Sparse Estimation of Movie Preferences via Constrained Optimization

Alexander Anemogiannis, Ajay Mandlekar, Matt Tsao

December 17, 2016

Abstract

We propose extensions to traditional low-rank matrix completion methods used in movie recommendation systems, with the ability to recommend out-of-sample movies. We have implemented three recommendation systems; the first is a standard SVD model which learns abstract features for prediction and generalizes to out-of-sample movies using a nearest neighbor search. The second model is a constrained matrix completion using interpretable features instead of latent features. The third model is constrained matrix completion using LASSO regularization. We found that model 3 generalizes best due to the intrinsic sparsity of user preferences.

Related Work

Matrix completion and collaborative filtering techniques have been popular and successful in predicting user ratings for movies based on the ratings of many similar users [3]. However, these methods fail to provide any insight as to which factors users take into consideration when making choices, since these models often learn latent factor representations, for which there is no natural interpretation [1]. The idea behind collaborative filtering techniques is to leverage the known scores of similar users to fill in missing ratings [2]. Therefore, these techniques cannot be used to predict ratings on a movie which none of the users in the dataset have seen. Such a movie is called *out-of-sample*. Predicting the reception of out of sample movies can be of great interest to content creators because it allows them to cater towards user preferences.

Introduction

One weakness in using latent factors is that it is unclear how to find the latent feature representation for an out-of-sample movie. Thus, to make the feature space generalizeable to new movies, we use an intepretable feature space. We also suspect that for an interpretable feature space, user preferences will be sparse. For example, a person may choose movies heavily based on their favorite genre or favorite actor, without paying too much attention to the director, runtime, release year, or movie rating. Sparse preference vectors are also easier to interpret and localize which factors a particular user sees as most important. Thus from both an operational perspective and an interpretation perspective, we believe the problem structure is intrisically sparse, which is confirmed in our experimental results. Inspired by this, we propose three approaches to address the issues of interpretablility and extension to out-of-sample movies, all of which utilize an interpretable feature space instead of using latent features. The inputs and outputs for all three approaches is the same: for a given movie ID and a specified user, we output a score estimate from 1 to 5 indicating how much we believe the user will like the movie. In section 1, we introduce our data set and our construction of an interpretable feature set. Section 2 describes in detail all three of our approaches. We report the results of our experiments and compare with related works in Section 3 and conclude with a discussion and future ideas in Section 4.

Notation

- The **feature size** k is the number of features we use to make recommendations. k is a positive integer. The regimes where $k \ll m$ and $k \ll n$ is what we are considering.
- A **movie feature vector** is a vector of k parameters that we use to represent the properties of a movie.
- Similarly, a **user feature vector** is a vector of k parameters that we use to represent the corresponding emphasis or preference toward a particular movie property.
- The **User Matrix** $U \in \mathbb{R}^{m \times k}$ is a matrix which describes the preferences of m users. The ith row of U is the user feature vector for user i.
- The **Movie/Media Matrix** $V \in \mathbb{R}^{n \times k}$ is a matrix with describes the attributes of the media files in our database. The ith row of V is the movie feature vector for movie j.
- For a vector v, we use the notation v_i to represent the ith component/entry of v.
- For a matrix A, we use the notation that A_{i*} is the ith row of A, and A_{*j} is the jth column of A.

1 Data set and Features

For our experiments, we used the 20M dataset from Movielens [10] which contains 20 million ratings of 27,000 movies by 138,000 users.[1]. We chose the 20M dataset because it has the largest sample variance and would give our techniques the most opportunity to demonstrate their effectiveness. We wrote a python script to extract information from IMDB for all of the movies in the 20M set about the genre, runtime, release date, rating (G, PG, etc.), IMDB score, leading actors and directors and language in the movie. We then created a feature mapping $\Psi : \mathbb{N} \to \mathbb{R}^k$ which takes a movie ID/index and produces the corresponding feature vector. For a given movie index v, $\Psi(v)$ is constructed as a concatenation of indicator functions. For example, we distinguished between 22 different genre types, so the genre portion of $\Psi(v)$ will have 22 components, each of which will have a value of 1 if the movie indexed by v falls into that genre category, and 0 otherwise. This pre-processing was done in MATLAB, however, the 20M MovieLens set was too large for MATLAB to import, so we had to implement most of our learning algorithms from scratch in C (with the exception of LASSO). We developed several optimizations to the code in order to speed up training, including the use of multithreading and emphasizing sequential memory accesses (to help with cache locality). Although actors and directors certainly influence a user's preference, we couldn't find a way of encoding them without blowing up our feature dimension. We used genre, rating, year of release, and runtime, each of which has dimension 22, 7, 10, and 5, to define Ψ with 44 features.

2 The Models

The following notation is for all three models. We have a database with m users and n movies, and for movies in the database, our estimate of the score that the ith user would give the jth movie is $U_{i*}V_{j*}^T + c_{ij}$ where c_{ij} is a bias term depends on i, j. Therefore, the matrix $UV^T \in \mathbb{R}^{m \times n} + C$ is such that the (i, j) entry is our estimate of the score that the ith user gives to the jth movie. The training data of size S has the form $\mathcal{S} = \{y_i, u_i, v_i\}_{i=1}^S$, which is to be interpreted as the u_ith user gave the v_ith movie a rating of y_i.

2.1 Model 1: Biased SVD & Nearest Neighbor

In this first model, in addition to the matrices U, V, our model uses two bias vectors $b \in \mathbb{R}^m$ and $b' \in \mathbb{R}^n$. Given a particular pair $1 \leq i \leq m, 1 \leq j \leq n$ of a user and a movie, our estimate of the score that i would give j is

$$\widehat{s}_{ij} := U_{i*}V_{j*}^T + b_i + b'_j \tag{1}$$

[1]The data set we used can be found here: http://grouplens.org/datasets/movielens/

We also employ ℓ_2 regularization to the entries of U and V, to prevent overfitting, therefore the objective function for our algorithm is:

$$L(U, Vb, b'; \mathcal{S}) = \lambda_U ||U||_F^2 + \lambda_V ||V||_F^2 + \sum_{i=1}^{S} \left(y_i - U_{u_i *} V_{v_i *}^T - b_{u_i} - b'_{v_i}\right)^2$$

Where λ_U, λ_V are the ℓ_2 regularization constants, and $||\cdot||_F$ is the Frobenius norm for matrices. We minimized this objective using stochastic gradient descent. (1) is then used to estimate the scores of all movies by all users in the database. To handle recommendation of out-of-sample movies, we proceed using a nearest neighbor approach. For an out-of-sample movie v, even though it is not in the database, we can still compute $\Psi(v)$, since the features comprising Ψ are interpretable statistics. Next we find the indices $N(v)$ of its L nearest neighbors, in the Ψ-feature space.

$$N(v) = \arg \min_{\substack{N \subset [n], \\ |N| = L}} \sum_{i \in N} ||\Psi(v) - \Psi(v_i)||_2 \tag{2}$$

Then if we wish to predict the score that user i would give to v, we take an average of the scores that i gave to the movies in $N(v)$.

$$\widehat{s}_i(v) := \frac{1}{L} \sum_{j \in N(v)} \widehat{s}_{ij}$$

2.2 Model 2: Feature Constrained Least Squares

In this model, V is pre-determined by our feature mapping Ψ, (for $1 \leq j \leq n$, the jth row is $\Psi(j)$) therefore U, b, b' are the only set of free parameters. The objective function is the same as in the first model, however now the optimization is done over U only with V fixed, again by gradient descent.

$$L(U, b, b'; V, \mathcal{S}) = \lambda_U ||U||_F^2 + \sum_{i=1}^{S} \left(y_i - U_{u_i *} V_{v_i *}^T - b_{u_i} - b'_{v_i} - \overline{y}\right)^2$$

Again, we can use (1) to estimate the unseen scores of movies in the database. Since this method produces user preferences with respect to an interpretable feature set, for an out-of-sample movie v and a particular user i, we can predict the rating by $\widehat{s}_i(v) = U_{i*}^T \Psi(v)$.

2.3 Model 3: Feature Constrained LASSO

This approach is similar to that of model 2, however we enforce sparsity constraints to the user feature vectors in this model. As described in the introduction, sparsity is both convenient for interpretation and expected to be intrinsic to the underlying data generating mechanism. An intuitive approach to enforce sparsity is to solve a modified version of linear regression

$$\min_{U, b, b'} \sum_{i=1}^{S} \left(y_i - U_{u_i *} V_{v_i *}^T - b_{u_i} - b'_{v_i}\right)^2 + \lambda ||U||_0$$

Where $||U||_0$ is the "ℓ_0 norm" of U which is the same as the number of non-zero entries in U.[2] Despite being a simple idea, this optimization is NP hard because there are exponentially many different supports that need to be considered. However, a powerful result [4] shows that doing ℓ_1 regularization via LASSO instead

[2]Note that this not actually a norm because it fails the absolute homogeneity property $||cv|| = |c| \cdot ||v||$ for any scalar c and vector v.

of ℓ_0 produces a near optimal solution. This is important because the ℓ_1 problem is a convex optimization problem and is tractable. With this in mind, in this model we employ LASSO

$$\min_{U,b,b'} \sum_{i=1}^{S} \left(y_i - U_{u_i *} V_{v_i *}^T - b_{u_i} - b'_{v_i}\right)^2 + \lambda ||U||_1$$

with V fixed. Once U is obtained through LASSO, we can use (1) to estimate the unseen scores of movies in the database. Since this method produces user preferences with respect to an interpretable feature set, for an out-of-sample movie v and a particular user i, we can predict the rating by $\widehat{s}_i(v) = U_{i*}^T \Psi(v)$. Our implementation of LASSO is a modified version of the code in [9].

3 Experiments & Discussion

Throughout all of our experiments, our performance metric is root mean square error (RMSE) with $\lambda_U = \lambda_V = 0.035$, and a learning rate of 0.001 (note that these parameters are irrelevant for the LASSO method). We used $\lambda = 1.0$ in our implementation of LASSO. The learning rate was tuned on the unconstrained SVD model by observing how the train RMSE changed per epoch. We tuned λ_U and λ_V on the unconstrained SVD model by holding out a portion of the training set for validation (roughly 20 percent). Once these parameters were determined for the unconstrained model, we used the same parameters in the constrained model for consistency, and the validation set was put back into the training set to improve the performance of our models. We chose λ for the LASSO model in a similar manner.

3.1 Matrix Completion: Estimating In-Sample Movie Ratings

In this experiment, we partitioned the 20 million known user-movie ratings from the 20M Movielens dataset into two sets, a training set (75 percent) and a hold-out test set (25 percent). For the model 1, we ran our optimization with 60 latent factors. For models 2 and 3 we use our 44 feature space. Stochastic gradient descent was run for 60 epochs. The table below summarizes our results:

	Model 1	Model 2	Model 3	Sample Mean
Train RMSE	0.697	0.762	0.823	1.05
Test RMSE	0.774	0.839	0.942	1.04

The relative performances of the three models is not surprising. Model 1 is the standard SVD for matrix completion, model 2 has V fixed, and model 3 has V fixed as well as a sparsity constraint. Thus, it makes sense that model 1 has the best performance during training and testing since it has the largest parameter space. These results are comparable to [6] which achieved a test RMSE of 0.815 via collaborative filtering on the 1M Movielens data set. While model 2 achieved an RMSE of 0.823, which is very close to the RMSE of [6] using interpretable features, it is important to note that we had more training data, since we worked with the 20M set.

3.2 Beyond Collaborative Filtering: Estimating Out-Of-Sample Movie Ratings

In this experiment, we chose 5 percent of movies in the 20M set at random and held out all ratings for those movies. The non-held out ratings served as the training set, so our algorithms do not get to see any ratings for any of the movies in the hold out set, effectively making these movies out-of-sample. Gradient descent was used for 60 epochs for all three models during training. The table below summarizes the results:

	Model 1	Model 2	Model 3	Sample Mean
Train RMSE	0.695	0.768	0.833	1.044
Test RMSE	0.995	0.960	0.931	1.052

Remarkably, the performance of the three models is the complete opposite from what happened in the first experiment. In terms of training error, the same argument about the number of constraints applies, and it is expected that model 1 outperforms model 2 which outperforms model 3. However, on the test set, the sparsity of user preferences enforced by LASSO for model 3 allows it to generalize to out-of-sample movies to achieve a RMSE similar (0.942 in experiment 1). The other models, however, suffer heavily from overfitting, and model 3 ends up with the best test RMSE. This confirms our belief that sparsity is intrinsic to the structure of our problem.

3.3 Interpretability

We do a case study to demonstrate the interpretability of our feature representation. The first two rows in the below table shows the preferences recovered for four genres for user 2 in the MovieLens data set.

	Action	Adventure	Biography	Crime
Constrained SVD	0.2896	0.3213	-0.0305	-0.035
LASSO & SVD	3.454	0.470	0	0
Avg rating by Genre	4.44	4.75	3.50	3.64

The next row shows the average score he gave to movies with a specific genre. The average rating given out by user 2 is 3.99. The user preferences recovered by both models tells us that user 2 likes action and adventure genres, which is reflected by the fact that user 2 assigns higher ratings to movies of this genre than the typical 3.99. Conversely, both models claim that user 2 does not care much for the biography or crime genres, which is again reflected by the fact that user 2 scores those kinds of movies much lower than his/her typical score of 3.99.

4 Conclusion and Goals

We proposed three methods of extending low-rank matrix factorization approaches to estimating ratings of out-of-sample movies. While the RMSE of predictions based on interpretable features was worse than that of latent factor models, the reduction in accuracy is not too severe and additional insight about user preferences is gained. Finally, we saw that enforcing sparsity in user preferences for our interpretable feature space improves generalization RMSE on out-of-sample movies, confirming our intuition that users typically make decisions based on a small set of personal tastes. Moving forward, there are several things that can be done to gain more insight and better performance. Further refinement of the interpretable feature mapping Ψ in order to make the features more comprehensive in their descriptions would help the learning process. Also, it seems that the 20M MovieLens data set has intrinsically low variance. The sample mean rating estimator which just estimates the average of all known scores achieves an RMSE of 1.05 across the entire dataset, which means that the sample variance of the data set is only about 1.1. We believed that some of our ideas were not able to shine because of the lack of variety in the data set. Aside from constraining the features to be interpretable, another idea that we had is to first learn k latent features using a more conventional algorithm, and then try to find a reasonably well-behaved function[3] f which can map the abstract features into an interpretable feature set. We do not expect linear functions to perform much better than method 2, so we expect such an f to be nonlinear.

[3]Of course, the smoothness and regularity of f is needed to ensure the model has low variance.

References

[1] J. Lee, M. Sun, G. Lebanon. "A Comparative Study of Collaborative Filtering Algorithms", CoRR abs/1205.3193, 2012.

[2] J. Herlocker, J. Konstan, L. Terveen, J. Reidl. "Evaluating Collaborative Filtering Recommender Systems", ACM Transactions on Information Systems, Jan 2004.

[3] Y. Koren, R. Bell, C. Volinsky. "Matrix Factorization Techniques for Recommender Systems", Computer. 42.8:20 27, 2009.

[4] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO", J. Royal Statistical Soc. B, vol. 58, no. 1, pp. 267-288, 1996.

[5] N. D. Lawrence R. Urtasun "Non-linear matrix factorization with Gaussian processes" Proc. Int. Conf. Mach. Learn. pp. 601-608 2009.

[6] B. Marlin, Collaborative Filtering: A Machine Learning Perspective M.S. thesis, Dept. of CS, University of Toronto, Toronto, CA, 2004.

[7] N.N. Liu , E.W. Xiang , M. Zhao , Q. Yang, Unifying explicit and implicit feedback for collaborative filtering, Proceedings of the 19th ACM international conference on Information and knowledge management, October 26-30, 2010, Toronto, ON, Canada [doi 10.1145/1871437.1871643]

[8] Y. Yue, "EE/CS 155 Lecture 13: Latent Factor Models & Non-Negative Matrix Factorization", California Institute of Technology, 2015-2016

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, vol. 3, no. 1, pp. 1122, 2011.

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872