INF367A Project 2

Naphat Amundsen

May 12, 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 user preferences based on preferences. 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 as well as estimating the uncertainty in the predictions. We try using three different 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 into two matrices $\underset{n\times k}{U}, \underset{k\times m}{V}$ such that $UV \approx X$, 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].

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.

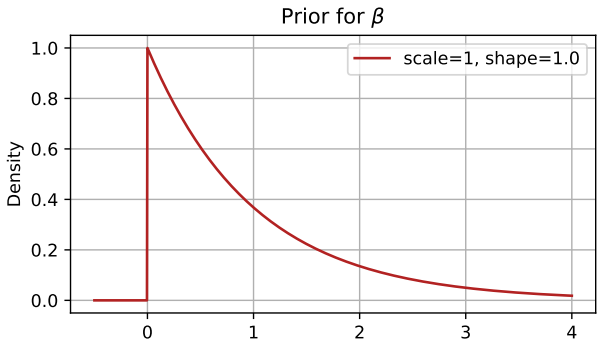


Figure 1: Probability density function for β, the assumed standard deviation of the data from the estimated values from UV.

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 data points are then assumed to be normally distributed, where the mean is what UV is at the corresponding element, and the standard deviation assumed to be distributed by a gamma distribution with user specified values for shape and scale. 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.

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

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.

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

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

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 og ratings even if the elements of U and V are around 1, thus we find it reasonable to expect values to not be very large. We achieve such a gamma distribution by setting the scale to 2 and the shape to 1.

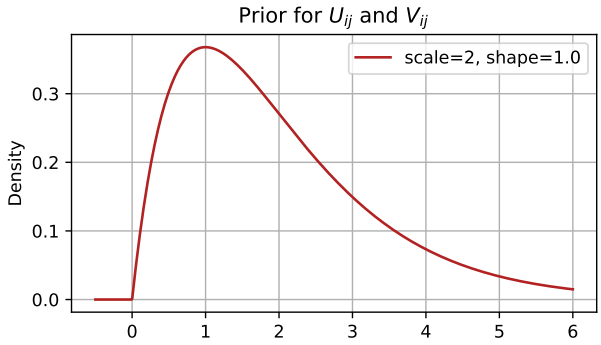


Figure 2: Probability density function for the elements in U and V.

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.

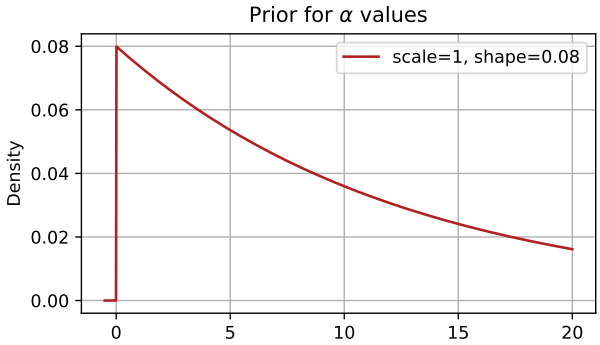


Figure 3: Probability density for α values. Again, this is effectively equivalent to an expononential distribution. Note that the rate is quite low (compared to the one used for β).

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 finding a good value for k, that is the latent dimension of the matrix factors $\underset{n\times k}{U}, \underset{k\times m}{V}$.

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

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 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 try 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.

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.

	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. The order of the rows correspond to Table 1. Rhat 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.

	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 standar 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].

	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 three different models, but with $k = 3$. The top 3 in ascending 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 variance (0.005) compared to the others. It also manages to get the highest value for mean number of effictive samples (318), but with a somewhat high standard deviation of around 223 (we want to have high mean and standard deviation). The ARD model comes in second concerning 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.

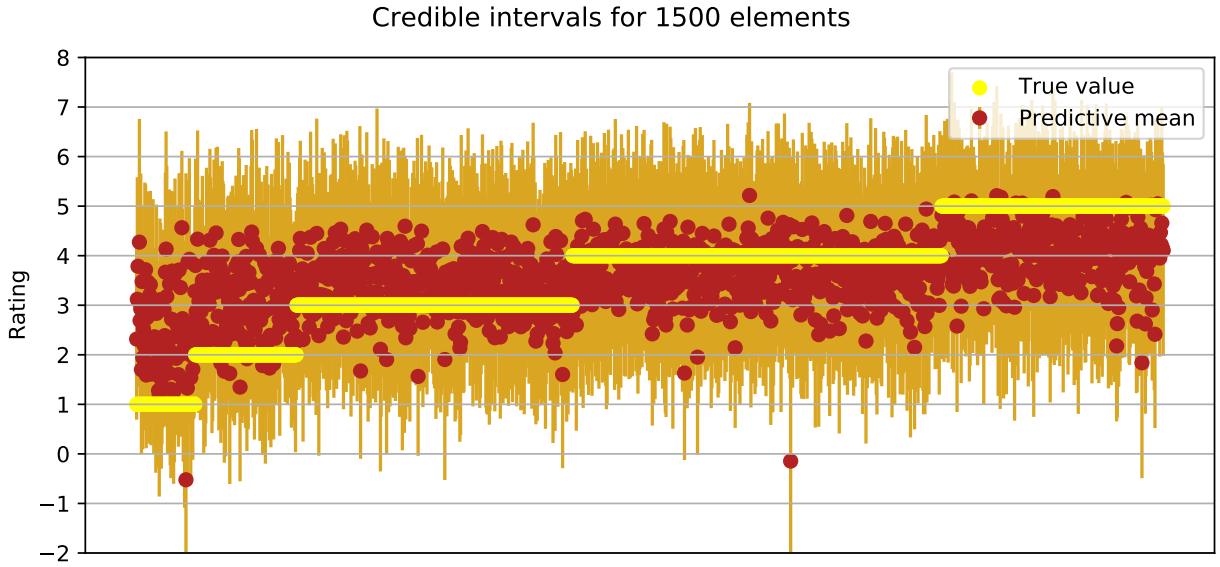
3 Evaluating the best model

We then try to train the best candidate from the model selection, that is the ARD 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 4: Credible intervals of 1500 elements from the predictive distribution. The firebrick-red dots are the means of predictive samples. The yellow dots are the true ratings. The elements correspond to the elements in the test-set.



4 Discussion

Bad Rhat and low effictive samples

Some models had low convergence, while some had low effective samples, and some were low on both. 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 expect more number of effective samples.

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].