Experiments in Bayesian Recommendation

Thomas Barnard and Adam Prügel-Bennett

Abstract The performance of collaborative filtering recommender systems can suffer when data is sparse, for example in distributed situations. In addition popular algorithms such as memory-based collaborative filtering are rather ad-hoc, making principled improvements difficult. In this paper we focus on a simple recommender based on naïve Bayesian techniques, and explore two different methods of modelling probabilities. We find that a Gaussian model for rating behaviour works well, and with the addition of a Gaussian-Gamma prior it maintains good performance even when data is sparse.

Key words: Recommender systems, Collaborative filtering, Bayesian methods, Naïve Bayes

1 Introduction

Recommender systems are information filtering systems that are widely used on the web to suggest items to users based on their preferences. Collaborative filtering recommenders use item ratings to suggest items preferred by similar users, based on the assumption that people who have agreed in the past will agree in the future[10].

Recommendation accuracy suffers in situations where information is limited[7], such as in distributed[12] or context-aware[1] recommender systems. In addition, some of the more widely used recommender system algorithms such as memory-based collaborative filtering, are rather ad-hoc, and so it is difficult to make princi-

Thomas Barnard
Information: Signals, Images, Systems, School of Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom, e-mail: tcb08r@ecs.soton.ac.uk

Adam Prügel-Bennett
Information: Signals, Images, Systems, School of Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom, e-mail: apb@ecs.soton.ac.uk

pled improvements. Motivated by these challenges, in this paper we present a simple recommender system based on probabilistic methods, which uses prior knowledge to reduce the impact of data sparsity.

After looking at related work in Section 2, we look at making recommendations using Bayes' theorem in Section 4. We then present two models for modelling user ratings, the first based on a multinomial distribution in Section 5, and the second based on a Gaussian distribution in Section 6. Finally we present the results of our experiments in Section 7 before concluding in Section 8.

2 Related Work

Naïve Bayesian techniques have been used to produce recommendations before. Breese et al.[2] present a cluster model using a naïve Bayes classifier, which groups users based on their rating habits, before predicting ratings given cluster membership. Miyahara and Pazzani [9] present a recommender system based on a naïve Bayes model that makes binary rating predictions (i.e. like or dislike). This approach differs to Breese's as they create a separate model for each user. Wang et al.[13] present a probabilistic relevance ranking method that is similar to the item-based collaborative filtering algorithm[11]. They use a Beta distribution to add prior knowledge.

Our approach builds on the simple technique used by Miyahara and Pazzani [9], but we apply the technique to ratings on a numerical rather than binary scale. The naïve Bayes approach is often overlooked in favour of more complex models. In addition, we incorporate prior knowledge into our probability estimates, as Wang et al.[13] do with binary ratings.

3 Notation

Before we present our recommender it is necessary to define the notation we will use in the rest of this paper. We denote the set of all users in our recommender system by U, and the set of all items by I. The rating made by user u on item i is given by $r_{u,i} = k$, where $k \in \mathbf{K}$, the set of possible rating values. $r_u = k$ states that the user u rates any item with value k. The set of items for which user u has made a rating is given by I_u, and the set of users who have made a rating on item i is given by U_i. Finally we define the set of items for which two users u and u' have both provided a rating to be $I_{u,u'} = I_u \cap I_{u'}$.

4 Bayesian Recommendation

Bayes' theorem is a simple probabilistic technique that allows us to update our beliefs about the likelihood of an event occurring given the evidence. These techniques are said to be naïve, in that strong independence assumptions are made about the independence of features. Despite these assumptions naïve Bayes models can achieve good performance[4].

In the case of CF recommenders, our beliefs are the probability that a user will make a rating of a given class on an item, and our features are the ratings made by other users. To simplify calculations we consider priors and likelihoods to be independent of the item of interest, and incrementally update the posterior given each feature,

$$P(r_{u,i} = k | r_{u',i} = k') = \frac{P(r_u = k)P(r_{u'} = k' | r_u = k)}{\sum_{k''} P(r_u = k'')P(r_{u'} = k' | r_u = k'')} \,. \tag{1}$$

Posterior probabilities are combined to find the expected value of the rating $E(r_{u,i})$ as in [2],

$$E(r_{u,i}) = \sum_{k \in \mathbf{K}} P(r_{u,i} = k)k \,. \tag{2}$$

To estimate the priors and likelihoods we first take simple point estimates. Then we use all of our data to create Bayesian priors we can update with more specific information. These priors can be learnt on a server with access to large amounts of information before being passed to devices with more modest resources for distributed recommendation. In the following sections we investigate two different distributions for modelling probabilities using this method.

5 Multinomial Model

The simplest method of obtaining estimates for our priors and likelihoods is by normalising rating counts, which is equivalent to taking maximum likelihood estimates of the parameters of a multinomial distribution,

$$P(r_u = k) = \frac{n_{u,k}}{\sum_{k'} n_{u,k'}} \,, \tag{3}$$

$$P(r_u = k, r_{u'} = k') = \frac{n_{u,u',k,k'}}{\sum_{k''} \sum_{k'''} n_{u,u',k'',k'''}} \,, \tag{4}$$

where $n_{u,k}$ is the number of times user u has given an item a rating k, and $n_{u,u',k,k'}$ is the number of times user u has rated k, when user u' rated k'. Given these probabilities calculating the likelihood is trivial.

Where rating counts are zero, these probabilities will be zero, so to remove these zeros we apply Laplace smoothing, adding one to each of our rating counts. This is the model used by Miyahara and Pazzani[9].

5.1 Dirichlet Prior

Prior knowledge is incorporated into our multinomial model using a Dirichlet distribution, parameterised by $\alpha = \alpha_i, \ldots, \alpha_k > 0$, which correspond to the number of times a particular outcome k has been observed. Laplace smoothing is a simple case of this where each parameter is set to one.

To obtain initial parameters we make use of a fixed-point iteration, described in [8]. To update these parameters for each specific case, we simply add the rating counts,

$$\alpha_u = \alpha + \mathbf{n}_u \; . \tag{5}$$

Parameters for our multinomial distribution, can then be obtained by taking the mean of the Dirichlet distribution,

$$E(\mathbf{p}) = \frac{\alpha}{\sum_k \alpha_k}. \tag{6}$$

Adding a Dirichlet prior seriously degraded the performance of the multinomial model. We attempted to improve the model by taking into account variance in the Dirichlet distribution by performing a stochastic expansion about the mean, but this only slightly improved results, so the details are not given here.

6 Gaussian Model

For our next model we decided to look at the differences in user ratings. We model these differences $r_u - r_{u'}$ as being drawn from a Gaussian distribution. Our model is not strictly Gaussian, as we make some simplifications removing constants, and our likelihoods are discrete, rather than continuous. The formula is

$$P(r_{u'} = k' | r_u = k) = \frac{exp(-\tau_{u,u'}/2\left(k - k' - \mu_{u,u'}\right)^2)}{\sum_{k''} exp(-\tau_{u,u'}/2\left(k'' - k' - \mu_{u,u'}\right)^2)} \; , \tag{7}$$

where $\mu_{u,u'}$ is the mean difference between the two user's ratings, and $\tau_{u,u'}$ is the precision of the Gaussian distribution, or σ^{-2} the reciprocal of the variance. We obtain values for the mean and precision through maximum likelihood estimates,

$$\hat{\mu}_{u,u'} = \frac{1}{|I_{u,u'}|} \sum_i \left(r_{u,i} - r_{u',i}\right) \; , \tag{8}$$

$$\hat{\sigma}_{u,u'} = \frac{1}{|I_{u,u'}|} \sum_i (r_{u,i} - r_{u',i} - \mu_{u,u'})^2 \,, \tag{9}$$

$$\hat{\tau}_{u,u'} = \frac{1}{\hat{\sigma}_{u,u'}} \,. \tag{10}$$

In cases where there are few ratings, such that $\sigma^2 = 0$, we set the precision to zero.

We can augment this model with a Gaussian-Gamma prior. Mean and variance are treated as unknown, with mean modelled by a Gaussian, and variance by a Gamma distribution. It has the following probability density function,

$$GG(\mu, \tau | \mu, \kappa, a, b) \sim N(\mu | \mu, \kappa\tau^{-1}) \Gamma(\tau | a, b) \,. \tag{11}$$

We obtain formulae to calculate its parameters by looking at the marginal distributions of μ and τ,

$$P(\tau) = \Gamma(a, b) \,, \tag{12}$$

$$P(\mu) = T_{2\alpha}(\mu, \frac{a\kappa}{b}) \,, \tag{13}$$

where T is a Student's T distribution[3]. Note that we use the parameterisation of the Gamma distribution where a is the shape parameter, and b is the rate parameter.

We have maximum likelihood estimates for the mean and variance of τ, and using the properties of the gamma distribution, we can derive these estimates for its parameters,

$$a_0 = \frac{\hat{\mu}_\tau^2}{\hat{\sigma}_\tau^2} \,, \tag{14}$$

$$b_0 = \frac{\hat{\mu}_\tau}{\hat{\sigma}_\tau^2} \,. \tag{15}$$

We use a similar procedure using the properties of Student's T distribution to obtain an estimate for κ_0,

$$\kappa_0 = \frac{b_0(a_0 - 1)}{\hat{\sigma}_\mu^2} \,. \tag{16}$$

Finally we set μ_0 to zero as our matrix of differences contains $r_{u,i} - r_{u',i}$ as well as $r_{u',i} - r_{u,i}$, these differences cancel out.

Once we have prior values for our parameters, we can perform a Bayesian update using the following equations, which can be found in [3],

$$\mu_n = \frac{\kappa\mu + n\hat{\mu}_{u,u'}}{\kappa_0 + n} \,, \qquad \kappa_n = \kappa_0 + n \,, \qquad a_n = a_0 + \frac{n}{2} \,,$$

$$b_n = b_0 + \frac{1}{2} \sum_{i=1}^{n} (r_u - r_{u'} - \hat{\mu}_{u,u'})^2 + \frac{\kappa_0 n (\hat{\mu}_{u,u'} - \mu_0)^2}{2(\kappa_0 + n)} \,.$$

We obtain parameters for our Gaussian by taking the expected values of the mean and precision from our posterior distribution $GG(\mu, \tau | \mu_n, \kappa_n, a_n, b_n)$,

$$E(\mu) = \mu_n \,, \tag{17}$$

$$E(\tau) = \frac{a_n}{b_n} \,. \tag{18}$$

7 Experiments

To compare the performance of the techniques described in this paper, we implemented several basic recommender system algorithms in Python[1]. We implemented simple recommenders based on using average user ratings and average item ratings to make predictions. We also implemented memory-based CF[2] using Pearson correlation as our similarity measure, and use a fixed neighbourhood size of the 500 most similar users in making predictions, as in our experience this provides the best results. Significance weighting, which weights the effect of users with more items in common, was used with a threshold of 50, as suggested in [5].

The dataset used for our experiments is the MovieLens[2] 100,000 rating dataset, which contains a collection of ratings made by users on films. Ratings are made on a 5-star integer scale. We use this dataset for its popularity which makes comparison with existing methods easier. The dataset is 94 % sparse.

For each experiment we perform 5-fold cross-validation, splitting the dataset randomly by rating, to produce training sets containing 80 % of the ratings, and test sets containing 20 % of the ratings. The same sets are used to test each algorithm. The results of each evaluation are averaged across the five runs.

Our second experiment looks at how these techniques perform under different levels of data sparsity. After splitting each dataset for k-fold cross validation, a varying proportion of the ratings are removed randomly. The results are then tested against the full dataset.

7.1 Evaluation Metrics

We use a small number of commonly used metrics to aid comparison with other techniques, and to give an overall picture of recommender system performance. Recommender system evaluation metrics fall into three main categories: coverage, statistical accuracy, and decision support[5]. In [6] decision support metrics are divided into classification-accuracy metrics, and ranking-accuracy metrics.

We use mean-absolute-error (MAE) to measure statistical accuracy, and the F1 measure to measure classification accuracy. As our ratings are not binary, we transform them by considering ratings greater than or equal to four to be positive, or relevant[5]. We do not use a coverage metric, because we find it does not give use-

[1] http:://www.python.org

[2] http://www.grouplens.org/

ful results. We also do not use a ranking metric, because although the system will return predictions with many different ranks, the ratings system only allows five.

7.2 Results

The results of this experiment are presented in descending order of MAE in Table 1. MAE is presented along with its standard error, and F1 score. Lower values of MAE are better, and higher F1 scores are better. We use a paired t-test at a level of 5 % to test the significance of MAE differences. Methods shown in bold are significantly different from the method below them. Most of the method names are self-explanatory. The variants of the Dirichlet method are corrected and uncorrected, PCC MBCF is memory-based CF using Pearson correlation, with and without significance weighting.

The Gaussian methods perform best on MAE, with the Dirichlet augmented multinomial models performing worst. On F1 score, the Gaussian models come out on top, but in contrast to MAE, the Dirichlet models outperform MBCF. In addition to the Gaussian results shown below which use a Dirichlet model for prior probabilities we tried Gaussian and Gaussian-Gamma priors. We found that these models did not produce significantly different results from the Dirichlet model, so they are not shown here.

Table 1 Results

Method	MAE	SE	F1 Score
Gaussian	0.7054	0.0042	0.6931
Gaussian-Gamma	**0.7059**	0.0042	0.6932
PCC MBCF (Weighted)	**0.7393**	0.0039	0.5613
Laplace	0.7438	0.0043	0.6566
PCC MBCF	**0.7438**	0.0038	0.5481
Item Average	**0.8183**	0.0041	0.3938
User Average	**0.8351**	0.0042	0.2993
Dirichlet (Corrected)	**0.8525**	0.0041	0.5881
Dirichlet	0.8649	0.0045	0.6196

7.3 Sparsity

We looked at a subset of the techniques which performed well in the general experiment for the sparsity experiment. We tested the averages, MBCF with significance weighting; Laplacian, Gaussian, and Gaussian-Gamma naïve Bayes. We tried

all variants for the Gaussian models, but found that the Dirichlet model for priors worked best, and so those are the only results reported here for the sake of clarity.

Figure 1 shows MAE against the relative density of the dataset, and Figure 2 shows F1 score against relative density. These graphs show a gradual improvement in performance as more of the original dataset is used. Each techniques maintains its relative performance compared with other techniques for most of the MAE graph. At conditions of extreme sparsity the simple multinomial technique slightly outperforms the others, it is not obvious why this should be so. It is also interesting to note that the Gaussian-Gamma technique, which showed little improvement over the simple Gaussian technique in the general tests, outperforms the Gaussian technique as the data becomes more sparse. It is likely that the prior helps to fill in gaps in its knowledge.

The picture is much the same for the F1 measure, although in this case the Gaussian-Gamma technique retains excellent levels of performance even when using the most sparse dataset. It is interesting to note that the performance of the Laplace smoothed algorithm suffers considerably when data is sparse. It is also interesting that increasing data density actually degrades the performance of the average methods.

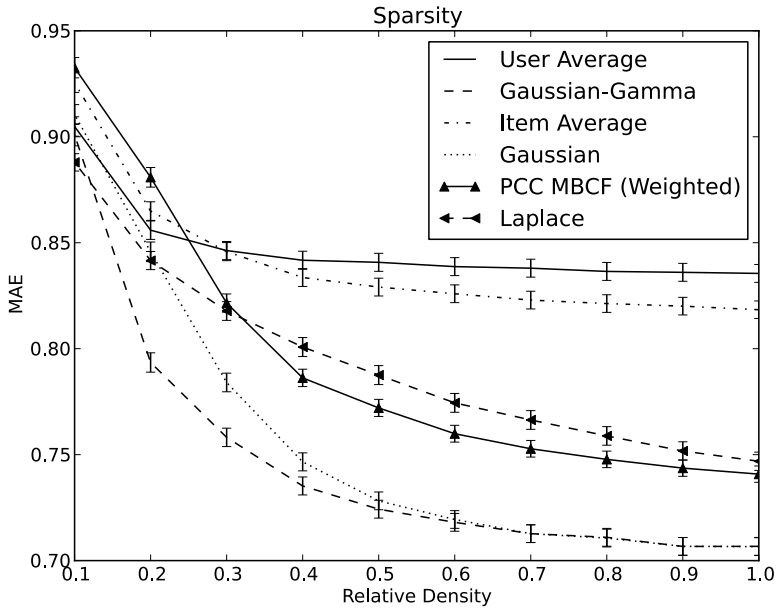

Fig. 1 MAE under conditions of varying sparsity

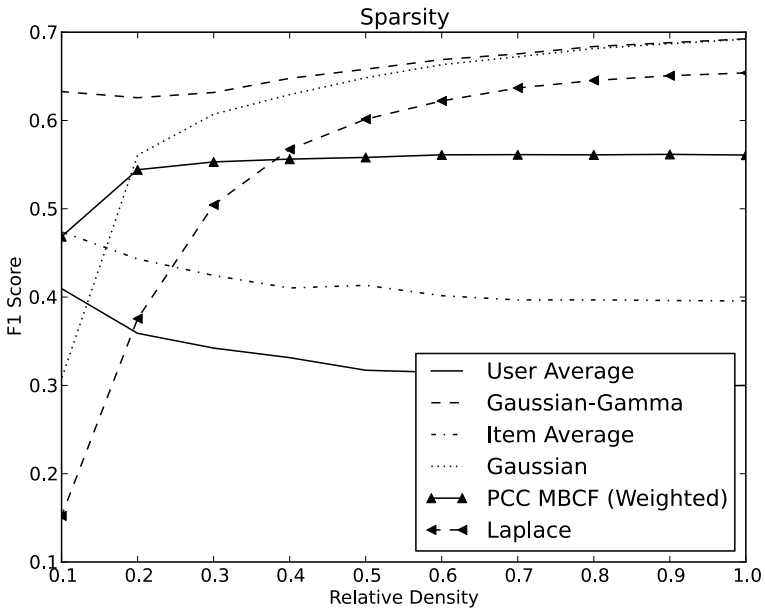

Fig. 2 F1 score under conditions of varying sparsity

8 Conclusions

In this paper we have shown that Bayesian recommenders making use of prior knowledge can produce results which are better than those used from memory-based collaborative filtering, and simple probabilistic recommenders not using prior knowledge. We have shown that these techniques maintain good levels of performance even when using sparse data. In particular we find that our Gaussian model produces the best results across most of our tests. Although the Gaussian-Gamma prior was found to perform similarly to the Gaussian model under conditions of relative data density, under sparser conditions it performed better. For our prior probabilities however, we found that the Dirichlet model produced better results.

In the case of the multinomial model, the addition of a prior is found to be harmful to the model. We believe this because we are trying to fit too many parameters given the limited information. In addition the multinomial model makes assumptions about the independence of rating categories which are unlikely to hold true. In cases where the underlying model is a better fit to the data, the addition of a prior appears to help fill in gaps in information leading to better performance in situations where data is sparse.

8.1 Future Work

Our next task is to do a more thorough comparison of our method with existing more complex methods of probabilistic, and sparse recommendation techniques. We also need to apply our technique to a wider range of datasets. As one of our goals is distributed recommendation, we also need to test our method in a distributed situation.

Although we found the Gaussian-Gamma model works well, we still wish to investigate other models. Multivariate Gaussian, or Gaussian mixtures are the next logical steps from this model, but they have more parameters and so might suffer from the same problems as the Dirichlet model. Another approach would be to stay with the Gaussian model, but learn clusters of users, applying different prior knowledge to different groups.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005). DOI http://doi.acm.org/10.1145/1055709.1055714
2. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: UAI, pp. 43–52. Morgan Kaufmann (1998)
3. DeGroot, M.H.: Optimal Statistical Decisions. McGraw-Hill Book Company (1970)
4. Hand, D.J., Yu, K.: Idiot's bayes—not so stupid after all? International Statistical Review **69**(3), 385–398 (2001)
5. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 230–237. ACM, New York, NY, USA (1999)
6. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
7. Maltz, D., Ehrlich, K.: Pointing the way: active collaborative filtering. In: CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 202–209. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1995). DOI http://doi.acm.org/10.1145/223904.223930
8. Minka, T.P.: Estimating a dirichlet distribution. Tech. rep., Microsoft (2003)
9. Miyahara, K., Pazzani, M.J.: Collaborative filtering with the simple bayesian classifier. In: In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence, pp. 679–689 (2000)
10. Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proc.Computer Supported Cooperative Work Conf. (1994)
11. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW '01: Proceedings of the 10th international conference on World Wide Web, pp. 285–295. ACM, New York, NY, USA (2001)
12. Tveit, A.: Peer-to-peer based recommendations for mobile commerce. In: WMC '01: Proceedings of the 1st international workshop on Mobile commerce, pp. 26–29. ACM, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/381461.381466
13. Wang, J., Robertson, S., Vries, A.P., Reinders, M.J.: Probabilistic relevance ranking for collaborative filtering. Inf. Retr. **11**(6), 477–497 (2008)