Movie Ratings Prediction with Recommender Systems

Ziqing Fu

Motivation and Task

Recommender systems are systems that help users discover items they may like. Rather than providing a static experience in which users search for and potentially buy products, recommender systems identify recommendations autonomously for individual users based on past purchases and searches, and on other users' behavior. The intensive study of recommender system is important, as nowadays the revenues of companies like Amazon and Netflix are large driven by user engagement and actual values they can deliver to customers. This project seeks to implement a few popular approaches in the field to better understand a variety of algorithms and their relative strengths, weaknesses, and potentials. The focus is on accurately predicting the ratings, which is closely related to the problem of recommending the best possible movies to a user.

Dataset

MovieLens 100K dataset consists of 100,000 ratings (1-5) from 943 users on 1682 movies, with each user having rated at least 20 movies. A data entry is in the format of (UserID, MovieID, Rating, Timestamp). In addition, simple information of users (demographics) and movies (genres) are provided separately. The dataset was split into a training set and a testing set with exactly 10 ratings per user in the testing set.

Approach Overview

The tasks of most of the recommender systems can be viewed as information filtering, and there are two main approaches to information filtering: collaborative filtering and content-based filtering.

Content-based filtering

Content-based filtering recommends items based on a comparison between the content of the items and a user profile. The prediction task can be modeled as a standard classification problem, treating every user u as an independent, separate classification problem. Every item i for which u has provided a rating $r_{u,i}$ is considered as a training instance with the rating as its target value. The dataset comes with the external information that allows the construction of user profiles and movie profiles, and I constructed a single feature vector for each instance, which includes 49 binary-valued attributes of movie genres (19), gender (2), occupations (21), and age groups (7). Random Forest and Multiclass Logistic Regression were selected to perform the regression task aiming at predicting a rating of 1-5. In brief, random forest constructs a multitude of decision trees through examining each predictor variable in model for optimal splits points that minimize loss subject to previous partitions, and logistic regression seeks to estimate the posterior probability of a class assignment.

Collaborative filtering

Alternatively, instead of using features of items and users combined to determine the similarities for predicting ratings, it is possible to rely on past user behavior, for example, previous transactions or product ratings, without requiring the creation of explicit profiles. Collaborative filtering analyzes relationships between users and interdependence among products to identify new user-item associations, and it might be preferable to content-based filtering as it can address data aspects that are

often elusive and difficult to profile while enjoying higher accuracies. The two primary areas of collaborative filtering are the neighborhood methods and latent factor models.

- Neighborhood methods are centered on computing the relationships between items or users. In item-based approach, the prediction is performed by looking for the movie's nearest neighbors - those tend to get similar ratings by the same user. A similar user-based approach can be taken, but item-based approach has better scalability and is easier to explain. The similarity measure between items is based on the Pearson correlation coefficient, and the predicted rating is taken as a weighted average of the ratings of the set of k neighboring items.

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}$$

  where $s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij}$, ρ_{ij} is the Pearson correlation coefficient, n_{ij} is the number of users that rated both i and j , $S^k(i;u)$ is the set of k neighbors, and $b_{ui} = \mu + b_u + b_i$ (μ is average rating over all movies, b_u and b_i are observed deviations of user u and item i, respectively from the average)
- Latent factor models are an alternative that tries to explain the ratings by characterizing both items and users on factors inferred from the ratings patterns, which might measure less well-defined or completely uninterpretable dimensions. Some of the most successful realizations of latent factor models are based on matrix factorization, which characterizes both items and users by vectors of factors, q_i and p_u, inferred from item rating patterns. One strength of matrix factorization is that it allows inference and incorporation of additional information when explicit user preferences (ratings) are not available. Singular value decomposition (SVD) is often applied to map each item and user to factor vectors, and stochastic gradient descent (SGD) is applied to solve the following regularized model

$$\min_{p*,q*,b*} \sum_{(u,i) \in \kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_3 (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

  where set $\kappa = \{(u,i) | r_{ui}\ is\ known\}$

Result and Discussion

Each model is trained on the training set, and accuracies were measured by Root Mean Squared Error (RMSE) on the testing set. I wrote implementations for all the models in Python.

	Method	RMSE
Regression	Random Forest	1.12273
	Multiclass Logistic Regression	1.20739
Neighborhood Model	Simplified Item-based CF	3.60394
	Improved Item-based CF [1]	0.94512
Latent Factor Model	Conventional SVD	2.82581
	Improved SVD [2]	0.96366

[1] Simplified IBCF treats missing values in the user-item matrix as 0; Improved IBCF (a) considers only the common user support, (b) adjusts Pearson correlation with shrunk correlation coefficient, (c) includes user and bias biases

[2] Conventional SVD relies on imputation to fill in missing ratings; Improved SVD model directly models only on the observed ratings and avoid overfitting through regularization

The regression models had higher RMSE compared to the collaborative filtering methods. The regression methods made use of the additional information about users and items available in the dataset, and in the data pre-processing stage I created the feature vector of binary attributes for a given user-movie

pair. Feature-based predictors for rating prediction might suffer from the problem of not having enough features at the granularity required to capture individual tastes and subtle differences between items and users. Moreover, with such large number of parameters, the model could be dominated by only a few parameter values while overlooking others. One possible improvement, though, is modeling $\mathrm{f}(\mathrm{u},\mathrm{m}) = X_{um} * \theta$ directly and learn the set of weights θ directly through minimizing the error function, as it is likely that neither of the existing regression models fit this specific prediction problem perfectly.

Neighborhood methods and latent factor models gave more accurate results. The two simplified models are given to serve as a baseline comparison, and the high RMSE's make sense as they over-simplified modeling through ignoring important assumptions. Theoretically, latent factor models are more expressive and tend to provide more accurate results than neighborhood models, as they can capture and learn the latent factors which encode users' preferences and items' characteristics. On the other hand, neighborhood models are relatively simpler and offer more intuitive explanations of the reasoning behind recommendations [1]. In testing, though, the latent factor model did not necessarily outperform the neighborhood model, and the reason might be possible improvements on tuning hyper-parameters through cross-validation when learning the values of involved parameters through minimizing the regularized error function.

Future Work

Following my work and the discussion in [1], there are further models that I would consider studying and implementing for the next step:

1. An improved neighborhood model that use global weights rather than user-specific weights in the prediction, so that the influence of missing ratings is considered. In addition, another set of weights should be added to include implicit feedback. In this scheme, users who provided many ratings or plenty of implicit feedback would incur greater deviations from baseline estimates.
2. Enhancements to latent factor models including asymmetric-SVD and SVD++. For asymmetric-SVD, instead of providing an explicit parameterization for users, users are represented through the items that they prefer. SVD++ includes implicit feedback that complements the learnt from the given explicit ratings.
3. An integration of the 2 models above in a single framework.
4. Other techniques that have been developed in the field, for example, Restricted Boltzmann Machines that provide another kind of latent factor approach, and ensemble methods that combine different algorithms to provide a single rating.

References

[1] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD), 4, January 2010.*

[2] Y. Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. *Proc. 14th ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining (KDD'08), pp. 426–434, 2008.*