users to different items, and then it identifies other similar items they might like. A hybrid approach is possible, involving collaboration via content which uses both the features of items and their rating in order to make recommendations.

Collaborative filtering is usually performed using either a memory-based or a model based method. A memory-based approach, such as the neighbourhood-based one (Breese et al., 1998) uses a rating matrix in order to be able to identify users and items similar to the query ones and compute a weighted average of their ratings. The underlying assumptions are that if users give similar ratings to some items, they will give similar ratings to the remaining ones as well and conversely for the items (Sarwar et al., 2001). Alternatively, model-based approaches for collaborative filtering fit a parametric model to the training data. This model can then be used to make new recommendations. Some examples of such model-based approaches involve clustering (Sarwar et al., 2002), matrix factorization (Lee & Seung, 1999), regression(Vucetic & Obradovic, 2005), classification (Billsus & Pazzani, 1998) and also Bayesian classification (Miyahara & Pazzani, 2000).

This project investigates a model based, probabilistic approach for performing collaborative filtering, which involves graphical models. The proposed model embeds the users and the items in a low dimensional latent space that best characterizes their properties. Then, the closer the users and items are in this low dimensional space, the higher the ranking of the user for the item is. The report focuses on the specific task of making restaurant recommendations to users, based on the data obtained from the Yelp dataset.

The advantage of using such a probabilistic model is that it can easily model uncertainty, which naturally arises in recommender systems. Moreover, graphical models, and in particular factor graphs, provide a compact representation of the joint distribution that implicitly models the assumptions

[1] Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom. Correspondence to: Ioana Bica <ib354@cam.ac.uk>.

Probabilistic Machine Learning,
MPhil in Advanced Computer Science 2017/2018

made about the system.

2. Related work

In order to understand the context of the graphical model developed in this project, this section illustrates other probabilistic approaches for building recommender systems. To begin with, TrueSkill, developed by Herbrich *et al.* (Herbrich et al., 2007) achieves state-of-the-art matchmaking and player raking. TrueSkill models the player skills as latent variables whose posterior distribution is inferred using approximate message passing (Kschischang et al., 2001).

Another popular probabilistic recommender system is Matchbox, built by Stern *et al.* (Stern et al., 2009), that can be used to obtained personalized recommendations for users of a web service. Matchbox takes into account user and item metadata (content information) in order to build a factor graph that can be trained to perform collaborative filtering. The users and items are also modelled in a trait space where the magnitude of the inner product is used to determine the rating and employs a message passing approach in order to compute posterior distributions over the latent variables.

This project takes a fundamentally different approach for inferring latent variables about the users and restaurants that can be used to make recommendations. In particular, we explore the use of Gibbs sampling for obtaining posterior probabilities over the latent traits.

3. Factor graph for collaborative filtering

The graphical model used in this project for building a recommender system was proposed by Bishop *et al.* (Bishop, 2013). To begin with, in order to perform collaborative filtering using a graphical model, the users and restaurants are represented as points in a trait space, and the following assumptions are made:

- each restaurant r is characterised by its position in the trait space (given by the vector $\mathbf{t}_r = [t_{r1}\, t_{r2}\, \ldots\, t_{rK}]^T$, which consists of a continuous number for each trait).
- each user u is characterised by its position in the trait space (given by the vector $\mathbf{p}_u = [p_{u1}\, p_{u2}\, \ldots\, p_{uK}]^T$, which also consists of a continuous number for each trait).
- the magnitude of the inner product between $\mathbf{p}_u$ and $\mathbf{t}_r$ denotes how close the user u and restaurant r are in the trait space; the higher the proximity the higher the probability of a positive rating.
- the trait values are independent (one dimension of the trait space is independent of the others).
- a user's rating for a restaurant depends only on the restaurants traits and it is independent of everything else.

In order to do so, we build a factor graph as illustrated in Figure 1. The factor graph consists of variables (circle nodes) and factors (square nodes) which define the relationship between the variables through a factor function. The edges indicate the dependencies between factors and variables.

A Gaussian prior is assumed on each individual trait p_{uk} and t_{rk}, which leads to a multivariate Gaussian prior on the restaurant trait $\mathbf{t}_r$ and on each user preference $\mathbf{p}_u$. Then, the user experience at a restaurant is computed as $x_{u,r} = \mathcal{N}(x_{u,r}; \mathbf{p}_u^T \cdot \mathbf{t}_r, 1)$. A user gives a positive rating for a restaurant $y_{u,r} = 1$ if $x_{u,r} > 0$ and a negative rating $y_{u,r} = -1$ otherwise. The observed variable in the factor graph is represented by the user's rating for a specific restaurant.

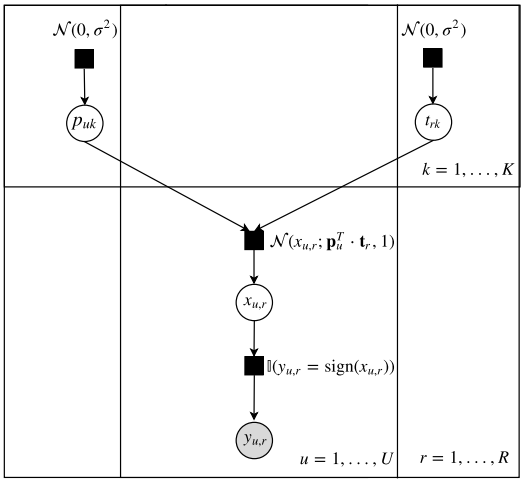

Figure 1. Factor graph for collaborative filtering that models multiple traits and a binary rating.

This collaborative filtering approach using the factor graph described requires a training dataset that can be used to learn the latent traits. For this purpose, we will use a subset of the Yelp dataset that consists of restaurant reviews. Thus, in order to model the binary ratings, we will have a dataset $\mathcal{D} = (u_i, r_i, y_i)_{i=1}^N$, where u_i represents a user id, r_i a restaurant id and $y_i \in \{+1, -1\}$ is the binary rating given by user u_i to restaurant r_i. The aim is to infer each user's preference and each restaurant's trait using this training data. Then we can use the traits learnt about all of the users and all of the restaurants to make recommendations.

4. Gibbs sampling

Performing inference using the factor graph requires integrating out the latent variables. However, doing so involves

sampling from full joint distributions of variables in the factor graph, which is computationally intractable. Therefore, this project uses Gibbs sampling to infer the full conditional distributions for the user preferences $\mathbf{p}_u$ and restaurant traits $\mathbf{t}_r$. The first step involves initialising these variables from their prior distributions:

- $p(\mathbf{t}_r) = \mathcal{N}(\mathbf{t}_r; \mathbf{0}, \mathbf{\Sigma}_0)$ for $r = 1, \dots R$
- $p(\mathbf{p}_u) = \mathcal{N}(\mathbf{p}_u; \mathbf{0}, \mathbf{\Sigma}_0)$ for $u = 1, \dots U$

Note that the prior covariance $\mathbf{\Sigma}_0$ for both the user preferences and restaurant traits is the same and it is a diagonal matrix since based on our assumptions, the individual traits are independent.

Then, we iterate through each variable v in the factor graph and update its posterior distribution by taking into account the factors that v is connected to and the values of the other variables connected to those factors. The value of variable v sampled at step s is denoted as v^s. Therefore, the Gibbs sampler will go through the following steps in turn for a pre-determined number of iterations.

- For each review in the dataset $\mathcal{D}$ sample the user experience $x_{u,r}$ given the user preference $\mathbf{p}_u$, restaurant trait $\mathbf{t}_r$ and restaurant rating $y_{u,r}$:

$$x_{u,r}^{s+1} \sim p(x_{u,r} \mid \mathbf{p}_u^s, \mathbf{t}_r^s, y_{u,r}^s) \tag{1}$$

where the conditional posterior is computed as:

$$\begin{aligned} &p(x_{u,r} \mid \mathbf{p}_u, \mathbf{t}_r, y_{u,r}) \\ &\quad\propto\quad \mathbb{I}(y_{u,r} = \text{sign}(x_{u,r})) \cdot \mathcal{N}(x_{u,r}; \mathbf{p}_u^T \cdot \mathbf{t}_r, 1) \end{aligned} \tag{2}$$

The user experience is sampled for each review in the dataset using rejection sampling. That is, we sample $x_{u,r}$ from $\mathcal{N}(\mathbf{p}_u^T \cdot \mathbf{t}_r, 1)$ and reject the sample if $\text{sign}(x_{u,r}) \neq y_{u,r}$.

- For each user $u \in \{1, 2, \dots, U\}$ sample the user preference $\mathbf{p}_u$ from the conditional posterior.

$$\mathbf{p}_u^{s+1} \sim p(\mathbf{p}_u \mid x_1^{s+1}, x_2^{s+1}, \dots, x_n^{s+1}) \tag{3}$$

A user's preference $\mathbf{p}_u$ is conditional on all the reviews given by user u, as illustrated in Figure 2, where n is the number of reviews that user u has given.

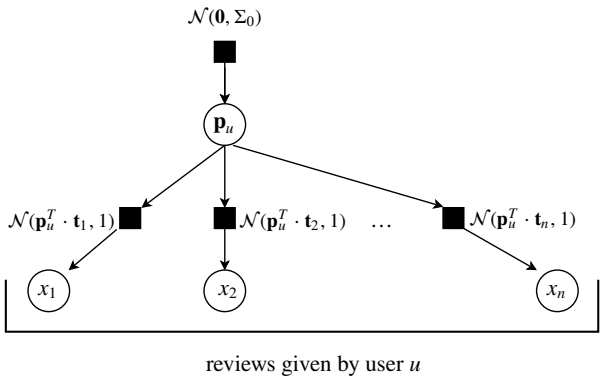

Figure 2. Section of the factor graph illustrating the factors that influence the user preference $\mathbf{p}_u$.

$$\begin{aligned} &p(\mathbf{p}_u \mid x_1, x_2, \dots, x_n) \\ &\quad\propto\quad p(\mathbf{p}_u) p(x_1, x_2, \dots, x_n \mid \mathbf{p}_u) \\ &\quad\propto\quad \mathcal{N}(\mathbf{p}_u; \mathbf{0}, \mathbf{\Sigma}_0) \prod_{i=1}^{n} \mathcal{N}(x_i; \mathbf{p}_u^T \mathbf{t}_i, 1) \\ &\quad\propto\quad \mathcal{N}(\mathbf{p}_u; \mathbf{0}, \mathbf{\Sigma}_0) \prod_{i=1}^{n} e^{-\frac{1}{2}(x_i - \mathbf{p}_u^T \mathbf{t}_i)^2} \\ &\quad\propto\quad \mathcal{N}(\mathbf{p}_u; \mathbf{0}, \mathbf{\Sigma}_0) \prod_{i=1}^{n} e^{-\frac{1}{2}(\mathbf{p}_u^T \mathbf{t}_i \mathbf{t}_i^T \mathbf{p}_u - 2 x_i \mathbf{p}_u^T \mathbf{t}_i + x_i^2)} \\ &\quad\propto\quad \mathcal{N}(\mathbf{p}_u; \mathbf{0}, \mathbf{\Sigma}_0) \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{p}_u; (\mathbf{t}_i \mathbf{t}_i^T) x_i \mathbf{t}_i, (\mathbf{t}_i \mathbf{t}_i^T)^{-1}\right) \\ &\quad\propto\quad \mathcal{N}(\mathbf{p}_u; \mu_u, \mathbf{\Sigma}_u) \end{aligned} \tag{4}$$

where

$$\mathbf{\Sigma}_u = \left(\mathbf{\Sigma_0}^{-1} + \sum_{i=1}^{n} \mathbf{t_i} \mathbf{t}_i^T\right)^{-1} \tag{5}$$

$$\mu_u = \mathbf{\Sigma}_u \left(\sum_{i=1}^{n} x_i \mathbf{t}_i\right) \tag{6}$$

- For each restaurant $r \in \{1, 2, ..., R\}$, sample restaurant traits $\mathbf{t}_r$ from conditional posterior.

$$\mathbf{t}_r^{s+1} \sim p(\mathbf{t}_r \mid x_1^{s+1}, x_2^{s+1}, \dots, x_n^{s+1}) \tag{7}$$

A restaurant's trait t_r is conditional on all the reviews given to the restaurant r. The derivation is analogous to the one for equation 4 .

$$p(\mathbf{t}_r \mid x_1, x_2, \dots, x_n) \propto \mathcal{N}(\mathbf{t}_r; \mu_r, \mathbf{\Sigma}_r) \tag{8}$$

where

$$\mathbf{\Sigma}_r = \left(\mathbf{\Sigma_0}^{-1} + \sum_{i=1}^{n} \mathbf{p_i} \mathbf{p}_i^T\right)^{-1} \tag{9}$$

$$\mu_r = \mathbf{\Sigma}_r \left(\sum_{i=1}^{n} x_i \mathbf{p}_i\right) \tag{10}$$

4.1. Ordinal regression for modelling star ratings

The current factor graph for collaborative filtering can only model binary ratings. While this can give us useful predictions, modelling star ratings can give us better information about the users. In order to obtain star ratings, we will use ordinal regression that involves modelling thresholds for each star. In this situation, the observed output for each training review j changes to $\mathbf{y}_{u,r} = [y_1\ y_2\ \dots\ y_5]^T$ where each $y_i \in \{1, -1\}$ indicates if the user u has given to restaurant r the i-th star or not. For instance, for a review with 3 stars, y_1, y_2 and y_3 will be positive and y_4 and y_5 will be negative. Figure 3 illustrates the changes to the factor graph when using star ratings. Note that this changed portion is within the plates for all of the users and all of the restaurants.

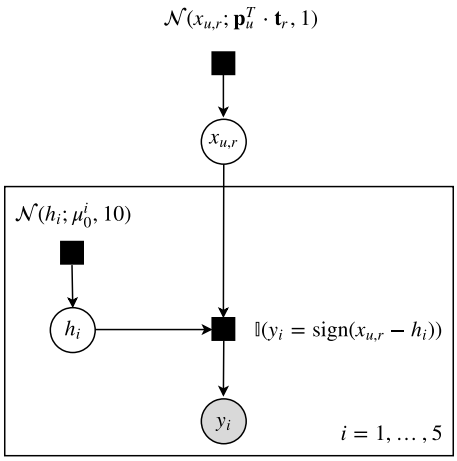

Figure 3. Section of the original factor graph that needs to be modified in order to model star ratings. This changed section remains within the plates for the users and restaurants.

We can still use Gibbs sampling for inferring the posterior distribution of the thresholds, but we need to change the way the user experience $\mathbf{x}_{ur}$ is sampled to take into account the thresholds:

- For each review in the dataset D, we now sample the user experience $x_{u,r}$ to also take into account the conditional dependence on the star thresholds:

$$\mathbf{x}_{u,r}^{s+1} \sim p(x_{u,r} \mid \mathbf{p}_u^s, \mathbf{t}_r^s, \mathbf{y}_{u,r}^s, \mathbf{h}^s) \tag{11}$$

  where the conditional posterior is now computed as:

$$\begin{aligned} p(x_{u,r} \mid \mathbf{p}_u, \mathbf{t}_r, y_1, \dots, y_5) \\ \propto \quad \mathcal{N}(x_{u,r}; \mathbf{p}_u^T \cdot \mathbf{t}_r, 1) \cdot \\ \prod_{i=1}^{5} \mathbb{I}(y_i = \text{sign}(x_{u,r} - h_i)) \end{aligned} \tag{12}$$

Algorithm 1 Gibbs sampling

Input: Dataset $\mathcal{D}$
Initialize $\mathbf{t}_r$ and $\mathbf{p}_u$.
for $s = 1$ **to** S **do**
  // Sample user experiences $x_{u,r}$
  if Modelling star ratings **then**
    for each review given by user u to restaurant r **do**
      $\mathbf{x}_{u,r}^{s+1} \sim p(x_{u,r} \mid \mathbf{p}_u^s, \mathbf{t}_r^s, \mathbf{h}^s, \mathbf{y}_{u,r})$
    end for
    for $i = 1$ **to** 5 **do**
      $h_i^{s+1} \sim p(h_i \mid x_1^{s+1}, \dots, x_N^{s+1}, y_i)$
    end for
  else
    for each review given by user u to restaurant r **do**
      $\mathbf{x}_{u,r}^{s+1} \sim p(x_{u,r} \mid \mathbf{p}_u^s, \mathbf{t}_r^s, \mathbf{y}_{u,r}^s)$
    end for
  end if

  // Sample user preferences $\mathbf{p}_u$
  for $u = 1$ **to** U **do**
    $\mathbf{p}_u^{s+1} \sim p(\mathbf{p}_u \mid x_1^{s+1}, x_2^{s+1}, \dots, x_n^{s+1})$
  end for

  // Sample restaurant traits $\mathbf{t}_r$
  for $r = 1$ **to** r **do**
    $\mathbf{t}_r^{s+1} \sim p(\mathbf{t}_r \mid x_1^{s+1}, x_2^{s+1}, \dots, x_n^{s+1})$
  end for
end for

- Each star threshold h_i is also updated to account for the user experiences and the observed ranking:

$$h_i^{s+1} \sim p(h_i \mid x_1^{s+1}, \dots, x_N^{s+1}, \mathbf{y}_i^s) \tag{13}$$

  where the posterior is computed as:

$$\begin{aligned} p(h_i \mid x_1, x_2, \dots, x_N, y_i) \propto \mathcal{N}(h_i; \mu_i, 10) \cdot \\ \prod_{i=1}^{N} \mathbb{I}(y_i = \text{sign}(x_i - h_i)) \end{aligned} \tag{14}$$

  Both the user experience and the star thresholds are sampled using rejection sampling meaning that we sample from their corresponding normal distributions and reject the samples if they do not satisfy the conditions from the indicator variables.

4.2. Gibbs sampling algorithm

Algorithm 1 gives a summary of the Gibbs sampling approach used to obtain the posterior distributions for the latent variables in the factor graph for collaborative filtering.

The number of iterations for Gibbs sampler is selected by taking into account the number of steps needed for the posterior distributions to converge. In addition, given the burn-in

time, which is the time required to get from the initial random location in the Gibbs sampler to a location that has a high probability under the posterior and the mixing time, the time required for the samples to become uncorrelated, a certain number of samples from the beginning will need to be discarded.

At the end of Gibbs sampling, the samples converge to the posterior distribution which is implicit in terms of the samples.

5. Results

5.1. Experimental set-up

Considering that the Yelp dataset consists of 4.5 million reviews, I decided to choose a subset of it in order to perform experiments. Therefore our training dataset $\mathcal{D}$ consists of 14627 restaurant reviews given by $U = 1013$ users to $R = 442$ restaurants. Due to the unequal distribution of the star ratings in the dataset, the boundary for a positive review was chosen to be at least 4 stars. The dataset was shuffled and then separated into 10000 reviews for training and 4627 reviews for testing.

The Gibbs sampler was run for 1500 iterations for the binary rating and only for 500 for star ratings (for reasons explained in section 5.5). After plotting the samples and the autocorrelation between consecutive samples, 50 samples were discarded as a "burn in" to account for the mixing time in the Gibbs sampling iterations.

5.2. Latent traits learnt for users and restaurants

Using Gibbs sampling on the factor graph in Figure 1 gives us posterior distributions over the position of the users and restaurants in the trait space. By plotting these points we can analyse distinctive features learned by the model.

Figure 4 illustrates a 2-dimensional trait space (setting $K = 2$) for the users and restaurants. The closer a user is to a restaurant in this space, the higher the probability the user will give a positive rating to the restaurant. It is noticeable that the model separates on the x-axis (representing the first trait) between American style restaurants and Asian restaurants. The y-axis, (representing the second trait) seems to make a separation from the restaurants offering dinner and the restaurants offering lunch.

Moreover, what we can clearly learn from this representation in the trait space is that users, in general, prefer Asian restaurants and restaurants offering dinner.

If we only modelling a single trait(setting $K = 1$), the users and restaurants will be positioned on a single line, as illustrated in Figure 5. While the separation between the American and Asian restaurants is still clear, there is not much more that can be inferred from this representation.

Therefore, by having multiple traits ($K > 1$), the graphical model can learn more distinctive characteristics of the users and restaurants. These features by themselves can be used for unsupervised learning purposes, for instance, to cluster similar restaurants or find out more about the kind of restaurants users tend to prefer.

5.3. Evaluation metrics

While the use of multiple traits can give us more information about users and restaurants, it is also important to assess the model using clearly defined evaluation metrics. The performance will be evaluated on the reviews in the test set.

The probability of a user liking a restaurant is computed by taking a large number of joint samples for $\mathbf{p}_u$ and $\mathbf{t}_r$ obtained through Gibbs sampling and computing the proportions for which $\mathbf{p}^T \cdot \mathbf{t}_r > 0$. If this probability is greater than 0.5 we predict that the user likes the restaurant. Then, the first evaluation metric that can be computed is the accuracy, which represents the proportion of test reviews for which our model predicts correctly if the user likes the restaurant, i.e. if it gives it more at least 4 stars.

The second metric used for assessment is the Normalized Discounted Cumulative Gain (NDCG) (Wang et al., 2013) which takes into account the actual value of the probabilities computed for each restaurants. In this case, for each user, we compute the like probability for the restaurants in the test set and rank them. We will only consider the top 5 ranked restaurants and their actual star rating. The Discounted Cumulative Gain (DCG) is computed as follows:

$$\text{DCG} = \sum_{i=1}^{5} \frac{1}{\log_2(i+1)} \cdot \text{stars}(\text{restaurant}_i) \tag{15}$$

However, the value for the DCG needs to be normalised because otherwise, it is biased towards the ratings given by a specific user for the restaurants in the test set. In order to achieve this, for each user, we compute an Ideal Discounted Cumulative Gain (IDCG) for the top 5 star ratings found for each user in the test.

$$\text{IDCG} = \sum_{i=1}^{5} \frac{1}{\log_2(i+1)} \cdot \text{stars}_i \tag{16}$$

Then, for each user, NDCG is computed as follows:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} \tag{17}$$

This metric penalizes the model if it has a low probability

Figure 4. Users and restaurants represented in a 2-dimensional trait space. That is, we set $K = 2$, where K is defined in section 3. The closer a user is to a restaurant, the higher chances the user will give a positive rating to the restaurant. The restaurants annotated are: Brasserie - breakfast and brunch restaurant, Raku - Japanesse restaurant, Citizens Kitchen Steakhouse and KoMex Fusion - Asian fusion resturant.

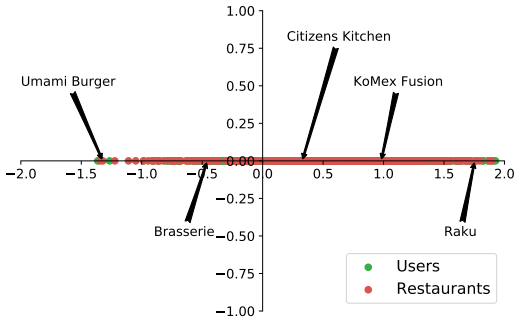

Figure 5. Restaurants and users modelled with a single trait ($K = 1$) with the same annotations as in Figure 4

.

for restaurants with high ratings. We will denote NDCG@5 the NDCG metric computed for the top 5 rankings.

5.4. Effect of the number of traits

In order to investigate if the number of traits used for inference affects performance, we have plotted the accuracy and NDCG@5 against a varying number of traits. The results are shown in Figures 6 and 7.

It is noteworthy that none of the accuracies of NDCG increase with the number of traits, but rather slowly decrease or remain the same. One explanation for this can be the fact that the training dataset does not contain enough information to accurately model a large number of traits. Another reason can be that two or three traits may be enough to capture the distinctive features of the users and of the restaurants in order to make the predictions and having more traits only adds redundant information.

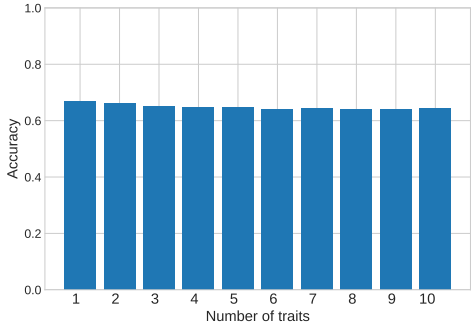

Figure 6. Accuracy of the model with a varying number of traits.

Figure 7. NDCG@5 of the model with a varying number of traits.

5.5. Star ratings

Directly modelling star ratings rather than just binary ratings can result in more powerful recommender systems. However, there were several problems we have encountered while doing so by using Gibbs sampling.

The rejection sampling approach used for sampling the user experience (equation 12) and the star thresholds (equation 14) resulted in a Gibbs sampling iteration taking 1.5 minutes on average. This is very limiting in the number of iterations we were able to run it for. The large sampling time is caused by the fact that many samples get rejected because they do not fit within the boundaries imposed by having star ratings.

Moreover, using Gibbs sampling also requires specifying different prior for the star thresholds, which is difficult to do without prior knowledge about the distribution of the star ratings datasets. For the purpose of the experiments here, we used as prior means for the stars $\mu_1 = -2, \mu_2 = -1, \mu_3 = 0, \mu_4 = 1, \mu_5 = 2$ and variance 10 to allow for the stars to move freely.

Due to the large iteration time and the fact that many samples need to be discarded to account for burn-in and mixing time in the Gibbs samples, we only modelled the star ratings for a single trait and used only 500 iterations in total.

After plotting the mean and variance of the star thresholds in Figure 8, we can notice that the thresholds for stars 2-5 have not moved at all. The threshold for the first star is not of interest because it can be set to any value lower than the lowest $\mathbf{x}_{u,r}$ sampled (this is because all of the reviews have at least one star).

It is also noticeable that the stars for which there are a lot of examples in the dataset (namely stars 4 and 5) have a much lower variance.

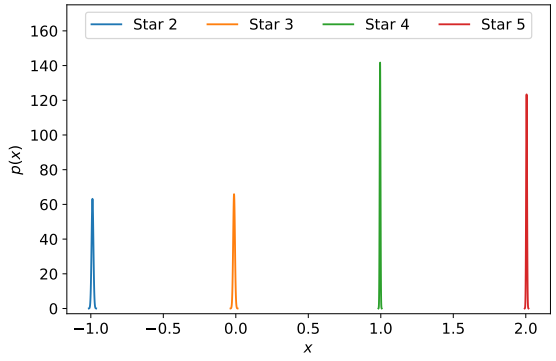

Figure 8. Posterior distributions for the star rating threshold from 2 to 5 stars.

These problems encountered with using Gibbs sampling for modelling star thresholds indicate that this approach might not be appropriate in this situation. An alternative solution, which is used in Matchbox (Stern et al., 2009) would be to use message passing instead to compute the posterior distributions.

In order to evaluate how well the model can predict the star ratings, we plotted a confusion matrix (9) for the stars predicted by the model on the test data and the actual star rating.

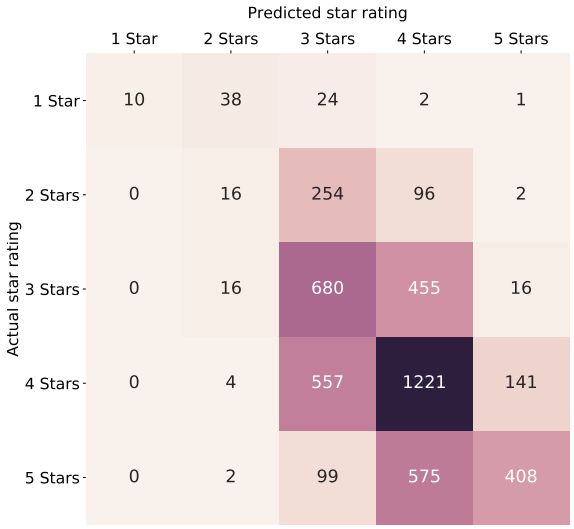

Figure 9. Confusion matrix compute on the test reviews to notice the differences between the star rating predicted by the model and the actual star rating.

The results indicate the model confuses star rating close to each other, but it also assigns four stars to a large proportion of the reviews in the test set. An evaluation metric that can be used to assess the performance of the system is the Mean Average Error (MAE), which is computed as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} \mid \text{actual rating}_i - \text{predicted rating}_i \mid \quad (18)$$

by summing over all of the reviews in the test set. Our model achieves MAE $= 0.6$ when modelling the star ratings.

5.6. Variance of learnt traits

Another issue observed was the fact that the posterior distribution when having a single trait has a high variance, particularly for users that did not give many reviews or for restaurants that have not received many reviews. Therefore, it was interesting to explore how the number of traits affects

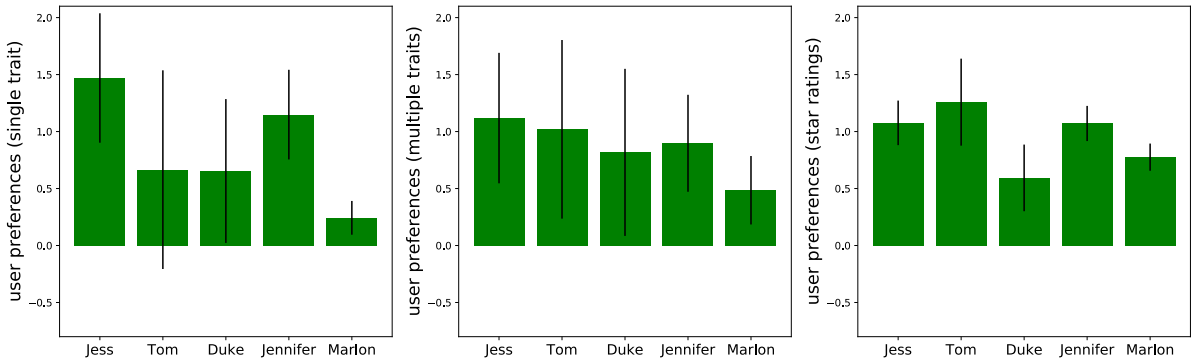

Figure 10. The mean and variance of the samples computed during Gibbs sampling for several user preferences when we model a single trait, three traits (with only one trait plotted) or when we model star ratings for a single trait.

Table 1. Number of users and restaurants under different settings for the number of traits (K) modelled or for using a star rating for which the variance of the posterior distribution is below the specified threshold. When more that one trait is modelled, we require that the variance for all the trait axes to be under the specified threshold.

		K=1	K=2	K=3	K=5	K=10	STAR RATING (K=1)
$\sigma^2 < 0.5$	USERS	466	208	143	37	4	803
	RESTAURANT	317	123	91	24	0	427
$\sigma^2 < 0.8$	USERS	792	379	290	85	19	971
	RESTAURANT	433	289	243	92	16	442

this variance and also the impact of the star ratings. The plot of the mean traits and their variance for 5 users when having a single trait, three traits (although only one of them is plotted) and star ratings in Figure 10 shows that the modelling star ratings reduces the variance the most.

The explanation for these results is given by the fact that having star thresholds and using rejection sampling on the user experience $\mathbf{x}_{u,r}$ to ensure that it fits within the thresholds for its observed rating, limits the range of values that can be taken by $\mathbf{x}_{u,r}$, thus making the restaurant traits $\mathbf{r}_t$ and user preferences $\mathbf{u}_p$ more localized in the trait space.

In order to understand the effect of the number of traits and of modelling star ratings on the variance, we used two variance thresholds and computed the number of users and restaurants that fall under the specified thresholds. The results are illustrated in 1 and they indicate that increases the number of traits does not reduce variance while modelling star ratings significantly does. One reason that may cause the increase in variance with increasing the number of traits is again, the fact that there is not sufficient training data.

6. Conclusion

This project has explored a probabilistic modelling based on factor graphs to train a recommender system that performs collaborative filtering using Yelp restaurant reviews. The restaurant and users were modelled as points in a trait space and the options for having both binary and star ratings were explored. Although Gibbs sampling was used in this project for computing the posterior probabilities of the latent variables, the problems encountered in sampling the star ratings suggest that other methods would be more appropriate.

The results obtained show that while using more traits can give us more distinctive latent features for the users and restaurants, this does not necessarily improve the evaluation metrics. Additionally, the high variance of the samples computed by Gibbs sampling for the users and restaurants indicate that a larger training dataset is required to obtain better performance.

Overall, this graphical model for performing collaborative filtering has a potential for achieving good results. One idea for future work would be to model user specific thresholds for both binary and star ratings. The intuition behind this is the fact that each user has their own bias in giving positive or negative reviews. Moreover, the incorporation of additional information about the users and restaurants in the factor graphs could reduce the variance of the samples and give better recommendations.

References

Billsus, Daniel and Pazzani, Michael J. Learning collaborative information filters. In *Icml*, volume 98, pp. 46–54, 1998.

Bishop, Christopher M. Model-based machine learning. *Phil. Trans. R. Soc. A*, 371(1984):20120222, 2013.

Breese, John S, Heckerman, David, and Kadie, Carl. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52. Morgan Kaufmann Publishers Inc., 1998.

Fain, Daniel C and Pedersen, Jan O. Sponsored search: A brief history. *Bulletin of the Association for Information Science and Technology*, 32(2):12–13, 2006.

Herbrich, Ralf, Minka, Tom, and Graepel, Thore. Trueskill: a bayesian skill rating system. In *Advances in neural information processing systems*, pp. 569–576, 2007.

Kschischang, Frank R, Frey, Brendan J, and Loeliger, H-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Lee, Daniel D and Seung, H Sebastian. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Miyahara, Koji and Pazzani, Michael J. Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International conference on artificial intelligence*, pp. 679–689. Springer, 2000.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295. ACM, 2001.

Sarwar, Badrul M, Karypis, George, Konstan, Joseph, and Riedl, John. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1, pp. 291–324, 2002.

Schafer, J Ben, Konstan, Joseph, and Riedl, John. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166. ACM, 1999.

Stern, David H, Herbrich, Ralf, and Graepel, Thore. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pp. 111–120. ACM, 2009.

Teevan, Jaime, Dumais, Susan T, and Horvitz, Eric. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 449–456. ACM, 2005.

Vucetic, Slobodan and Obradovic, Zoran. Collaborative filtering using a regression-based approach. *Knowledge and Information Systems*, 7(1):1–22, 2005.

Wang, Yining, Wang, Liwei, Li, Yuanzhi, He, Di, Chen, Wei, and Liu, Tie-Yan. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*, 2013.