Structured Collaborative Filtering in Recommender Systems

Abstract

Recommendation systems are widely used in world leading corporations such as Google, Facebook and Netflix. These systems provide us with relevant ads and recommendations that help us make the most out of our time. In this project, we compare two different approaches to the recommendation problem - matrix factorization models and graphical models. We implement Gaussian Matrix Factorization (GMF), Hierarchical Poisson Factorization (HPF) and SVD++ which are matrix factorization models, and Markov Random Fields and Ordinal Random Fields (ORF) which are graphical models. We then compare the different models in the field of movie recommendation. Our hypothesis is that graphical models outperform matrix factorization methods due to a better interaction representation but take more time to train. The results show that they outperform them in both recommendations and training time.

1 Introduction

A recommender system is a system that seeks to predict the rating or preference a user would give to an item. This is a structured task because the output domain is the set of all rating matrices $\mathcal{R} \in \mathbb{R}^{U \times I}$ where U and I are the quantities of users and items in the system, respectively. The most widely used approach to the design of recommender systems is collaborative filtering. Collaborative filtering is a method of making predictions about the interests of a user by collecting preferences from many users. The underlying assumption of this approach is that if a person A shares opinion with person B on an issue, then person A is more likely to share person B's opinion on a different issue than that of a randomly chosen person.

The goal of recommender systems is to predict whether a particular user would prefer an item or not based on the user's profile. To achieve this goal we employ a variety of collaborative filtering methods. We wish to compare the accuracy and performance of matrix factorization models to graphical models. We hypothesize that graphical models achieve better predictions than matrix factorization models for the price of computation complexity [1].

Basic Part We use several matrix factorization methods. Matrix factorization methods are used to decompose the incomplete ratings matrix into two latent factor matrices, for users and items. These matrices are then multiplied to produce the full ratings matrix. For that purpose, we use the SVD++ factorization [2], Gaussian [3] and Poisson [4] probabilistic matrix factorization.

Advanced Part Following our hypothesis that graphical models could improve recommendation performance, we implement two graphical models. We use Markov Random Fields[5] to model the item-item interactions, and Ordinal Random Fields[6] to model the rating distribution.

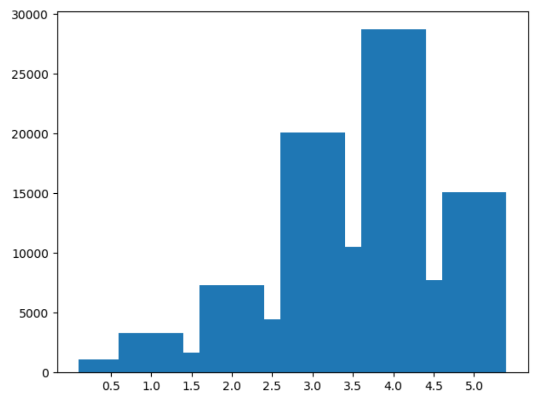
Creative Part Retrieval lists fusion has been shown [7] to improve retrieval performance. We fuse the retrieval lists of Hierarchical Poisson Factorization (HPF) and SVD++ to improve the individual performances.

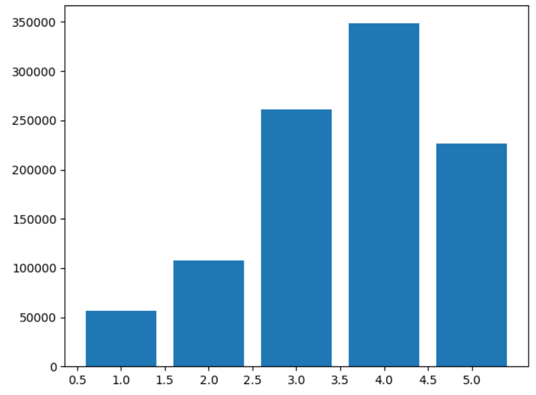
Results and Conclusions Contrary to our initial hypothesis we achieved better performance using graphical models, without increasing training time. Thus we conclude that graphical models show a promising future in collaborative filtering.

2 MovieLens Datasets

	No. Records	Unique Users	Unique Movies	Sparsity	Release Date
MovieLens 100K	100,000	1000	1700	5.88%	4/1998
MovieLens 1M	1,000,000	6000	4000	4.16%	2/2003

The users in both datasets were selected at random and thus are independent of each other. Every entry in the dataset is a tuple (u, i, r) where r is the rating user u gave to movie i. u and i are unique user and movie identifiers. We chose to use the MovieLens dataset because it is widely used in recommender system research papers.
The MovieLens 1M dataset has only integer ratings (1-5) while the MovieLens 100K dataset has also intermediate ratings (0.5, 1.5, ...).
We could not evaluate our model on the larger versions of the MovieLens data due to lack of computation resources.


(a) Rating histogram for MovieLens 100K


(b) Rating histogram for MovieLens 1M

3 Basic Part - Matrix Factorization

3.1 Gaussian Factorization

Notation	
r_{ui}	Rating of user u to item i
θ_u	Latent factors of user u
β_i	Latent factors of item i
K	Number of latent factors
$\lambda_\theta, \lambda_\beta$	Inverse standard deviation of θ_u, β_i, respectively
p_{ui}	Propensity score of user u for item i
$\mathcal{D}$	Dataset of rating samples $(user, item, rating)$

The method assumes latent Gaussian models on both the user intrinsic scores and the item intrinsic scores while assuming diagonal normal priors on the latent factors

$$r_{ui}|e_{ui} = 1 \sim \mathcal{N}\left(\theta_u^T \beta_i, \lambda_y^{-1}\right) \qquad \theta_u \sim \mathcal{N}\left(0, \lambda_\theta^{-1} I_K\right) \qquad \beta_i \sim \mathcal{N}\left(0, \lambda_\beta^{-1} I_K\right)$$

We minimize the following regularized loss function using an efficient iterative EM algorithm.[3]

$$\theta_u \leftarrow \left(\sum_{m:(u,i,r)\in\mathcal{D}} \frac{1}{p_{ui}} \beta_i \beta_i^T + \lambda_\beta I_K \right)^{-1} \sum_{i:(u,i,r)\in\mathcal{D}} \frac{1}{p_{ui}} r_{ui} \beta_i$$

$$\beta_i \leftarrow \left(\sum_{u:(u,i,r)\in\mathcal{D}} \frac{1}{p_{ui}} \theta_u \theta_u^T + \lambda_\theta I_K \right)^{-1} \sum_{u:(u,i,r)\in\mathcal{D}} \frac{1}{p_{ui}} r_{ui} \theta_u$$

Predictions are formed by $\hat{r}_{ui} = \mathbb{E}\left[r_{ui}|e_{ui} = 1\right] = \theta_u^T \beta_i$.

3.2 Hierarchical Poisson Factorization

We place Gamma priors on the latent attributes and latent preferences, which encourage the model towards sparse representations of the users and items. Furthermore, we place additional priors on the user and item rate parameters:

$$\xi_u \sim Gamma\left(a', \frac{a'}{b'}\right) \qquad \theta_{uk} \sim Gamma\left(a, \xi_u\right)$$

$$\eta_i \sim Gamma\left(c', \frac{c'}{d'}\right) \qquad \beta_{ik} \sim Gamma\left(c, \eta_i\right)$$

ξ_u is the user activity rate (the rate in which a user tends to rate items) and η_i is the item popularity rate (the rate in which a item tends to be watched). A benefit of such hierarchical parameterization is capturing the diversity of users and items - some users tend to rate more items and some items are generally watched more often. Furthermore, the distributions of real-world user activity and item popularity tend to be long tailed, a behavior that is better captured by HPF. Under these assumptions,

$$r_{ui}|e_{ui} = 1 \sim Poisson\left(\theta_u^T \beta_i\right)$$

Predictions are formed by $\hat{r}_{ui} = \mathbb{E}\left[r_{ui}|e_{ui} = 1\right] = \theta_u^T \beta_i$.
We fit the latent features using an iterative EM algorithm which learns the latent parameters of both users and items while choosing prior parameters using cross-validation.[4]

3.3 SVD++

The SVD++ algorithm is an extension to the Singular Value Decomposition, taking into account implicit ratings.[2]
The prediction is set as $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j\right)$
Where I_u are the items that user u has rated and the y_j terms are a new set of item factors that capture implicit ratings. Here, an implicit rating describes the fact that a user u rated an item j, regardless of the rating value. If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i, q_i and y_i.

4 Advanced Part - Graphical Models

4.1 Ordinal Random Fields

4.1.1 Ordinal Matrix Factorization

Previous methods heavily rely on numerical preferences, whereas the importance of ordinal preferences wasn't fully recognized[6]. In general OMF aims to generate an ordinal distribution $Q\left(r_{ui}|u, i\right)$ over all possible rating values for each user-item pair. Using that distribution for rating prediction is done by choosing $\hat{r}_{ui}$ that maximizes $Q\left(\hat{r}_{ui}|u, i\right)$. The model assumes latent factors on the users and items, p_u and q_i, respectively, while modeling the user interest in items with a latent utility

$$x_{ui} \sim Logi(b_{ui} + p_u^T q_i, s_{ui})$$

The ratings in our data are discrete values that are naturally ordinal. Therefore, we perform the ordinal assumption

$$r_{ui} = l \text{ if } x_{ui} \in (\theta_{l-1}, \theta_l] \text{ for } l < L \text{ and } r_{ui} = L \text{ if } x_{ui} > \theta_{L-1}$$

where L is the number of ordinal levels and θ_l are the thresholds (e.g. (0.5,1], (1,1.5], ...).
The probability of receiving a rating l is

$$Q\left(r_{ui} = l|u, i\right) = \int_{\theta_{l-1}}^{\theta_l} \mathbb{P}\left(x_{ui}|\theta\right) dx_{ui} = F\left(\theta_l\right) - F\left(\theta_{l-1}\right)$$

where $F\left(\theta_l\right)$ is the cumulative logistic distribution evaluated at θ_l.

4.1.2 Markov Random Fields

We aim to use a combination of MRF and OMF to exploit both of their advantages. While MRF successfully models localized interactions, OMF captures the higher order interactions among users and items. This combination results in a joint distribution over all ratings

$$\mathbb{P}(\mathcal{R}) \propto \prod_{u} \Psi_u(\mathbf{r}_u) \prod_{u,i} Q(r_{ui}|u,i) \qquad \Psi_u(r_u) = \exp\left(\sum_{r_{ui},r_{uj}\in\mathbf{r}_u} w_{ij} f_{ij}(r_{ui}, r_{uj})\right)$$

where $f_{ij}(r_{ui}, r_{uj})$ are the item-item correlation features[1] and w_{ij} are the corresponding weights. These correlations are captured by $f_{ij}(r_{ui}, r_{uj}) = g(|(r_{ui} - \bar{r}_i) - (r_{uj} - \bar{r}_j)|)$ where g is the sigmoid function. We eventually learn the MRF structure by maximizing the pseudo-likelihood

$$\mathcal{L}(\mathbf{w}) = \sum_{r_{ui}\in\mathcal{R}} \log \mathbb{P}(r_{ui}|\mathbf{r}_u \setminus r_{ui}) - \frac{1}{2\sigma^2} \sum_{u\in\mathcal{U}} \mathbf{w}_u^T \mathbf{w}_u$$

The prediction is given by $\hat{r}_{ui} = \arg\max_{r_{ui}} \mathbb{P}(r_{ui}|\mathbf{r}_u)$.

4.2 Markov Random Fields

We use an MRF to model the dependencies among the items that are recommended to a user, while a user corresponds to a sample drawn from the distribution of the MRF. The nodes in the MRF correspond to the $m = |\mathcal{I}|$ items and are associated with a vector of random variables $X = (X_1, ..., X_m) \sim \mathcal{N}\left(\vec{0}, \Sigma\right)$. A user corresponds to a sample drawn from this distribution. The predicted score for item i given a user's observed interactions with all *other* items, $x_{\mathcal{I}\setminus\{i\}}$, is given by the expectation

$$\mathbb{E}\left[X_i | X_{\mathcal{I}\setminus\{i\}} = x_{\mathcal{I}\setminus\{i\}}\right] = \sum_{j\in\mathcal{I}\setminus\{i\}} \beta_{j,i} x_j = x \cdot \mathbf{B}_{\cdot,i}$$

where $\mathbf{B} \in \mathbb{R}^{m\times m}$ is a matrix of item-item correlation features with a zero diagonal, $\mathbf{B}_{i,i} = 0$, as to exclude X_i when computing the expectation and to avoid self-similarity of items.

Following the experimental setting in [5], we use a binary interaction matrix $\mathbf{X}$ where $\mathbf{X}_{ui} = 1$ indicates that user u has rated item i. We implement Besag's pseudo-likelihood which yields asymptotically consistent estimates when the auto-normal parameterization is used. Adding L2-norm regularization with hyper-parameter $\lambda > 0$ results in

$$\hat{\mathbf{B}} = \arg\min_{\mathbf{B}} \|\mathbf{X} - \mathbf{X}\mathbf{B}\|_F^2 + \lambda \|\mathbf{B}\|_F^2 \quad \text{s.t.} \quad \text{diag}(\mathbf{B}) = \mathbf{0}$$

Solving with Lagrangian multipliers yields the closed-form solution

$$\hat{\mathbf{B}} = \mathbf{I} - \hat{\mathbf{C}} \cdot \text{dMat}(1 \oslash \text{diag}(\hat{\mathbf{C}})) \qquad \hat{\mathbf{C}} = \left(n^{-1}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)\right)^{-1}$$

where $\oslash$ is the element-wise division operator, n is the number of users and dMat is a diagonal matrix.

Note that the recommendation accuracy highly depends on the popularity bias in the data. The different item popularities affect the means and covariances in the Gaussian MRF, but the popularity bias is expected to be similar in the training and test data because of the way the data was split (Section 6).

5 Creative Part - Fusion

Retrieval lists fusion was long studied as a method to improve retrieval performance[7]. The three key reasons that a fusion should work are:

- **The skimming effect** – A user is likely to be satisfied by top ranked items.
- **The chorus effect** – Agreement between retrieval lists implies higher probability for an item to be relevant.
- **The dark horse effect** – Different retrieval methods have different objectives and therefore might come across different outliers. When fusing different lists it is possible to introduce relevant outliers that were detected by only one method.

We attempt to fuse the recommendation retrieval lists of HPF and SVD++ by applying linear fusion with uniform weights. We normalize the ratings with min-max normalization.

[1] We used the 10 most correlated pairs of items according to Pearson correlation, per user.

6 Experimental Settings

For every user in the dataset we split its rated items into training (60%), validation (20%) and test (20%) sets. We then perform 5-Fold cross validation and choose the best hyper-parameters by evaluating the validation set. Finally, we present MAE, RMSE, nDCG@100 (averaged on all users) metrics for the test set, averaged on all folds.
For the MRF model, we split the users to disjoint training (60%), validation (20%) and test (20%) sets. Following, we further split the items of the validation and test. We use 80% of the items to learn the user representation using the trained model, and the rest (20%) for performance evaluation. We evaluate the model using only nDCG@100 (averaged on all users) because the model predicts recommendations and not ratings and as such should be evaluated as a retrieval system, rather than a prediction problem.

$$\text{MAE} = \frac{1}{|\mathcal{R}|}\sum_{r_{ui}\in\mathcal{R}} |\hat{r}_{ui} - r_{ui}| \quad \text{RMSE} = \sqrt{\frac{1}{|\mathcal{R}|}\sum_{r_{ui}\in\mathcal{R}} (\hat{r}_{ui} - r_{ui})^2} \quad \text{nDCG@k}^1 = \frac{\sum_{i=1}^{k}\frac{\text{rel}_i}{\log_2(i+1)}}{\text{IDCG@k}}$$

7 Results

		GMF	HPF	SVD++	MRF[2]	ORF	Fusion[3]
MovieLens 100K	MAE	0.764	0.748	**0.737**	–	3.544	–
	RMSE	0.967	0.945	**0.936**	–	3.698	–
	nDCG@100	0.382	0.179	0.226	**0.391**	–	0.203
	Runtime	45s	15s	1.5m	**10s**	–	0s
MovieLens 1M	MAE	0.730	0.725	**0.692**	–	–	–
	RMSE	0.939	0.916	**0.883**	–	–	–
	nDCG@100	0.180	0.103	0.163	**0.202**	–	0.147
	Runtime	6m	2m	20m	**20s**	–	0s

The ORF model defines features for every item-item interaction which results in a huge amount of model parameters. We attempt to decrease the amount of features by selecting the top 10 Pearson-correlated features for every user. Despite our efforts, training the model takes about 5 hours per iteration on the MovieLens 100k (the smaller) dataset. We couldn't manage to train the model until convergence but we believe that with enough computation resources, the model could perform well.

8 Conclusions

In this project we investigated the task of movie recommendations and compared performance of baseline probabilistic matrix factorization to that of models utilizing graphical dependency structure. We evaluated the different models on the MovieLens datasets and concluded that graphical models could be superior both in training time and recommendation quality.
Although the prediction accuracy measures of GMF are the lowest, its nDCG measure is comparable to that of the MRF. We presume that this result is achieved by the inverse propensity weighting (IPW) done in GMF, which further models user-item interactions.
Using the recommendations of SVD++ we achieved improved nDCG@100 for HPF with fusion. SVD++ has high runtime, but it can be used at a lower frequency than HPF to improve the recommendations of HPF (e.g. Run HPF daily and SVD++ weekly then use slightly outdated SVD++ recommendations for fusion with HPF).
To conclude, this is the second project in which we research recommendation systems and it was exciting to experience new approaches to this problem.

[1]IDCG is the ideal DCG, achieved by the optimal ranking.
[2]MRF predicts recommendations and not ratings and thus we cannot compute MAE or RMSE, the sums of deviations from the real ratings.
[3]Fusion of HPF and SVD++ recommendation lists

References

[1] L. Sharma and A. Gera, “A survey of recommendation system: Research challenges,” *International Journal of Engineering Trends and Technology (IJETT)*, vol. 4, no. 5, pp. 1989–1992, 2013.

[2] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434, 2008.

[3] D. Liang, L. Charlin, and D. M. Blei, “Causal inference for recommendation,” in *Causation: Foundation to Application, Workshop at UAI*, AUAI, 2016.

[4] P. Gopalan, J. M. Hofman, and D. M. Blei, “Scalable recommendation with poisson factorization,” *arXiv preprint arXiv:1311.1704*, 2013.

[5] H. Steck, “Markov random fields for collaborative filtering,” in *Advances in Neural Information Processing Systems*, pp. 5473–5484, 2019.

[6] S. Liu, T. Tran, and G. Li, “Ordinal random fields for recommender systems,” in *Asian Conference on Machine Learning*, pp. 283–298, 2015.

[7] C. Vogt and Vogt, “Fusion via a linear combination of scores 1,” 2014.