Recommender Systems

New Approaches with Netflix Dataset

Robert Bell
Yehuda Koren
AT&T Labs

ICDM 2007

Presented by Matt Rodriguez

Outline

Overview of Recommender System Approaches which are Content based or Collaborative Filtering

New Nearest Neighbor model from the leaders of the Netflix Competition

Background

[Adomvicious, Tuzihilan IEEE 2005] Toward the Next Generation of Recommender Systems : A Survey of the State-of-the-Art and Possible Extensions

[Bell, Koren ACM 2007] Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights

Approaches for Recommender Systems

Content Based Approach: user is recommended items similar to the ones the user preferred in the past.

Collaborative Filtering: user is recommended items that people with similar tastes liked in the past

Hybrid Approach: combine the two approaches

Content Based approach

Content based system uses a utility function

$$s(u,i) = score(ContentBasedProfile(u), Content(i))$$

u: user

i: item

The Drawbacks:

1. Costs associated with gathering users profiles and extracting features from the content
2. Limited Content Analysis
3. Overspecialization
4. No notion of overall popularity; global effect
5. No new features for the new trends.

Collaborative Filtering approach

CF uses ratings from other users to recommend items. Useful when item features are expensive or difficult to determine.

Memory based approach uses a function to determine rating of a user item pair, the function examines the ratings of the nearest neighbors(NNs).

1 . Neighbor of the user

2. Neighbor of the item

Model based approach: uses the ratings to learn a model which is then used to make rating predictions.

Memory Based Approach

Derive a rating from similar users, or items. A similarity function determines the neighbors.

$$r_{ui} = \sum_{v \in \hat{U}} sim(u, v) \times r_{vi}$$

$$sim(u, v) = \frac{\sum_{i \in \hat{I}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in \hat{I}} (r_{ui} - \bar{r_u})^2 \sum_{i \in \hat{I}} (r_{vi} - \bar{r_v})^2}}$$

u, v: users
i: items
$\hat{U}$: set of similar users
$\hat{I}$: set of items rated by both users

Unified Collaborative and Content Based Model

x_{ij}: iteraction attributes
z_i: user attributes
w_j: item attributes

e_{ij}: noise
λ_i: user heterogeneity
γ_j: item heterogeneity

r_{ij}: rating

$$r_{ij} = x_{ij}\mu + z_i\gamma_j + w_j\lambda_i + e_{ij}$$

$$e_{ij} \sim N(0, \sigma^2)$$

$$\lambda_i \sim N(0, \Lambda)$$

$$\gamma_j \sim N(0, \Gamma)$$

The model parameters are $\mu, \sigma^2, \Lambda, \Gamma$

Pitfalls of Recommender Systems

How to handle new users or new items?

Overspecialization: Bad to recommend items that are *too* similar to items previously. Goal is to find *new* items that are desirable.

Sparsity:

1. Many users do not have sufficient ratings

2. Many items do not have sufficient ratings

Outline

Overview of Recommender System Approaches which are Content based or Collaborative Filtering

New Nearest Neighbor model from the leading group in the Netflix Competition

Scalable Collaborative Filtering with Jointly Derived Neighborhood Weights

Contributions are normalizing global effects through the dataset.

Capture interdependence of the neighbors by deriving their weights simultaneously.

Present a scalable technique for finding similarities between large number of users.

Previous Rating Techniques

Compare user-user similarities
or item-item similarities

$$R = r_{ui}\ _{1 \le u \le m, 1 \le i \le n}$$

$$r_{ui} = \frac{\sum_{v \in N(u;i)} s_{uv} r_{vi}}{\sum_{v \in N(u;i)} s_{uv}}$$

$$r_{ui} = \frac{\sum_{j \in N(u;i)} s_{ij} r_{uj}}{\sum_{v \in N(u;i)} s_{ij}}$$

R: user-item ratings $m \times n$ matrix
$N(u;i)$: set of neighbors to u
s_{uv}: simularity score between users
s_{ij}: simularity score between items
r_{ui}: rating by user u of item i
r_{vi}: rating by user v of item i

Koren-Bell New Strategy

1. Remove global effects.
2. Find Interpolation weights for the neighbors simultaneously, by solving an optimization problem
3. Reduce the dimensionality of users by performing an iterative version of SVD.

Addresses the following concerns

1. NN methods are not good at accounting for global effects.
2. Previous weighting methods fail to account for interdependencies between neighbors.
3. Previously, interpolation weights always sum to one . If there is no useful neighborhood information then it is better to ignore it.
4. NN methods work poorly if the number of ratings differs substantially.

Normalizing by Removing Global Effects

Some users may give higher ratings than others users.

Some items may systematically receive higher ratings that other items.

Ratings may change over time.

For global effects consider:

1. average rating for an item
2. average rating for a user
3. number of ratings for an item
4. number of ratings for a user

Parameter Estimation for Global Effects

Sequentially estimate one effect at a time. At each step use the residuals from the previous step.

$$r_{ui} = \theta_u x_{ui} + error$$

$$\hat{\theta_u} = \frac{\sum_i r_{ui} x_{ui}}{\sum_i x_{ui}^2}$$

r_{ui}: residual
x_{ui}: variable for user u and item i
θ_μ: user parameter

Parameter Estimation

For sparse data user parameter estimate is not reliable because there is too much variance.

$$\theta_u \sim N(\mu, \tau^2)$$

$$\hat{\theta}_u|\theta_u \sim N(\theta_u, \sigma_u^2)$$

Shrink the parameter based on how many ratings it had.

This is an example of Bayesian shrinkage.

Approximation in Estimating User Parameter

Koren-Bell use a simpler estimator used for the user parameter.

$$\theta_u = \frac{n_u \hat{\theta}_u}{n_u + \alpha}$$

The user parameter can be estimated from the posterior mean.

$$E(\theta_u|\hat{\theta}_u) = \frac{\tau^2 \hat{\theta}_u + \sigma_u^2 \mu}{\tau^2 + \sigma_u^2}$$

$$\tau^2 = \frac{\sum_u [(\hat{\theta}_u - u)^2 - \sigma_u^2]/(\tau^2 + \sigma_u^2)^2}{\sum_u (\tau^2 + \sigma_u^2)^{-2}}$$

Examples of Global Effects

Results are from the Probe dataset 1.4 million ratings.

Allows rating to change over time and based on average rating or support.

Effects are processed sequentially.

Effect	RMSE	Improvement
Overall mean	1.1296	NA
Movie effect	1.0527	.0769
User effect	0.9841	.0686
User$\times$Time(user)$^{1/2}$	0.9809	.0032
User$\times$Time(movie)$^{1/2}$	0.9786	.0023
Movie$\times$Time(movie)$^{1/2}$	0.9767	.0019
Movie$\times$Time(user)$^{1/2}$	0.9759	.0008
User$\times$Movie average	0.9719	.0040
User$\times$Movie support	0.9690	.0029
Movie$\times$User average	0.9670	.0020
Movie$\times$User support	0.9657	.0013

Table 1. RMSE for Netflix probe data after adding a series of global effects to the model

Neighborhood Relationships Model Overview

Compares item-item similarity.

Defines a model that can predict a rating given weights.

Sets up an optimization problem to derive the weights.

Accounts for sparsity by shrinking ratings with low support toward the common mean.

Jointly derives the weights using a QP solver.

Neighborhood Relationships Model

- Find the K most similar items using similarity function
- Provides a prediction for a rating once the weights have been calculated
- Computing the weights simultaneously is a least squares optimization problem.

$$\{w_{ij} | j \in N(i;u)\}$$

$$r_{ui} \leftarrow \sum_{j \in N(i;v)} w_{ij} r_{uj}$$

$$\min_{w} \sum_{v \neq u} \left(r_{vi} - \sum_{j \in N(i;u)} w_{ij} r_{vj} \right)^2$$

Optimal Weights

A is K x K matrix that stores item-item ratings by different users.

b is a vector with ratings from similar items multiplied by the current item.

This ignores many pairwise relationships among ratings by the same user

$$Aw = b$$

$$A_{jk} = \sum_{v \neq u} r_{vj} r_{vk}$$

$$b_j = \sum_{v \neq u} r_{vj} r_{vi}$$

Neighborhood Relationships Model

Normalizes A and b by dividing it by the number of users that rated items i and j.

$$\bar{A}_{jk} = \frac{\sum_{v \in U(j,k)} r_{vj} r_{vk}}{|U(j,k)|}$$

$$\bar{b}_j = \frac{\sum_{v \in U(i,j)} r_{vj} r_{vi}}{|U(i,j)|}$$

Neighborhood Relationship Model

This steps shrinks ratings that have not been rated by many users.

$$\hat{A_{jk}} = \frac{|U(j,k)| \cdot \bar{A_{jk}} + \beta \cdot avg}{|U(j,k)| + \beta}$$

$$\hat{b_j} = \frac{|U(i,j)| \cdot \bar{b_j} + \beta \cdot avg}{|U(j,k)| + \beta}$$

$$\hat{A}w = \hat{b}$$

avg is the average of all $\bar{A_{jk}}$ values.
β controls the extent of the shrinkage.
Typical values for β are 500.

Preprocessing

1. Calculate similarity between each item pairs.
2. Shrink sij based on then number of ratings
3. Normalize A based on the number of users rated each pair of items

$$s_{ij} = s_{ij0}\frac{|U(i,j)}{(|U(i,j)| + \alpha)}$$

$$\bar{A_{jk}} = \frac{\sum_{v \in U(j,k)} r_{vj} r_{vk}}{|U(j,k)|}$$

Preprocessing is quadratic in number of items, linear in ratings

Calculation of the weights

The weights are constrained to be non-negative. This makes it a QP optimization problem.

Determine the weights by using a quadratic algorithm.

Algorithm uses Gradient Projection method.

NonNegativeQuadraticOpt ($A \in \mathbb{R}^{K \times K}, b \in \mathbb{R}^K$)
% Minimize $x^T Ax - 2b^T x$ *s.t.* $x \geqslant 0$

do
  $r \leftarrow b - Ax$ *% the residual, or "steepest gradient"*
  % find active variables - those that are pinned due to
  % nonnegativity constraints; set respective r_i*'s to zero*
  for $i = 1, \ldots, k$ **do**
    if $x_i = 0$ and $r_i < 0$ **then**
      $r_i \leftarrow 0$
    end if
  end for
  $\alpha \leftarrow \frac{r^T r}{r^T A r}$ *% max step size*
  % adjust step size to prevent negative values:
  for $i = 1, \ldots, k$ **do**
    if $r_i < 0$ **then**
      $\alpha \leftarrow \min(\alpha, -x_i / r_i)$
    end if
  end for
  $x \leftarrow x + \alpha r$
while $\|r\| < \epsilon$ *% stop when residual is close to 0*
return x

Figure 1. Minimizing a quadratic function with non-negativity constraints

Rating process

1. Compute item-item similarity score
2. Compute item- item matrix
3. Given a user and an item, find the neighboring K items.
4. Use the QP solver to solve K x K system of equations which derives the weights.
5. Compute the rating for that specific user and item.

User Oriented computation

Many internet recommendation system will have more users than items.

If a user has not rated items similar to an item, we can find other users that are similar and derive a rating from them.

Users can passively provide other information to the system: transactions, web page views use this information to find similar users

Low dimensional embedding of the users

$R_{m \times n}$ matrix with m users and n items. Each user is in a space within n movies. SVD computes best rank-f approximation of R^f.

$R^f = P_{m \times f} Q_{n \times f}$ where $||R - PQ^T||_F$ is minimized.

When all entries of R are known SVD is equivalent to the following optimization problem.

$$\min \sum_{(u,i) \in \kappa} (r_{ui} - p_u^T q_i)^2$$

Regularizing R

Not all entries of R are know so regularize R by performing ridge regression. The regularization penalizes the model by the norm of each p_u and q_i. λ is typically 0.05.

$$\min \sum_{(u,i)\in\kappa} (r_{ui} - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

Iteratively perform SVD

1. Fix Q, solve for P.
2. Fix P, solve for Q.
3. Repeat until convergence.

$Q[u]$ is a $n_u \times f$ matrix that contains ratings of items by one user u.
$r_u \in \mathbb{R}$ contains the ratings for the items in $Q[u]$
p_u is a row in the P matrix
Λ is a $f \times f$ diagonal matrix with entries $\sqrt{\lambda n_u}$

$$\begin{pmatrix} Q[u] \\ \Lambda \end{pmatrix} p_u = \begin{pmatrix} r_u \\ 0 \end{pmatrix}$$

Unifying user and item-relations in single model

A different user v might ratings might be a good predictor for some items but for all items

$$\min_{w} \sum_{j \neq i} s_{ij} \left(r_{uj} - \sum_{v \in N(u;i)} w_{uv} r_{vj} \right)^2$$

Gives higher weight to items that are similar to i.

Experimental Study

The experiments were run on the Netflix Quiz and Probe test sets.

The test sets contain results from users that do not rate very much which makes it more difficult to predict.

The results measured the RMSE.

Experiments with the Probe set

1. Jointly interpolated weights performed better than standard correlation based weights.
2. Neighborhood model improved predictions after all data normalizations.
3. Neighborhood size of K depended on the amount of previous normalization.
4. Neighborhood interpolation had diminishing returns after global effects are removed.

Data normalization	No interpolation ($k = 0$)	Correlation-based interpolation $k = 20$	$k = 35$	$k = 50$	Jointly derived interpolation $k = 20$	$k = 35$	$k = 50$
none (raw scores)	NA	0.9947	1.002	1.0085	0.9536	0.9596	0.9644
double centering	0.9841	0.9431	0.9470	0.9502	0.9216	0.9198	0.9197
global effects	0.9657	0.9364	0.9390	0.9413	0.9194	0.9179	0.9174
factorization	0.9167	0.9156	0.9142	0.9142	0.9071	0.9071	0.9071

Experiments with the Quiz set

Uses information derived from the Probe set.

Results are better than Probe Set.

RMSE is .9086 when global effects are removed.

RMSE is .8982 when factorization is applied.

Experiments with the user-oriented approach

User-User experiments have a larger neighborhood size of 100.

Low dimensionality embedding improved query

Accuracy is not as good as the item-item. RMSE is 0.9157 after global effects are removed.

Future approach that uses a mixture model of user-user and item-item to provide ratings

Summary

1. Remove global effects.
2. Find Interpolation weights for the neighbors simultaneously, by solving an optimization problem
3. Reduce the dimensionality of users by performing an iterative version of SVD.

Sources

[BK07], [BKV07], [Adomvicious, Tuzihilan IEEE 2005]

Presented by Matt Rodriguez

mar008@cs.ucsd.edu