INF367A Project 2

Naphat Amundsen

May 13, 2020

Introduction

This project is about creating a bayesian recommender engine specifically for movie ratings. There are different types of recommender systems. For this project, we will focus on *collaborative filtering* where we try to predict a user's preference based on preferences of other users. We are given the MovieLens 100K dataset, which essentially consists of movies, users and the users ratings for the movies. The dataset contains 100000 ratings from 943 users on 1682 movies. The data is sparse, because not every user has rated every movie. The task is to predict the users ratings on movies that they have not seen.

The recommender system is based on Bayesian matrix factorization, that is, we want to predict ratings by matrix factorization methods as well as estimating the uncertainty in the predictions. We try using three different probabilistic models to estimate the matrix factors.

1 The models

The user-rating pairs can be represented with the matrix $\underset{n\times m}{X}$, where each row represents a user, and each column represents as movie. To predict the users' rating on unreviewed movies, we try to factorize matrix $\underset{n\times m}{X}$ into two matrices $\underset{n\times k}{U}, \underset{k\times m}{V}$ such that $UV = \hat{X} \approx X$, where k denotes the number of the latent dimensions of the factors. The matrices U, V will be approximated using using Hamitonian Monte Carlo implemented in Stan [1]. Hopefully, the reconstruction of X from U and V manages to predict unseen ratings, that is UV should "fill" the missing ratings with ratings that fits each user well.

Data standard deviation

All the models share the same distribution for the data, that is $X_{ij} \sim N((UV)_{ij}, \beta)$. The prior for β is chosen to be a gamma distribution with scale= 1, and shape= 1, which seems reasonable as we expect the data points to be close to whatever UV estimates. Note that this is effectively an exponential distribution with rate 1.

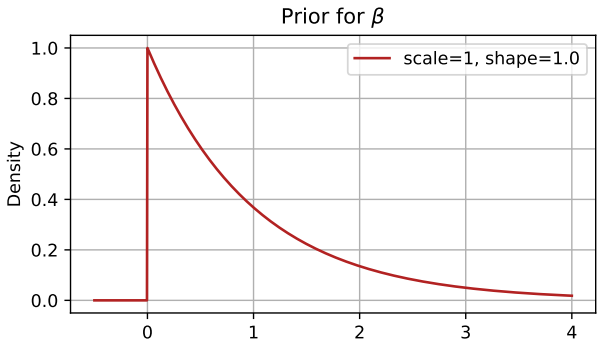


Figure 1: Probability density function for β, the assumed standard deviation of the data from the estimated values from UV.

1.1 Normal model

This model is inspired from the "regular" way of doing Bayesian linear regression, that is the elements of U and V are normally distributed. To give more flexibility to the user, the normal distributions for U and V each has different sets of user specified parameters, that is the means and standard deviations. The model rather is "vanilla", and will serve as a baseline to be beaten. We would say that it is reasonable to expect the model to at least be better than random because it generally works regression tasks.

$$\begin{aligned} U_{ij} &\sim N(\mu_U, \sigma_U) \\ V_{ij} &\sim N(\mu_V, \sigma_V) \\ \beta &\sim \text{Gamma}(a_\beta, b_\beta) \\ X_{ij} &\sim N((UV)_{ij}, \beta) \end{aligned}$$

User defined parameters: $\mu_U, \sigma_U, \mu_V, \sigma_V, a_\beta, b_\beta$.

We set the the means for the elements in U and V to be 0, and set the standard deviations to be 5, just to cover the range of the ratings within one standard deviation. That is: $\mu_U = 0,\ \sigma_U = 5,\ \mu_V = 0,\ \sigma_V = 5,\ a_\beta = 1,\ b_\beta = 1$.

1.2 Non-negative factorization model

The idea here is to constrain U and V to consist only of positive numbers, as the ratings are only positive after all. This model is very much like the normal model mentioned above, but the elements of U and V are gamma distributed instead. This model should still work based on the same logic as why the normal model works, it can do regression, but this one here has the constraint that the model parameters must be positive.

$$\begin{aligned} U_{ij} &\sim \text{Gamma}(a_U, b_U) \\ V_{ij} &\sim \text{Gamma}(a_V, b_V) \\ \beta &\sim \text{Gamma}(a_\beta, b_\beta) \end{aligned}$$

$$X_{ij} \sim N((UV)_{ij}, \beta)$$

User defined variables: $a_U, b_U, a_V, b_V, a_\beta, b_\beta$.

We specify the scale and shape such that the density "huddles" around 1, as multiplying $\underset{n\times k}{U}$ and $\underset{k\times m}{V}$ will involve a lot of multiplications and additions when reconstructing $\underset{n\times m}{X}$, which will have the flexibility to cover the whole range and ratings even if the elements of U and V are around 1, thus we find it reasonable to expect that the elements to not be very large. We achieve such a gamma distribution by setting the scale to 2 and the shape to 1.

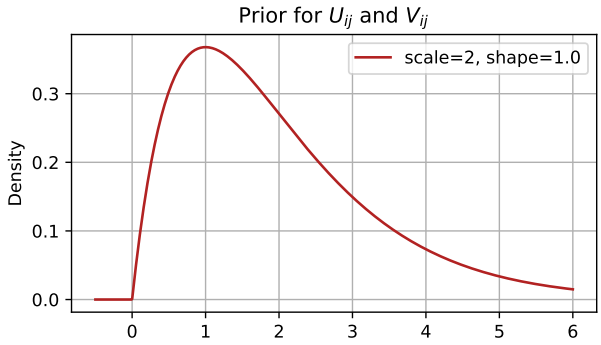


Figure 2: Probability density function for the elements in U and V.

To summarize: $a_U = 2,\ b_U = 1,\ a_V = 2,\ b_V = 1,\ a_\beta = 1,\ b_\beta = 1$.

1.3 ARD model

This model is inspired by the ARD (Automatic Relevance Determination) model used for regression tasks. This model can be viewed as an extension of the aforementioned "Normal model". The matrices $\underset{n\times k}{U}$ and $\underset{k\times m}{V}$ are still assumed to be distributed with gaussians. However, the difference is that each column (essentially components) of U and V^T has their own standard deviations. The standard deviations are assumed to be distributed from a gamma distribution with user specified parameters. We denote the standard deviations for the columns with the array α of size k, where each element correspond to each column of U and V^T.

$$\begin{aligned}
\alpha_j &\sim \text{Gamma}(a_\alpha, b_\alpha) \\
U_{ij} &\sim N(\mu_U, \alpha_j) \\
V^T_{ij} &\sim N(\mu_V, \alpha_j) \\
\beta &\sim \text{Gamma}(a_\beta, b_\beta) \\
X_{ij} &\sim N((UV)_{ij}, \beta)
\end{aligned}$$

User defined variables: $\mu_U, \mu_V, a_\alpha, b_\alpha, a_\beta, b_\beta$.

We specify the same parameters as the Normal model for the ARD model where we can, that is the means and standard deviations for U and V. We are unsure how the α values for

the ARD should take, so we pick parameters for the prior distribution that does such that it does not assume much.

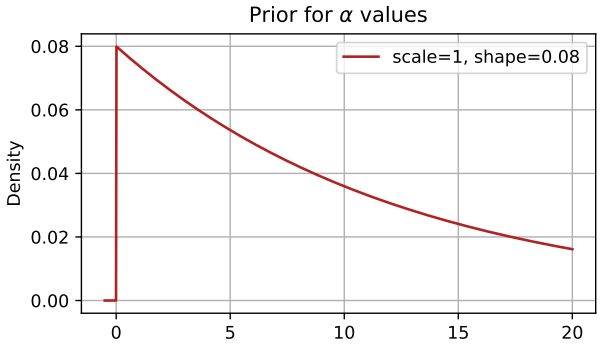


Figure 3: Probability density for α values. Again, this is effectively equivalent to an exponononential distribution with rate 0.08. Note that the rate is quite low (compared to the one used for β).

To summarize the parameters for the ARD priors: $\mu_U = 0,\ \sigma_U = 5,\ \mu_V = 0,\ \sigma_V = 5,\ a_\alpha = 1,\ b_\alpha = 0.08,\ a_\beta = 1,\ b_\beta = 1$.

2 Model selection

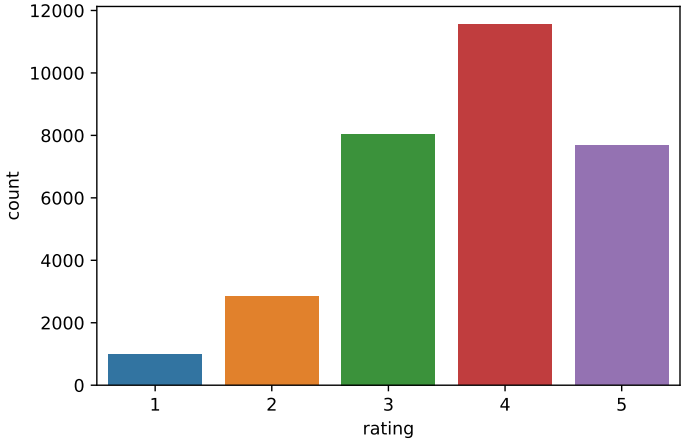
We do model selection to find the best performing model on the dataset. There are many hyperparameters that can be tuned such as the scale and shape of gamma distributions for the β-value for the models. However, as the training time takes quite a while for each model, we limit the hyperparameter search to just finding a good value for k, that is the latent dimension of the matrix factors $\underset{n\times k}{U}, \underset{k\times m}{V}$. We will subsample the data for the model selection part to speed up the process.

After determining the best candidate model from the model selection, said candidate will be trained on 90% of all the data, and then validated on the remaining 10%.

Quick look at the data

A rating is an integer ranging from 1 to 5. As mentioned earlier, the dataset contains 100000 ratings from 943 users on 1682. That means we have a 943×1682 matrix, where only 100000 elements are observed, implying we have a sparse matrix, where only 6.3% of the elements are observed.

Figure 4: Distribution of ratings.



Not only is the matrix rather sparse, but the ratings are also unbalanced, meaning we have an unbalanced dataset. This may lead to models that overfit to the most frequent rating, that is the rating of 4 in this case.

2.1 Data subsampling

To speed up the model selection process we do model selection on a subset of the data. We subsample 250 users and 250 movies. The matrix is sparse, so we want to subsample the data such that we get the "dense parts" of the matrix. To do this we pick the top 250 users based on number of movies rated, then we pick the top 250 movies with the most ratings from said top users. This produces a 250×250 matrix with about 50% observed elements. From the subset we create a train (90% of the subset) and a hold out set (the remaining 10%). The train set will be used for training, while the hold out set will be used for validation.

2.2 Model selection results

We tried 1 to 5 latent dimensions for each model. We do 2000 iterations using Stan's HMC sampler, and set the "max_treedepth" control argument to 15 (the warnings from Stan told to) when sampling for each model. The reason we try such "low" numbers for latent dimensions come from undocumented experementing, which revealed that the number of latent dimensions is probably on the lower side, as in under 10.

Scoring metric

To score the models we take the mean absolute error of all the predicted ratings to the actual ratings. That is, for each sampled U and V, we get a corresponding $\hat{X}$, which contains predictions for the movie ratings. We calcuate the absolute error of the ratings contained in $\hat{X}$ to the ratings in the validation set. We do this for every sampled pairs of U and V, and then we calculate the mean of all the errors.

Table 1: k is the latent dimension, train time shows training time in seconds, train MAE is the mean absolute error on the training set, while val MAE is the mean absolute error on the validation set. The table is sorted with respect to val MAE in ascending order (lower is better).

	model	k	train time (seconds)	train MAE	val MAE
1	ARD	3	4748.8171	0.6510	0.6822
2	Non-negative	3	2111.0567	0.6499	0.6827
3	Normal	3	3204.8550	0.6488	0.6827
4	ARD	4	5602.4438	0.6415	0.6832
5	Non-negative	2	2071.6710	0.6630	0.6843
6	ARD	2	4142.7847	0.6635	0.6845
7	Normal	2	1749.6510	0.6628	0.6850
8	Non-negative	4	2727.4210	0.6406	0.6860
9	ARD	5	5765.4504	0.6327	0.6876
10	Normal	4	3805.0494	0.6384	0.6900
11	Non-negative	5	2528.1382	0.6321	0.6938
12	Normal	5	5514.9162	0.6283	0.6997
13	Normal	1	1099.4541	0.6991	0.7083
14	Non-negative	1	1589.8258	0.6992	0.7084
15	ARD	1	2599.5240	0.6992	0.7084

Table 2: This table shows the min, max, mean and standard deviation of the $\hat{\boldsymbol{R}}$ values from the sampling of all the models. The order of the rows correspond to Table 1. $\hat{R}$ gives an indication of whether the model parameters have converged [2]. It should be noted that convergence does not imply that the sampling process managed to sample from the true posterior distribution. According to the Stan warnings, a $\hat{R}$ value for a parameter between 0.9 and 1.1 indicates likely convergence. Ideally, the $\hat{R}$ values are 1.

	model	k	Rhat min	Rhat max	Rhat mean	Rhat std
1	ARD	3	0.9990	1.3138	1.0411	0.0887
2	Non-negative	3	0.9990	1.0338	1.0032	0.0055
3	Normal	3	0.9990	2.1189	1.2102	0.2135
4	ARD	4	0.9990	1.3835	1.0594	0.0971
5	Non-negative	2	0.9990	1.0250	1.0026	0.0044
6	ARD	2	0.9990	1.2645	1.0478	0.0788
7	Normal	2	0.9990	1.3543	1.0640	0.0827
8	Non-negative	4	0.9990	1.0730	1.0052	0.0092
9	ARD	5	0.9990	2.2013	1.1479	0.2843
10	Normal	4	0.9990	2.4649	1.3759	0.3153
11	Non-negative	5	0.9990	1.1202	1.0072	0.0126
12	Normal	5	0.9990	1.9551	1.2106	0.2251
13	Normal	1	0.9994	1.1562	1.0909	0.0238
14	Non-negative	1	0.9991	1.2715	1.1814	0.0382
15	ARD	1	1.0024	1.5671	1.2801	0.1115

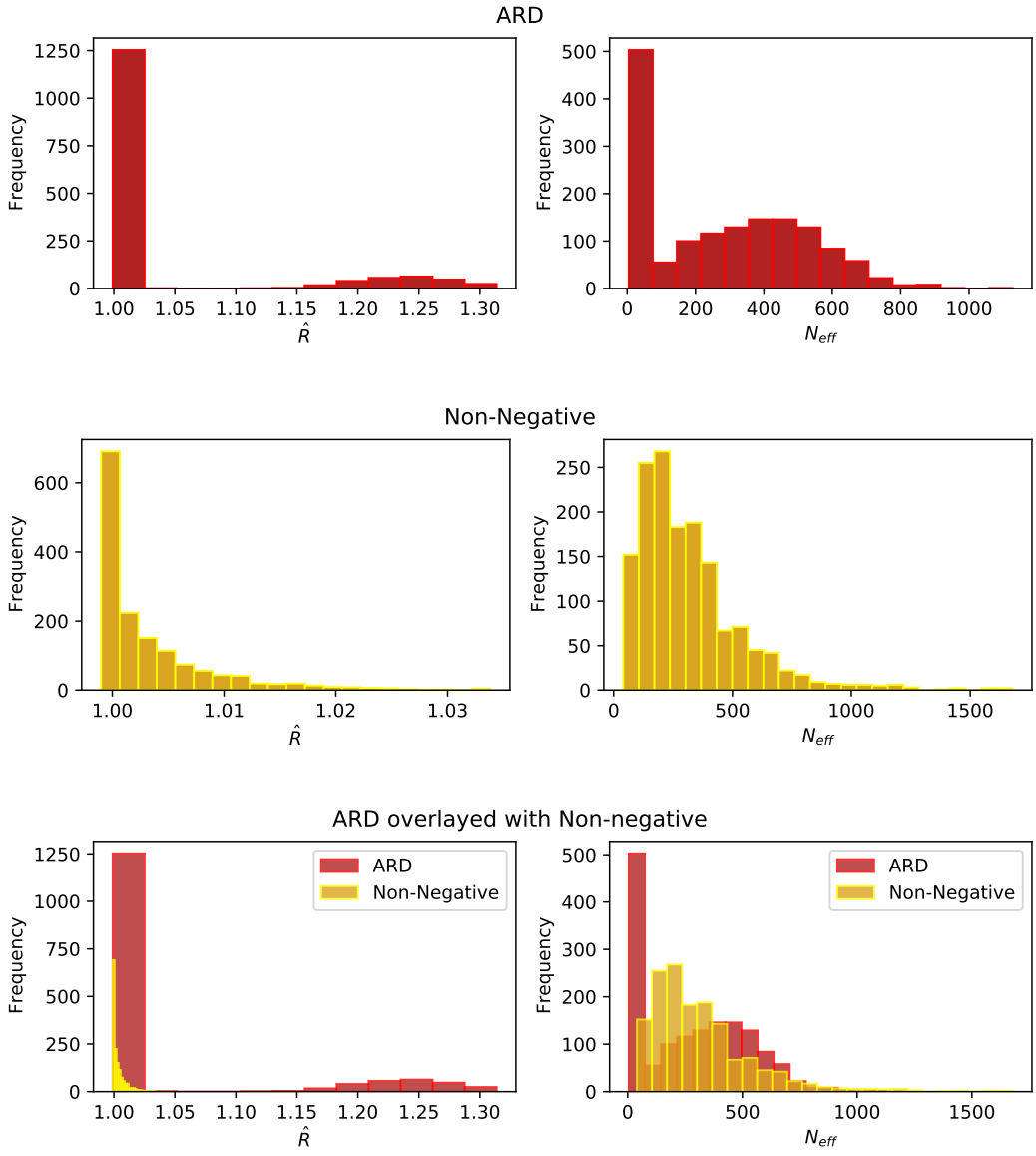
Table 3: This table shows the min, max, mean and standard deviation of the $\boldsymbol{N}_{\text{eff}}$ (number of effective samples) values from the sampling. The order of the rows correspond to Table 1. Number of effective samples is a estimate on how many samples of each model parameter got sampled from the true posterior [3]. It is desired to have many effective samples.

	model	k	N_{eff} min	N_{eff} max	N_{eff} mean	N_{eff} std
1	ARD	3	4.8574	1127.6726	275.7620	231.6963
2	Non-negative	3	40.4421	1677.8139	318.3954	223.1981
3	Normal	3	3.4360	1020.3357	10.5009	28.1623
4	ARD	4	4.8997	1571.0058	220.2744	272.0015
5	Non-negative	2	33.8124	1403.8710	234.1758	191.7673
6	ARD	2	5.5987	1657.3678	297.9770	320.3376
7	Normal	2	3.9082	1127.2144	10.0871	37.3944
8	Non-negative	4	16.0936	1799.1824	187.5072	157.2839
9	ARD	5	2.9357	1231.9205	136.8569	228.5099
10	Normal	4	2.7750	1068.2205	8.1337	25.1019
11	Non-negative	5	35.9622	1523.1771	268.6275	162.7551
12	Normal	5	3.3734	894.0220	10.0690	20.1732
13	Normal	1	10.4850	1007.2907	21.6554	47.9306
14	Non-negative	1	5.2699	771.6474	10.6626	38.0074
15	ARD	1	4.3445	586.1205	8.2810	25.9324

It clearly seems to be that 3 latent dimensions, that is the k value for $\underset{n\times k}{U}$ and $\underset{k\times m}{V}$, is the most optimal one with respect to validation MAE since the top three are all the three different types of models, all with $k = 3$. The top 3 in descending order is ARD, Non-negative and Normal, with the validation MAEs of $0.6822, 0.6827, 0.6827$ respectively. The scores are somewhat close to each other, but let us assess the convergence ($\hat{R}$) and the number of effective samples.

We assess table 2 and 3 to get some indication on the convergence and N_{eff}. The Non-negative model manages to get the best indications for convergence, the mean $\hat{R}$ is closest to one (1.0032), and has least standard deviation (0.005) compared to the others. It also manages to get the highest value for mean number of effective samples (318), but with a rather high standard deviation of around 223 (we want to have high mean and low standard deviation). The ARD model comes in second with the mean $\hat{R}$ of 1.04 and mean N_{eff} of 275. The Normal model looks rather horrible with bad convergence (mean $\hat{R}$ of 1.21), and a very low mean N_{eff} of 10, so we will not take it into any other further consideration. To get a closer look at the $\hat{R}$ and N_{eff} values for the ARD and the Non-negative model, we plot the correspoding histograms.

Figure 5: Here we see the $\hat{R}$ histograms on the left, and the N_{eff} histograms on the right. The top histograms shows the values from the ARD model, the middle histograms shows the values from Non-negative model, whilst the bottom histograms shows the top two histograms together.



We can see that the Non-negative model fares much better regarding the $\hat{R}$ values, the $\hat{R}$ values for the parameters are clearly much closer to 1 overall compared to ARD. Regarding the number of effective samples we see that the Non-negative model "looks better", as it does not have that extreme peak at the low number bin for N_{eff} opposed to the ARD model. However, if one ignores the peak on the N_{eff} histogram for the ARD model, it seems to be that the ARD manages to get more effective samples for the parameters that "makes it".

We conclude that the Non-negative model is the overall winner of the model selection,

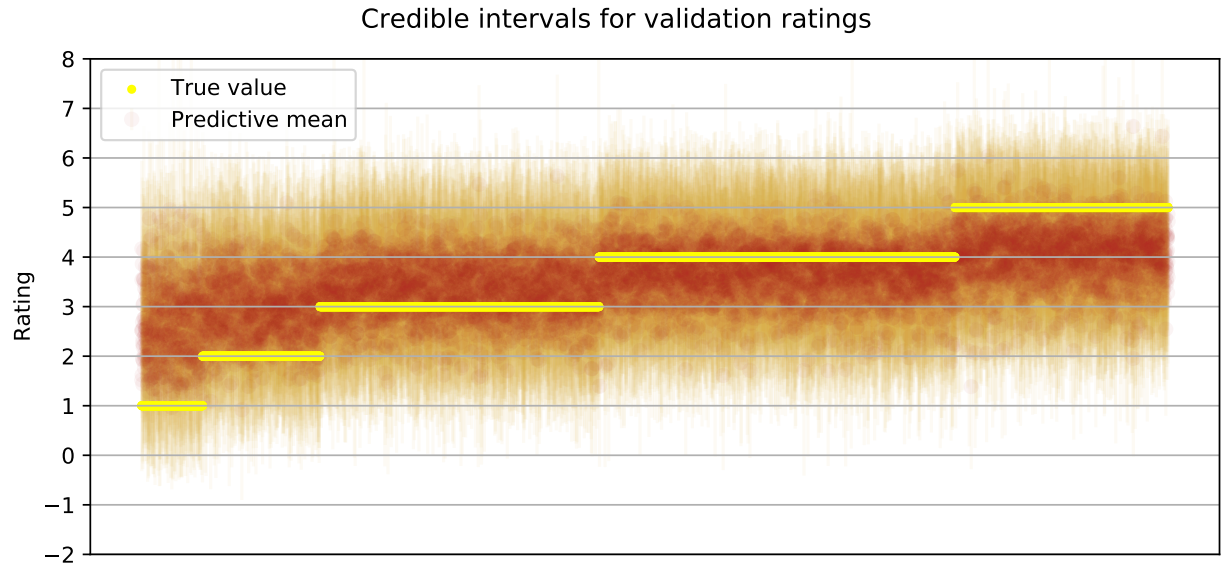
even if it got second place regarding the MAE on the validation set, but has way better indication of convergence opposed to ARD, and has a much nicer looking histogram of N_{eff}, which indicates that it probably managed to sample more from the true posterior. This decision is more of a Bayesian way of thinking.

3 Evaluating the best model

We train the best candidate from the model selection, that is the Non-negative model using 3 latent dimentions on a much larger amount of data. It should be noted that we effectively scrambled the data quite a bit during the early stages of experementing (we didn't seed in the beginning), thus we don't really have a proper test set, that is data that we have not touched whatsoever.

Using the whole dataset of 100000 ratings, we create yet another train and validation set, where 90% is used for the training set, and the rest for validation.

Figure 6: We visualize the credible intervals for the validation ratings. Each spaghetti colored vertical line represents the 95% credible interval of a rating from the predictive distribution. Each credible interval is computed from 1000 samples from the predictive distribution. Each spaghetti sauce colored dot is the mean of said 1000 predictive samples for each rating. The yellow dots (together, they resemble lines) are the true ratings. The plot is plotted with high transparency to de-emphasize outliers and "undense" parts.



As expected, the model has in a sense overfitted to the rating of 4, that is the predictive means are generally closer to the value of 4 opposed to rest. However, the model is not clueless to predicting ratings, as we can see that the lower bounds, upper bounds and means

of the credible intervals has this stair like structure respective to the true value. This at least indicates that the model does not guess randomly.

Table 4: This table shows the average 95% credible interval length in Figure 6, the models MAE on the training set as well as the validation set.

Metric	value
Mean CI length	3.5460
Train MAE	0.6845
Validation MAE	0.7435
Train time	144172 s

The train MAE and validation MAE on the train and validation sets for model selection were better with the values of 0.6510 and 0.6822 respectively (compared to the values on the much larger dataset). This is reasonable to expect, as the larger dataset is much more sparse compared to the small one used for model selection. With a validation MAE of 0.7435, we can expect that the model is gennerally off with a 1 value on unseen ratings (altough keep in mind that this is a validatioh set, and not a proper test set).

4 Discussion

Bad Rhat and low effictive samples

Some models had low convergence and low number of effective samples. As an attempt to mitigate said problems, we simply tried to increase the number of sampling iterations. We ended up using 2000 iterations for both the model selection and model evaluation. We did not see that much of a difference between 1200 sampling iterations and 2000 when it came convergence, but of course we could generally expect a little higher number of effective samples at the cost of computating times.

Ways to avoid overfitting

We could for example train on a stratified subsample, that is subsamples that are perfectly balanced in terms of ratings, however that would significantly reduce the amount of available training data, as we have to throw away quite a bit.

Extremely long sampling time when many datapoints for Non-negative model

For some reason, training the Non-negative model on 900000 ratings was extremely slow (2000 iterations, max_treedepth=15). It took 144172 seconds to finish, that is a bit over 40 hours!

What we learned

- When working with Bayesian models we should take into consideration the uncertainty of things by for example plotting the credible intervals.
- We should be aware of the implications when we choose a "best model" based on deterministic scoring metrics such as mean absolute error. We could have choosen the ARD model based on the simple reasoning that it scored best on the validation set used in model selection without any further investigation, but after assessing the indicators for convergence and sampling quality, that is the $\hat{R}$ and N_{eff} values, we observed the samples of the ARD model may not be as good as we would like. Especially when we have a candidate was right behind in terms of MAE, but had better indicators for both convergence and sampling quality. Not that it would necessarily be wrong to choose the ARD anyways, it is simply doing a thoughtfull and well educated choice.
- It is quite a challenge to predict the behaviour of Stan's Hamitonian Monte Carlo sampler. Some models which we thought should be the easiest and fastest to sample from were the slowest. The sampling time varied a lot between models and number and sampling parameters such as number of iterations.

What could have been improved

- There is no doubt that there more room for better hyperparameter search. We could try different hyperparameters for the β value for example.
- We should have been more carefull when experementing with the data from the beginning, making sure that we put aside a a proper test set from the start. We did not seed anything in the beginning and we repeatedly sampled from the full dataset for experementing.
- As mentioned, we could at try to avoid overfitting by for example stratified subsampling.

5 Conclusion

On the MovieLens100K dataset, we tried to predict movie ratings using Bayesian matrix factorization. The tried three models, Normal, Non-negative and ARD. We conducted model selection on subsample of the data consisting of the top 250 users respective to the number of ratings they have given and the top 250 movies based on the number of ratings they have received from said top 250 users. The ARD showed the best performance in terms of mean average error with 3 latent dimensions for the matrix factors, followed by the Non-negative with 3 latent dimensions as well. The Non-negative model however had noticeably better indication on sampling convergence and quality and was deemed to be the best model for the task. The Non-negative model certainly produces results that is than randomly guessing, but is far from "sure" of the ratings.

References

[1] Stan documentation, "Hamiltonian monte carlo," 2020. [accessed 08-May-2020].

[2] Stan documentation, "Convergence assessment using rhat," 2020. [accessed 12-May-2020].

[3] Stan documentation, "Effective sample size," 2020. [accessed 12-May-2020].