Team 5

Members: Zifeng Zhou, Chao Zhao, Tianze Liang

In our project, we would like to choose **MovieLens** as our dataset. The goal of our project is to build a movie recommender system that will automatically generate a recommended move list, which may contains about 10 movies, for each user according to their own rating scores, the movie classification, the score list of the other users who have some common preference. Here are some algorithms we might use:

1. User-based Collaborative Filtering **(Zifeng Zhou)**

The main idea behind UB-CF is that people with similar characteristics share similar taste. For example, if you are interested in recommending a movie to our friend Bob, suppose Bob and I have seen many movies together and we rated them almost identically. It makes sense to think that in future as well we would continue to like similar movies and use this similarity metric to recommend movies. The method identifies users that are similar to the queried user and estimate the desired rating to be the weighted average of the ratings of these similar users.

1. Model-based Collaborative Filtering
2. Matrix Factorization **(Chao Zhao)**

In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation.

Recommender systems rely on different types of input data, which are often placed in a matrix with one dimension representing users and the other dimension representing items of interest. The most convenient data is high-quality explicit feedback, which includes explicit input by users regarding their interest in products. We refer to explicit user feedback as ratings. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items.

One strength of matrix factorization is that it allows incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion by observing user behavior. Implicit feedback usually denotes the presence or absence of an event, so itis typically represented by a densely filled matrix.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f. Accordingly, each item i is associated with a vector qi ∈ Rf, and each user u is associated with a vector pu ∈ Rf. For a given item i, the elements of qi measure the extent to which the item possesses those factors, positive or negative. For a given user u, the elements of pu measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, qiT pu, captures the interaction between user u and item i based on the user’s overall interest in the item’s characteristics. This approximates user u’s rating of item i, which is denoted by rui , leading to the estimate rui = qiT pu.

To learn the factor vectors (qiT pu), the system minimizes the regularized squared error on the set of known ratings:



where κ is the set of (u,i) pairs for which rui is known (the training set).

The system learns the model by fitting the previously observed ratings. However, the goal is to generalize those previous ratings in a way that predicts future, unknown ratings. Thus, the system should avoid overfitting the observed data by regularizing the learned parameters, whose magnitudes are penalized. The constant λ controls the extent of regularization and is usually determined by cross-validation.

The approach to minimizing the equation above is stochastic gradient descent . Stochastic gradient descent:

The algorithm loops through all ratings in the training set. For each given training case, the system predicts rui and computes the associated prediction error.



Then it modifies the parameters by a magnitude proportional to γ in the opposite direction of the gradient, yielding:



1. Classification Analysis **(Tianze Liang)**

Different people have different preferences. Some people like popular movies and some people may like non-mainstream movies. Some people only watch those well-made movies regardless of their subject matter, but some people just the opposite. It is needed to define a boundary to judge whether a user likes or dislikes a movie. The method extracts the features of the movies in the user viewing history, such as average score, label, subject matter, number of scoring users. Then we train a model based on these features and use this model to recommend other movies to users.

Each of us is in charge of one algorithm, and after that, we will combine these three algorithms to get a more accurate result.