Evaluating Probabilistic Matrix Factorization on Netflix Dataset

Anonymous Author(s)
Affiliation
Address
email

Abstract

Collaborative Filtering attempts to make automatic taste recommendations by examing a large number of taste information. Methods for achieving Collaborative Filtering can be broadly categorized into model based, and memory based techniques. In this project, we review and implement three variants of Probabilistic Matrix Factorization, a model based Collaborative Filtering algorithm. We compare the performance of Probabilistic Matrix Factorization to a memory based Collaborative Filtering algorithm, the K nearest neighbor algorithm. The model performance is compared using three datasets derived from the full NetFlix movie dataset, each with varying data sparsity. Specifically, the root mean square error measure (RMSE) is used as the metric to compare the relative performance between the different CF algorithms. In our performance evaluation, we discovered when the data sparsity is sufficiently low, Probabilistic Matrix Factorization out performs K nearest neighbor, achieving RMSE of as low as 0.83. However, when the data sparsity is high, KNN yields marginally better performance, obtaining RMSE score of 1.0453 on the most sparse dataset, while the regular Probabilistic Matrix Factorization scored a 1.0771 on the same dataset

1 Introduction

The process of automatically inferring the liking of a customer and users is of vast interest to businesses and other fields. Extra sales revenue can be gained by automatically selecting and recommending merchandise that suits the user's taste. Customer's user experience can be improved by presenting contents tailored toward the customer's personal taste. Collaborative Filtering has been proposed to make automatic user recommendations, and it has been studied widely by the machine learning community. Collaborative Filtering algorithms can be classified into two broad categories. Model based algorithms attempt to uncover latent variables to explain the observed data. The latent factors can then be used to compute the prediction for the test dataset. On the other hand, memory based algorithms makes recommendation by first defining a similarity measure between data points. The neighboring points most similar to the query point is then used to compute the test dataset recommendations.

The Netflix dataset contains 100,480,507 ratings from 480,189 random-chosen, anonymous users on 17,770 movies. It also provides a validation data containing 1,408,395 ratings. The user ratings from the Netflix dataset can be consolidated into a $U^T \times V$ user to movie rating matrix R. Collaborative Filtering via Probabilistic Matrix Factorization algorithm uncovers a number of latent factors from the training matrix R, and builds a probabilistic model to reconstruct the whole matrix though the observed elements. Probabilistic Matrix Factorization assumes that the element R_{ij} is generated from a normal distribution with mean $U_i^T V_j$, where U_i, V_j are latent feature vector for user i and movie j, respectively. The interpretation of latent feature vector is that each dimension of the latent

feature vector represents a certain aspect of a movie. Therefore, the value $(U_i)_k \cdot (V_j)_k$, which corresponds to the k-th feature, contributes to part of the rating from user i on movie j.

We also apply K nearest neighbor algorithm to the Netflix dataset. This mechanism of Collaborative Filtering uses training user rating data to compute similarity between users in the test dataset. The top ranked users most similar to the test user are then used to compute the movie rating for the query. In this project, we operate directly on the movie rating vectors from R, normalized onto the unit hypersphere with with the Euclidean 2 norm as the similarity measure.

2 Probabilistic Matrix Factorization

2.1 Previous Work

Salakhutdinov and Minh proposed Probabilistic Matrix Factorization (PMF) and constrained Probabilistic Matrix Factorization and applied them on Netflix dataset in [1]. Koren introduced the idea of temporal dynamics in [2], which takes into consideration factors such as time, user movie preference. Our model of Probabilistic Matrix Factorization with bias is designed to the temporal dynamics with adding bias terms in PMF models.

2.2 Basic PMF

Given data matrix R where entries R_{ij} represents the rating of movie j from user i, Probabilistic Matrix Factorization probabilistically decomposes R as the product of two matrices of lower rank $R = U^T \times V$. The column vector U_i, V_j represents the latent feature vector of user i and movie j, respectively. We assume that the Movie rating R_{ij} can be modeled by the following normal distribution:

$$P(R_{ij}|U, V, \sigma^2) = \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2) \tag{1}$$

Let $\mathcal{O}$ be the set of observed ratings, i.e. $(i, j) \in \mathcal{O}$ if and only if user i rated movie j.

$$P(R|U, V, \sigma^2) = \prod_{(i,j)\in\mathcal{O}} \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2) \tag{2}$$

We place a zero-mean spherical Gaussian priors on user and movie feature vectors:

$$P(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}), P(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}). \tag{3}$$

The negative log of the posterior distribution over the user and movie features is given by

$$-\ln p(U, V|R, \sigma^2, \sigma_U^2, \sigma_V^2) = \frac{1}{2\sigma^2} \sum_{(i,j)\mathcal{O}} (R_{ij} - U_i^T V_j)^2 + \frac{1}{2\sigma_U^2} \sum_{i=1}^{N} U_i^T U_i + \frac{1}{2\sigma_V^2} \sum_{j=1}^{N} V_j^T V_j + C_{\sigma,\sigma_U,\sigma_V} \tag{4}$$

where $C_{\sigma,\sigma_U,\sigma_V}$ is a constant when we kept $\sigma, \sigma_U, \sigma_V$ fixed. Therefore maximizing the posterior probability is equivalent to minimizing (4), which is equivalent to minimizing

$$E = \frac{1}{2} \sum_{(i,j)\in\mathcal{O}} (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2} \sum_{i=1}^{N} ||U_i||^2 + \frac{\lambda_V}{2} \sum_{j=1}^{M} ||V_j||^2 \tag{5}$$

where $\lambda_U = \sigma^2/\sigma_U^2, \lambda_U = \sigma^2/\sigma_U^2$. Since the movie rating is range from 1 to 5, we change the data by the function $f : x \to (x - 1)/4$ that maps $[1, 5]$ to $[0, 1]$, and adjust our PMF model to be the following:

$$P(R|U, V, \sigma^2) = \prod_{(i,j)\in\mathcal{O}} \mathcal{N}(R_{ij}|g(U_i^T V_j), \sigma^2) \tag{6}$$

here $g(\cdot)$ is the logistic function given by $g(x) = 1/(1 + \exp(-x))$, which maps $\mathbf{R}$ to $(0, 1]$. The corresponding error function becomes:

$$E = \frac{1}{2} \sum_{(i,j)\in\mathcal{O}} (R_{ij} - g(U_i^T V_j))^2 + \frac{\lambda_U}{2} \sum_{i=1}^{N} ||U_i||^2 + \frac{\lambda_V}{2} \sum_{j=1}^{M} ||V_j||^2 \tag{7}$$

We minimize (7) by performing gradient descent method.

2.3 PMF with Bias

There is evidence from [2] that some fact about the rating, such as the release date of the movie, the date of the rating, etc., relays only on the movie or the user, but not the interaction of movie and user. It is sensible to introduce a bias term for each movie and user, i.e., modify (6) slightly by

$$P(R|U,V,Bu,Bv,\sigma^2) = \prod_{(i,j)\in\mathcal{O}} \mathcal{N}(R_{ij}|g(U_i^T V_j + Bu_i + Bv_j),\sigma^2) \tag{8}$$

here the bias term Bu, Bv are also latent variables. We put a zero-mean Gaussian prior on the bias term as well:

$$P(Bu|\sigma_U^2) = \mathcal{N}(0,\sigma_U^2\mathbf{I}), P(Bv|\sigma_V^2) = \mathcal{N}(0,\sigma_V^2\mathbf{I}). \tag{9}$$

We do maximum posterior probability estimation for U, V, Bu, Bv, which is equivalent to minimizing the following

$$E = \frac{1}{2}\sum_{(i,j)\in\mathcal{O}} (R_{ij} - g(U_i^T V_j + Bu_i + Bv_j))^2 + \frac{\lambda_U}{2}(\sum_{i=1}^{N} ||U_i||^2 + ||Bu||^2) + \frac{\lambda_V}{2}(\sum_{j=1}^{M} ||V_j||^2 + ||Bv||^2) \tag{10}$$

2.4 Constrained PMF

Constrained PMF is introduced in [1], in order to place a better model in the case that many users had rated a very few movies (i.e. less than 10 movies). This model is build on the assumption that the user feature U_i is determined by latent variable Y_i as well as latent variables W_k for which movie k the user has rated. As a result, users that have seen similar movies will have similar prior distributions for their feature vectors. More specifically, The feature vector for user i is given by

$$U_i = Y_i + \frac{\sum_{k\in\mathcal{O}_i} W_k}{|\mathcal{O}_i|} \tag{11}$$

where $\mathcal{O}_i = \{k|(i,k)\in\mathcal{O}\}$, i.e. the set of movies that user i has rated. The conditional distribution over the observed rating is defined as

$$P(R|Y,V,W,\sigma^2) = \prod_{(i,j)\in\mathcal{O}} \mathcal{N}(R_{ij}|g[Y_i + \frac{\sum_{k\in\mathcal{O}_i} W_k}{|\mathcal{O}_i|}],\sigma^2) \tag{12}$$

and W is also regularized by placing a zero-mean spherical Gaussian prior on it

$$P(W|\sigma'^2) = \prod_{i=1}^{N} \mathcal{N}(W_i|0,\sigma_W^2\mathbf{I}) \tag{13}$$

Again, we estimate a maximum posterior probability estimation by minimizing

$$E = \frac{1}{2}\sum_{(i,j)\in\mathcal{O}} (R_{ij} - g[Y_i + \frac{\sum_{k\in\mathcal{O}_i} W_k}{|\mathcal{O}_i|}]^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} ||U_i||^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} ||V_j||^2 + \frac{\lambda_W}{2}\sum_{k=1}^{M} ||W_k||^2 \tag{14}$$

2.5 Collaborative Filtering via K nearest neighboring

This approach to CF entails finding the K closest neighbors in the test dataset and compute the recommendation by averaging the recommendations from the closest neighbors. In the context of this project, data vectors of user movie ratings are first normalized onto the unit hypersphere. For each user in the validation set, the weighted average ratings for the movie from the K closest users in the training set is reported as the movie rating for the query. Specifically, we use Euclidean 2 norm $||x||_2 = \sqrt{x\cdot x}$ as the distance measure. After the top K neighbors have been identified, the movie ratings of the K neighbors are passed through Inverse Distance Weighing to compute the movie rating for the test user.

$$R_{ij} = \sum_{l=1}^{K} \frac{w(i,n_l^i)}{\sum_{c=1}^{K} w(i,n_c^i)} R_{n_l^i,j} \tag{15}$$

Here n_l^i denotes the $l-$th nearest neighbor who has rated movie j, and $w(i,s) = 1/distance(i,s)$. In order to reduce the effect of users in the further neighborhood have on the movie rating, a slight modification to the distance metric is introduced by setting $w(i,s) = 1/\exp(distance(i,s) \cdot b)$. For our experiment on the Netflix data, the parameters are set to K = 1000 and b = 2.

Fast KNN computation can be realized by utilizing binary space partitioning data structures such as KD-tree [4]. In our experiment, KD-tree can be constructed quickly with millions of training instances, each in a high dimensional (>8000) movie rating feature space. We note that empirically finding the K nearest neighbor using KD-tree can be done quickly.

3 Experiments

3.1 Description of data sets

Salakhutdinov and Mnih provided us with Dataset 1, a subset of the full Netflix data generated by randomly selecting 50,000 users and 1850 movies. Dataset 1 contains 1,082,982 training and 2,462 validation user/movie pairs. Over 50% of the users in the training dataset have less than 10 ratings. Dataset 1 is a very sparse dataset and it can be used to demonstrate the advantage of constrained PMF.

Dataset 2 is created by randomly selecting 30,000 users and 4000 movies. It contains 3,491,319 training and 35,202 validation user/movie pairs. Dataset 3 is created by randomly selecting 20,000 users and 8000 movies. It contains 4,806,240 training and 48,466 validation user/movie pairs. Dataset3 is the most dense dataset over all three.

3.2 Details of Training

In our experiments, we used 10 dimensional latent features vectors (D = 10) for PMF on all datasets. We subdivided datasets into mini-batches of size 10,000 (user / movie / rating triples), and updated the feature vectors after each mini-batch. We trained For training on dataset 1, we set 0.005 as the learning rate with momentum of 0.9. For dataset 2 and 3, we used 0.002 as our learning rate and momentum of 0.9. The regularization parameters for PMF are set to $\lambda_U = \lambda_V = \lambda_W = 0.002$. For the K nearest neighbor algorithm, the parameters are set to $K = 1000$ with the exponential decay constant $b = 2$.

3.3 Experimental Results

We evaluate the model performance by their root mean squared error (RMSE) on each validation set. The basic PMF model works well on dataset 2 and 3. Training the basic PMF converged very quickly and the basic PMF achieved 0.8550 and 0.8365 for dataset 2 and 3, respectively. However, the basic PMF did not produce good prediction on dataset 1. It achieved a RMSE of 1.0772 before overfitting, which is slightly larger than the movie average (1.0640). Due to the sparsity of dataset 1, the number of observed ratings for most users is low, causing performance depredation on the basic PMF. By adding a bias term to the basic PMF, the RMSE decreased to 1.0176. which is better than constrained PMF (1.0472 in our experiment and 1.0510 in [1]).

For dataset 2 and 3, PMF with bias achieved RMSE of 0.8595 and 0.8401, respectively; constrained PMF achieved RMSE of 0.8499 and 0.8397. They are equally good as basic PMF. KNN achieved slightly higher RMSE (0.9917 on dataset 2 and 0.9802 on dataset 3).

Based on the dataset that we have generated, we observe that given an adequate number of ratings for each user, the basic PMF model captures the interaction between user and movie quite well. It is mathematically simpler to express and have fewer parameters to estimate compare to PMF with bias and constrained PMF. However, it may have trouble in the case when observed ratings are limited.

The constrained PMF also works well on all 3 datasets. However, the complexity of the model is relatively higher, which results in a much longer training time during our experiment. On the other hand, PMF with bias has good performance on all dataset we tried and it also enjoys the simple mathematical form, with fast training convergence rate.

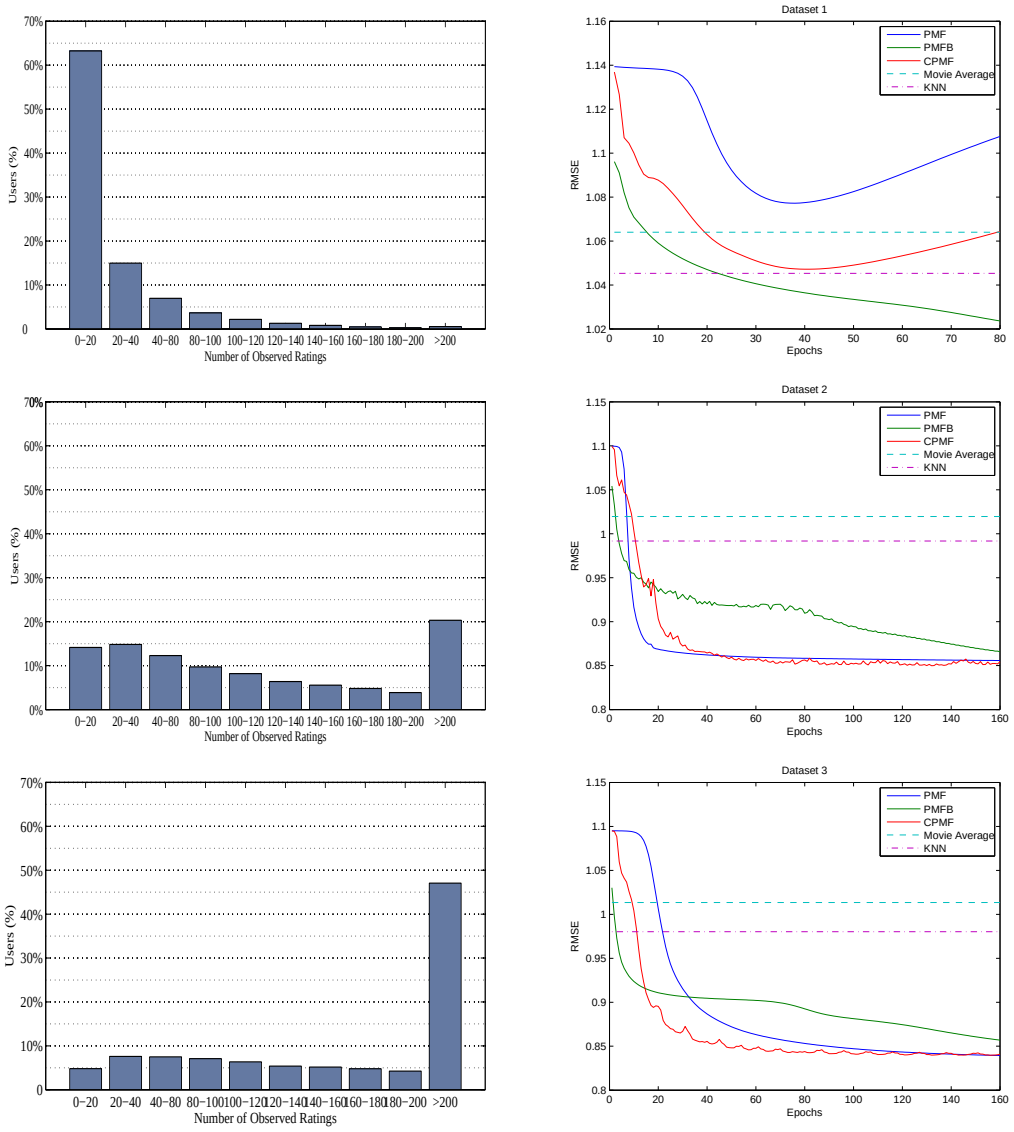

Figure 1: Left panel (from top to bottom): Distribution of users in training dataset 1,2,3. The x-axis displays the number of observed ratings and the y-axis displays percentage of total users. Right panel (from top to bottom): Performance of Probabilistic Matrix Factorization (PMF), PMF with bias (PMFB), constrained PMF (CPMF), Movie Average (a model that always predicts the average rating for each movie), and K-nearest neighbor (KNN) on dataset 1,2,3. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs through the entire dataset.

KNN has an adequate performance on each dataset. It is not as accurate, in terms of RMSE, as PMF based models on dataset 2 and 3. However, it obtained a lower RMSE of 1.0453 on dataset 1 compare to PMF and constrained PMF.

4 Summary and Discussion

In this project we have reviewed and implemented three variants of Probabilistic Matrix Factorization, a model based approach to Collaborative Filtering. We compared the performance of all three variants of PMF to the K nearest neighbor algorithm on three Netflix datasets of varying sparsity. In the experiment, we showed the relative robustness of PMF in predicting user movie ratings using the Netflix dataset, disregarding the sparsity of the data.

The fact that our K nearest neighbor algorithm operates directly on the movie rating feature space may be the culprit that hinders its performance. The K nearest neighbor method can be enhanced easily by applying standard machine learning techniques, such as dimensionality reduction on the feature space, or by incorporating semantics data from the movies during the computation of the similarity between users. While incorporating semantics data maybe very costly to the computation of K nearest neighbor, the addition of linguistic knowledge may increase the performance of K nearest neighbor significantly.

Acknowledgement

We thank Richard Zemel, Andriy Aminh and Ryan Adams for many helpful discussions and suggestions.

References

[1] Ruslan Salakhutdinov. & Andriy Mnih. Probabilistic Matrix Factorization. , *Advances in Neural Information Processing Systems (NIPS 2007)*

[2] Yehuda Koren. Collaborative Filtering with Temporal Dynamics . *The 15th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2009*

[3] Wei Chu. & Zoubin Ghahramani. Probabilistic Models for Incomplete Multi-dimentional Arrays, *In: AISTATS Twelfth International Conference on Artificial Intelligence and Statistics 2009, 16-18 April 2009, Florida, USA*.

[4] Andrew Moore. An Intoductory Tutorial On KD-tree *Technical Report No. 209, Computer Laboratory, University of Cambridge. 1991*