INF367A Project 2

Naphat Amundsen

May 9, 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].

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 is 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.

$$U_{ij} \sim N(\mu_u, \sigma_u)$$

$$
\begin{aligned}
V_{ij} &\sim N(\mu_v, \sigma_v) \\
X_{ij} &\sim N((UV)_{ij}, \beta) \\
\beta &\sim Gamma(a_\beta, b_\beta)
\end{aligned}
$$

User defined parameters: $\mu_u, \sigma_u, \mu_v, \sigma_v, a_\beta, b_\beta$.

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.

$$
\begin{aligned}
U_{ij} &\sim Gamma(a_u, b_u) \\
V_{ij} &\sim Gamma(a_v, b_v) \\
X_{ij} &\sim Normal((UV)_{ij}, \beta) \\
\beta &\sim Gamma(a_\beta, b_\beta)
\end{aligned}
$$

User defined variables: $a_u, b_u, a_v, b_v, a_\beta, b_\beta$.

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 U and 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.

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

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

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 $\underset{n\times k}{U}, \underset{k\times m}{V}$.

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 Prior distributions

2.3 Model selection results

Table 1: k is the latent dimension, fit_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.

model	k	fit_time	train_mae	val_mae
ARD_Factorizer	10	5140.2271	0.6509	0.6825
NormalFactorizer	5	1559.3734	0.6489	0.6826
NonNegativeFactorizer	3	1523.8056	0.6500	0.6830
ARD_Factorizer	20	6256.7602	0.6326	0.6874
NonNegativeFactorizer	15	2177.1438	0.6321	0.6939
NormalFactorizer	5	4159.6307	0.6285	0.6995
ARD_Factorizer	15	5380.6819	0.5989	0.7070
NonNegativeFactorizer	10	1982.6133	0.6055	0.7115
NonNegativeFactorizer	20	2594.4184	0.5926	0.7239
ARD_Factorizer	3	6514.1666	0.5761	0.7260
NonNegativeFactorizer	3	3129.1756	0.5919	0.7283
ARD_Factorizer	5	7254.7662	0.5585	0.7400
NormalFactorizer	10	15866.0379	0.5859	0.7496
NormalFactorizer	15	35323.2319	0.5505	0.8026
NormalFactorizer	20	35803.3160	0.5167	0.8551

References

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