A Combined Collaborative Filtering Model for Recommender System

Team *1UP+*: Di Fu[†] and Zongxiao He[‡]

***Abstract*—Collaborative filtering is a widely used and successful recommendation paradigm that models users' collaborative behaviors reflected in previous transactions for recommendation. Neighborhood model and latent factor model are most common approaches to collaborative filtering. In this project, we build several rating prediction models based upon neighborhood model and latent factor model and applied them on the MovieLens dataset. We evaluated the performance each model and ensembled a final model to predict a numerical rating for each missed entry in MovieLens dataset.**

I. Dataset

The MovieLens academic dataset provides ratings of the 6,040 users to 3,187 movies (Figure 1). The Whole Training Set (WTS) available to students includes 801,051 (4.15%) ratings, user meta-data (gender, age, and occupation), and movie meta-data (name and genres). Ratings are integers between 1 to 5. The Qualifying Set on Kaggle is split into two parts: Quiz set, which is used for evaluate the submission in public leaderboard, and Test Set, which will be used for calculating final score. For the purpose of training and comparing models, the hand-out WTS is split randomly into Training Set (TS, $\sim 97\%$ of WTS) and Probe Set ($\sim 3\%$ of WTS).

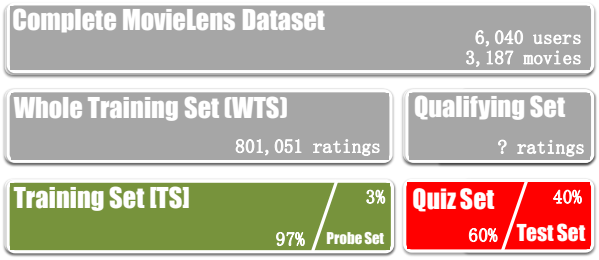

Fig. 1. The structure of MovieLens dataset.

We model users and movies as nodes, with ratings as directed edges. The distributions of some statistics about this dataset are shown in Figure 2. The ratings tend to more positive than negative (right panel). A important fact about this dataset is that the variance of average rating per movie (left panel) is significantly larger than that of average rating per movie (middle panel), which suggests more information is hidden under item-item pairs.

[†]Department of Statistics, Rice University, df14@rice.edu.
[‡]Department of Genetics, Baylor College of Medicine, zh6@rice.edu.

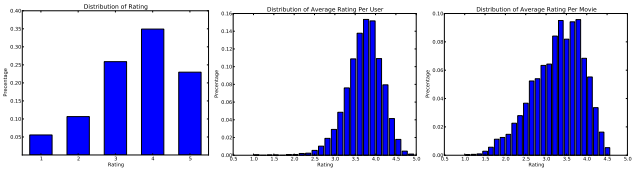

Fig. 2. *Left To Right*: Distribution of Rating, Distribution of User Average Rating, Distribution of Movie Average Rating.

II. Methodology

A. Notations

u and v will be reserved as indexing letters for users, as for items(movies) i and j. r_{ui} indicates the rating by user u for item i, and $\hat{r}_{ui}$ is the predicated value of it. The set of (u, i) pairs in which r_{ui} is known is $K = \{(u, i)|r_{u,i}$ is know$\}$. Regularization parameters are denoted as $\lambda_1, \lambda_2, \ldots$ and step sizes are denoted as $\gamma_1, \gamma_2, \ldots$. The root mean square error on Probe Set is denoted as Probe_RMSE and will be used to measure the convergence of our iterative optimization algorithms. The average of root mean square error in cross validation is denoted as mean_RMSE and will be used for model selection.

B. Parameter selection

Each models we developed has dataset dependent parameters (e.g. regularization, step size, number of features etc). In order to select the correct ones, we use an approach including three-round grid search and an 10-fold cross validation. Starting with a wide range ($[0.0005, 0.05]$ for λ's and γ's, $[50, 400]$ for number of features), each round of grid search narrows down the parameter spaces. The best 30 parameter sets (defined as smallest Probe_RMSEs) of last round grid search will be evaluated using 10-fold cross validation, and the optimal parameter sets are chosen by smallest mean_RMSEs.

C. Baseline Model

To adjust user-specific and item-specific effects, the baseline estimate b_{ui} of rating r_{ui} is given as

$$b_{ui} = \bar{r} + b_u + b_i \tag{1}$$

where $\bar{r}$ is the global average rating, b_u and b_i are user bias and movie bias [1].

The b_u and b_i can be calculated by solving the least squares problem:

$$\min_{b_*} \sum_{(u,i)\in K} (r_{ui} - \bar{r} - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2) \tag{2}$$

Instead of directly solving the equation (2), the b_u and b_i can be estimated via gradient descent method, as shown in algorithm 1.

Algorithm 1: Baseline model algorithm

Input: Training data K; γ_1; λ_1.
Output: b_*; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $(u,i) \in K$ **do**
    $e_{ui} := r_{ui} - \hat{r}_{ui}$
    $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
  end
  $\gamma_1 \leftarrow \gamma_1 \times 0.9$
  calculate $Probe\_RMSE$
end

D. Global neighborhood (GN) model

1) Neighborhood model: Neighborhood model is very effective at detecting very localized relationships and rely on a few significant neighborhood relations. The item-oriented neighborhood [2] is given as

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + \frac{\sum_{j \in S^k(i;u)} (r_{uj} - b_{uj}) \cdot s_{ij}}{\sum_{j \in S^z(i;u)} s_{ij}} \quad (3)$$

where $S^z(i;u)$ denotes the "neighbors", which is defined as z items rated by u and most similar to i. The s_{ij} is the Pearson correlation coefficient that measures the tendency of users to rate movie i and movie j similarly.

Model defined in (2) is fully rely on the neighbors, since the interpolation weights sum to one. This would be problematic in cases where neighborhood information is absent. An improved model [1] is suggested as

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} \quad (4)$$

where $R(u)$ is the set of items rated by user u. The user-specific interpolation weight $s_{ij} / \sum_{j \in S^z(i;u)} s_{ij}$ is replaced by w_{ij}, which denote the global weight from j to i and is learned from the data through optimization.

2) Implicit feedback: An important lesson from the Netflix Prize competition is the importance of integrating different forms of implicit feedbacks, such as purchase history, browsing history, search patterns[3]. The MovieLens dataset does not only tell us the rating values, but also which movies users rate, regardless of how they rated these movies. It has been shown that incorporating this kind of implicit data significantly improved prediction accuracy [1].

Following up model (4), implicit feedback can be added as another set of weights,

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} + \sum_{j \in N(u)} c_{ij} \quad (5)$$

where $N(u)$ contains all items for which u provided an implicit preference. c_{ij} is offset added to baseline estimates, which is expected to be high if j is predictive on i.

To avoid the great deviations from baseline estimates for users that provided many ratings or plenty of implicit feedback, add an regularization item to the model (5) as

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} c_{ij} \quad (6)$$

The model (6) can be learned by solving the regularized least squares problem

$$\min_{b_*, w_*, c_*} \sum_{(u,i) \in K} \Big(r_{ui} - \bar{r} - b_u - b_i - |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} - |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} c_{ij} \Big)^2 + \lambda_3 (b_u^2 + b_i^2 + \sum w_{ij}^2 + \sum c_{ij}^2) \quad (7)$$

A gradient descent solver for this optimization problem (7) is shown in algorithm 2.

Algorithm 2: GN model algorithm

Input: Training data K;γ_1, γ_3; λ_1, λ_3.
Output: b_*, w_*, c_*; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $(u,i) \in K$ **do**
    $e_{ui} := r_{ui} - \hat{r}_{ui}$
    $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
    for $j \in R(u)$ **do**
      $w_{ij} \leftarrow$
      $w_{ij} + \gamma_3 \cdot (e_{ui} \cdot |R(u)|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) - \lambda_3 \cdot w_{ij})$
    end
    for $j \in N(u)$ **do**
      $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot (e_{ui} \cdot |N(u)|^{-\frac{1}{2}} - \lambda_3 \cdot c_{ij})$
    end
  end
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \times 0.9$
  calculate $Probe\_RMSE$
end

E. Singular value decomposition (SVD) model

Latent factor model is another most common approach to collaborative filtering. An drawback of neighborhood model is that it ignores the vast majority of ratings by a user and therefore it's unable to capture the totality of weak signals encompassed in all of a user's ratings. Latent factor models are generally effective at estimating overall structure that relates simultaneously to most or all items. We will focus on models that are induced by Singular Value Decomposition (SVD) on the user-item ratings matrix.

In this model, each user u is associated with a user-factors vector $p_u \in \mathbb{R}^k$, and each item i with an item-factors vector $q_i \in \mathbb{R}^k$. The prediction rule is

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + p_u^T q_i \tag{8}$$

Simon Funk[4] introduced a gradient descent optimization approach (shown in algorithm 3) to learn parameters that minimized the associated squared error

$$\min_{b_*,q_*,p_*} \sum_{(u,i)\in K} \Big(r_{ui} - \bar{r} - b_u - b_i - p_u^T q_i\Big)^2 + \lambda_2(b_u^2 + b_i^2 + ||p_u||^2 + ||q_i||^2) \tag{9}$$

Algorithm 3: SVD model algorithm

Input: Training data K; k; γ_1, γ_2; λ_1, λ_2.
Output: b_*, q_*, p_*; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $(u,i) \in K$ **do**
    $e_{ui} := r_{ui} - \hat{r}_{ui}$
    $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
    $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot p_u - \lambda_2 \cdot q_i)$
    $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_2 \cdot p_u)$
  end
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \times 0.9$
  calculate $Probe\_RMSE$
end

F. Asymmetric-SVD (SVDasym) model

In the plain SVD model, a user is represented by the feature vector q_u. An extended SVD model integrating neighborhood relationship and implicit feedback is suggested by Paterek [5] and Koren [1] is given as

$$\begin{aligned}\hat{r}_{ui} =& \bar{r} + b_u + b_i + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} (r_{uj} - b_{uj}) q_i^T x_i \\ &+ |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} q_i^T y_j \\ =& \bar{r} + b_u + b_i + q_i^T \Big(|R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} (r_{uj} - b_{uj}) x_j \\ &+ |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\Big)\end{aligned} \tag{10}$$

where $y_j \in \mathbb{R}^k$ is factor vector associated with item j representing implicit feedback. A gradient descent appraoch given in algorithm 4 can be employed to solve the system.

G. SVD++ model

Koren [1] proposed a more direct modification of model (8) to integration of implicit feedback, and the model is given as

$$\hat{r}_{ui} = \bar{r} + b_u + b_i + q_i^T \Big(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\Big) \tag{11}$$

Algorithm 4: SVDasym model algorithm

Input: Training data K; k; γ_1, γ_2; λ_1, λ_2.
Output: b_*, q_*, x_*, y_*; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $u = 1, \ldots, m$ **do**
    $p_u \leftarrow |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} (r_{uj} - b_{uj}) * x_j + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j$
    $sum \leftarrow 0$
    for $i \in R(u)$ **do**
      $e_{ui} := r_{ui} - \hat{r}_{ui}$
      $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
      $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
      $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot p_u - \lambda_2 \cdot q_i)$
    end
    for $j \in R(u)$ **do**
      $x_j \leftarrow x_j + \gamma_2 \cdot \big(|N(u)|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) \cdot sum - \lambda_2 \cdot x_j\big)$
    end
    for $j \in N(u)$ **do**
      $y_j \leftarrow y_j + \gamma_2 \cdot (|N(u)|^{-\frac{1}{2}} \cdot sum - \lambda_2 \cdot y_i)$
    end
  end
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \times 0.9$
  calculate $Probe\_RMSE$
end

Model can be trained by minimizing the associated squared error function through gradient descent, as shown in algorithm 5.

Algorithm 5: SVD++ model algorithm

Input: Training data K; k; γ_1, γ_2; λ_1, λ_2.
Output: b_*, q_*, p_*, y_*; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $(u,i) \in K$ **do**
    $e_{ui} := r_{ui} - \hat{r}_{ui}$
    $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
    $q_i \leftarrow q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j) - \lambda_2 \cdot q_i)$
    $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_2 \cdot p_u)$
    for $j \in N(u)$ **do**
      $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |N(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_2 \cdot y_i)$
    end
  end
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \times 0.9$
  calculate $Probe\_RMSE$
end

H. An integrated model (SVDGN)

An common problem of SVD based models is that these model are poor at detecting strong associations among a small set of closely related items, precisely where neighborhood

models do best. Koren[1] proposed a combined model that improves prediction accuracy by capitalizing on the advantages of both neighborhood and latent factor approaches. The integrated model is proposed to sum the model (6) and model (11).

$$\begin{aligned}\hat{r}_{ui} =& \bar{r} + b_u + b_i + q_i^T \Big(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \Big) \\ &+ |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} \\ &+ |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} c_{ij} \end{aligned} \tag{12}$$

Model parameters can be determined by minimizing the associated regularized squared error function through gradient descent, as shown in algorithm 6.

Algorithm 6: SVDGN model algorithm

Input: Training data K; k; $\gamma_1, \gamma_2, \gamma_3$; $\lambda_1, \lambda_2, \lambda_3$.
Output: $b_*, q_*, p_*, y_*, w_*, c_*$; $Probe\_RMSE$.
while *Probe_RMSE keep decreasing* **do**
  for $(u, i) \in K$ **do**
    $e_{ui} := r_{ui} - \hat{r}_{ui}$
    $b_u \leftarrow b_u + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_1 \cdot (e_{ui} - \lambda_1 \cdot b_i)$
    $q_i \leftarrow$
    $q_i + \gamma_2 \cdot (e_{ui} \cdot (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) - \lambda_2 \cdot q_i)$
    $p_u \leftarrow p_u + \gamma_2 \cdot (e_{ui} \cdot q_i - \lambda_2 \cdot p_u)$
    for $j \in N(u)$ **do**
      $y_j \leftarrow y_j + \gamma_2 \cdot (e_{ui} \cdot |N(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_2 \cdot y_i)$
    end
    for $j \in R(u)$ **do**
      $w_{ij} \leftarrow$
      $w_{ij} + \gamma_3 \cdot (e_{ui} \cdot |R^k(i;u)|^{-\frac{1}{2}} \cdot (r_{uj} - b_{uj}) - \lambda_3 \cdot w_{ij})$
    end
    for $j \in N(u)$ **do**
      $c_{ij} \leftarrow c_{ij} + \gamma_3 \cdot (e_{ui} \cdot |N^k(i;u)|^{-\frac{1}{2}} - \lambda_3 \cdot c_{ij})$
    end
  end
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \times 0.9$
  calculate $Probe\_RMSE$
end

I. Blending

The combination of different kinds of algorithms have been shown to be able to significantly improve performance over individual algorithms in Netflix Prize Challenge [6]. Assume to combine the predictions of K models, and set the original ratings on Probe Set as $\mathbf{b} \in \mathbb{R}^P$ (P is the number of ratings in Probe Set). The following framework is used to combine the predications:

- Train
  1) Use Training Set (Probe Set excluded) to train individual models, the predictions on Probe Set from kth model is $\mathbf{x}_k \in \mathbb{R}^P$. Set $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K)$;
  2) Use Whole Training Set (Probe Set included) to train the same models (i.e the same regularization parameters, step sizes, iteration number and number of features), and predict all missing entries. Set the predictions as $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K)$, where $y_k \in \mathbb{R}^Q$ is the prediction from kth model and Q is the number of missing entries;
- Combine
  1) Use predictions $\mathbf{X}$ and original ratings $\mathbf{b}$ to train blending model $f(\cdot)$;
  2) The final prediction of all missing entries is given as $f(\mathbf{Y})$.

In our project, we made few modification the open software *ensemble learning framework (ELF)* [7] to perform prediction blending. Two approaches, Ridge Regression and Neural Network, are used to combine the predictions.

1) Ridge regression: Assuming a quadratic error function and a ridge regularization λ, the linear combination weights $\mathbf{w}$ (vector of length K) can be given as

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{b}$$

10-fold cross validation is used to select a proper ridge regularization constant λ.

2) Neural network: The training of neural networks is performed by stochastic gradient descent. The output neuron n has a sigmoid activation function with an output swing of -1 to 1. To generate rating predictions in the range of $[1, 5]$, the transformation $r = 3.6 * n + 3.0$ is used (works well for Netflix data).

III. Result

The progress of our project is shown in Figure 3.

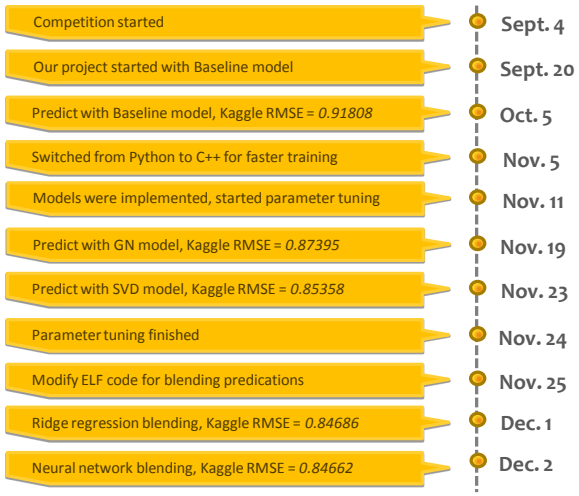

Fig. 3. Team *1UP+* project progress.

The Baseline model is evaluated on a wide parameter space of γ_1 and λ_1, and the optimal Baseline model is obtained when

model	mean_RMSE	description
GN	0.860872	$\gamma_1 = 0.02, \gamma_3 = 0.05, \lambda_1 = 0.01, \lambda_3 = 0.005$
GN	0.860883	$\gamma_1 = 0.02, \gamma_3 = 0.05, \lambda_1 = 0.005, \lambda_3 = 0.005$
GN	0.860922	$\gamma_1 = 0.01, \gamma_3 = 0.05, \lambda_1 = 0.01, \lambda_3 = 0.005$
GN	0.860934	$\gamma_1 = 0.02, \gamma_3 = 0.05, \lambda_1 = 0.002, \lambda_3 = 0.005$
GN	0.860936	$\gamma_1 = 0.01, \gamma_3 = 0.05, \lambda_1 = 0.005, \lambda_3 = 0.005$
SVD	0.842237	$k = 400, \gamma_1 = 0.02, \gamma_2 = 0.04, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD	0.842259	$k = 400, \gamma_1 = 0.02, \gamma_2 = 0.04, \lambda_1 = 0.01, \lambda_2 = 0.03$
SVD	0.842287	$k = 400, \gamma_1 = 0.02, \gamma_2 = 0.04, \lambda_1 = 0.005, \lambda_2 = 0.03$
SVD	0.842289	$k = 300, \gamma_1 = 0.02, \gamma_2 = 0.04, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD	0.842310	$k = 400, \gamma_1 = 0.02, \gamma_2 = 0.04, \lambda_1 = 0.002, \lambda_2 = 0.03$
SVDasym	0.843231	$k = 300, \gamma_1 = 0.015, \gamma_2 = 0.01, \lambda_1 = 0.005, \lambda_2 = 0.05$
SVDasym	0.843250	$k = 300, \gamma_1 = 0.01, \gamma_2 = 0.01, \lambda_1 = 0.01, \lambda_2 = 0.05$
SVDasym	0.843265	$k = 300, \gamma_1 = 0.012, \gamma_2 = 0.015, \lambda_1 = 0.005, \lambda_2 = 0.06$
SVDasym	0.843266	$k = 300, \gamma_1 = 0.02, \gamma_2 = 0.01, \lambda_1 = 0.001, \lambda_2 = 0.05$
SVDasym	0.843269	$k = 300, \gamma_1 = 0.01, \gamma_2 = 0.01, \lambda_1 = 0.008, \lambda_2 = 0.05$
SVD++	0.844711	$k = 200, \gamma_1 = 0.04, \gamma_2 = 0.022, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD++	0.844712	$k = 200, \gamma_1 = 0.035, \gamma_2 = 0.022, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD++	0.844725	$k = 200, \gamma_1 = 0.03, \gamma_2 = 0.022, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD++	0.844745	$k = 150, \gamma_1 = 0.035, \gamma_2 = 0.022, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVD++	0.844756	$k = 150, \gamma_1 = 0.03, \gamma_2 = 0.022, \lambda_1 = 0.02, \lambda_2 = 0.03$
SVDGN	0.840031	$k = 400, \gamma_1 = 0.05, \gamma_2 = 0.035, \gamma_3 = 0.012, \lambda_1 = 0.01, \lambda_2 = 0.035, \lambda_3 = 0.15$
SVDGN	0.840056	$k = 400, \gamma_1 = 0.05, \gamma_2 = 0.035, \gamma_3 = 0.012, \lambda_1 = 0.008, \lambda_2 = 0.035, \lambda_3 = 0.15$
SVDGN	0.840101	$k = 400, \gamma_1 = 0.05, \gamma_2 = 0.035, \gamma_3 = 0.012, \lambda_1 = 0.005, \lambda_2 = 0.035, \lambda_3 = 0.15$
SVDGN	0.840106	$k = 400, \gamma_1 = 0.04, \gamma_2 = 0.035, \gamma_3 = 0.012, \lambda_1 = 0.01, \lambda_2 = 0.035, \lambda_3 = 0.15$
SVDGN	0.840132	$k = 400, \gamma_1 = 0.04, \gamma_2 = 0.035, \gamma_3 = 0.012, \lambda_1 = 0.008, \lambda_2 = 0.035, \lambda_3 = 0.15$
support	-	The natural logarithm of the number of ratings per user.

TABLE I. THE DATASET USED FOR THE FINAL PREDICATION BLENDING.

$\gamma_1 = 0.001$, $\lambda_1 = 0.01$, and number of iteration is 15, which gives mean_RMSE 0.906189. This model is not included in the final combination, but the b_u and b_i from this model are used to initialize the b_u and b_i when training other models.

For GN, SVD, SVDasym, SVD++, and SVDGN models, the best 5 parameter sets for each model and the corresponding mean_RMSE are shown in Table I. All models significantly improved the prediction accuracy, compared to Baseline model. The SVD based models (SVD, SVDasym, SVD++) had a better performance than neighborhood based model (GN), and combined neighborhood and SVD model (SVDGN) had smallest mean_RMSE.

All models listed in Table I were included in the final combination, along with the "support", which is the natural logarithm of the number of ratings per user. Blending these 26 inputs with ridge regression could achieve a RMSE of 0.84686 on Kaggle Quiz Set, and blending with neural network reaches a RMSE of 0.84662. The final prediction chosen for Kaggle Test Set is the latter one, and the final result is RMSE = 0.84536.

IV. CONCLUSION

The conclusion goes here.

ACKNOWLEDGMENT

We would like to thank Dr. Genevera Allen for the wonderful class and competition. We acknowledge the use of the cluster of Computational and Integrative Biomedical Research Center at Baylor College of Medicine and the cluster of Center for Statistical Genetics at Baylor College of Medicine for assistance in computation of this project. We also thank Michael Jahrer for his open software *ELF*.

D.F. and Z.X.H. conceived of the project, implemented the methods and performed parameter tuning. Z.X.H. performed blending and wrote the report.

REFERENCES

[1] Factorization meets the neighborhood: a multifaceted collaborative filtering model, http://public.research.att.com/ volinsky/netflix/kdd08koren.pdf

[2] Item-Based Collaborative Filtering Recommendation Algorithms, http://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf

[3] Lessons from the Netflix Prize Challenge, http://public.research.att.com/˜volinsky/netflix/sigkddexp.pdf

[4] Netflix update: Try this at home, http://sifter.org/˜simon/journal/20061211.html

[5] Improving regularized singular value decomposition for collaborative filtering, http://www.cs.uic.edu/ liub/KDD-cup-2007/proceedings/Regular-Paterek.pdf

[6] Combining Predictions for Accurate Recommender Systems, http://www.commendo.at/UserFiles/commendo/File/kdd2010-paper.pdf

[7] ELF - ensemble learning framework, http://elf-project.sourceforge.net/