Collaborative Filtering Ensemble

KDD Cup 2011 - Track 1 (2nd place)

Michael Jahrer, Andreas Töscher

commendo research & consulting GmbH

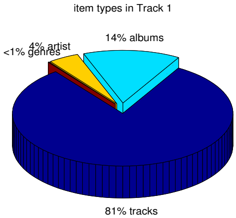
August 21^{th}, 2011

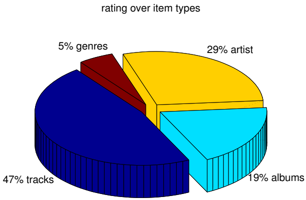
Approach Overview

- Problem is similar to the Netflix Prize $\rightarrow$ ERROR = RMSE
- We had experience in optimizing the RMSE
- So our approach is:
  - use collaborative filtering $\rightarrow$ sparse matrix
  - use all the ideas from the Netflix Prize
  - re-write and optimize them
  - use the time information for data cleaning
  - apply models on residuals of others
  - blend them together using a neural network
- Sadly we do not exploit the full potential of the taxonomy source (like in Track 2) ☹

The Dataset I

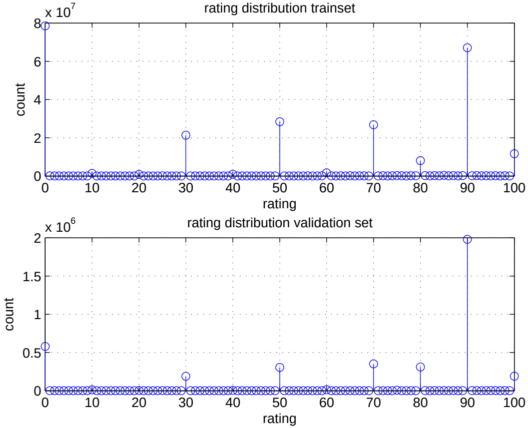
#ratings	262M
#users	1M
#items	625k
→ #tracks	507k
→ #albums	89k
→ #artists	28k
→ #genres	1k
rating date?	yes ☺





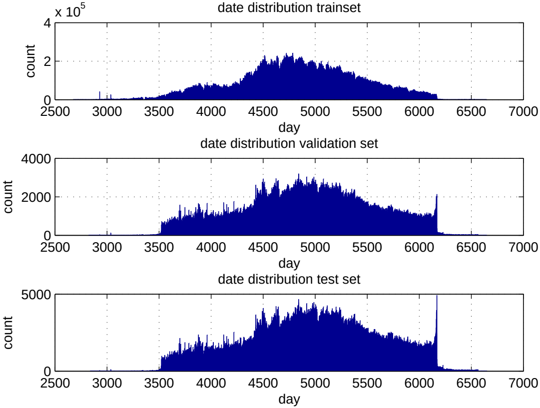
The Dataset II

- Ratings: 0, 30, 50, 70, 90 have highest occurence
- But we do not use this observation anyway



The Dataset III

- approx. 10 years of data



Baselines

predictor	validation RMSE
global mean (μ=48.8)	38.2138
item mean	32.3941
user mean	27.6525

- Our best submission: RMSE=18.9092 on the validation set

Algorithms

First approach

- How can we clear the data ?

Clear the data 1/5

- Introduce simple biases
- Learned by stochastic gradient descent

symbol	name	#parameters
μ	global bias	1
μ_u	user bias	1,000,990
μ_i	item bias	624,961
μ_{minute}	global minute bias	5,726,101
μ_{hour}	global hour bias	95,436
μ_{day}	global day bias	3,978
$\mu_{u,day}$	user day bias	13,027,048

$$\hat{r}_{ui}^{(0)} = \mu + \mu_u + \mu_i + \mu_{minute} + \mu_{hour} + \mu_{day} + \mu_{u,day}$$

Clear the data 2/5

- user day bias $\mu_{u,day}$
- The bias is not predictive !
- Ratings of the testset are in the "future" $\rightarrow \mu_{u,day} = 0$
- Only helps in estimating better values for e.g. μ_u, μ_i

Clear the data 3/5

- Introduce frequency biases
- $freq(u, day)$: #ratings from user u on the particular day
- $freq(i, day)$: #ratings of item i on the particular day
- We limit the frequency to < 10

$$\begin{aligned}\hat{r}_{ui}^{(0)} =&\mu + \mu_u + \mu_i + \mu_{minute} + \mu_{hour} + \mu_{day} + \mu_{u,day} + \\ &\mu_{u,freq(u,day)} + \mu_{i,freq(i,day)}\end{aligned}$$

Clear the data 4/5

- Introduce time factorization
- user x minute, user x hour, user x day, item x minute, ...
- F = # features

symbol	name	#parameters
$\mathbf{p}_u^{(0)}$	user feature	F x 1,000,990
$\mathbf{y}_{minute}^{(0)}$	minute feature	F x 5,726,101

- dot product = prediction: $\mathbf{p}_u^{(0)^T} \mathbf{y}_{minute}^{(0)}$

Clear the data 5/5

- $\rightarrow$ The "Baseline model"
- Does not learn any "user - item" interactions !
- Only clears the data from time effects

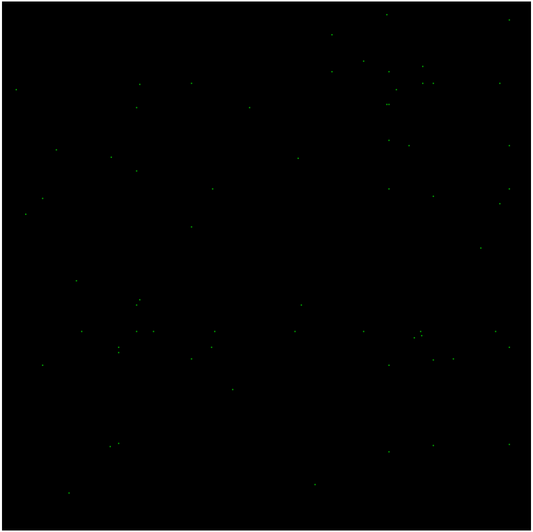
$$\hat{r}_{ui}^{(0)} = \mu + \mu_u + \mu_i + \mu_{minute} + \mu_{hour} + \mu_{day} + \mu_{u,day} + \mu_{u,freq(u,day)} + \mu_{i,freq(i,day)} + \mathbf{p}_u^{(0)^T} \mathbf{y}_{minute}^{(0)} + \mathbf{p}_u^{(1)^T} \mathbf{y}_{hour}^{(0)} + \mathbf{p}_u^{(2)^T} \mathbf{y}_{day}^{(0)} + \mathbf{q}_i^{(0)^T} \mathbf{y}_{minute}^{(1)} + \mathbf{q}_i^{(1)^T} \mathbf{y}_{hour}^{(1)} + \mathbf{q}_i^{(2)^T} \mathbf{y}_{day}^{(1)}$$

- RMSE=19.67 on the training set (50 features)
- RMSE=23.37 on the validation set
- RMSE=25.20 on the leaderboard

Second approach

- How can we factorize the rating matrix $\mathbf{R}$?

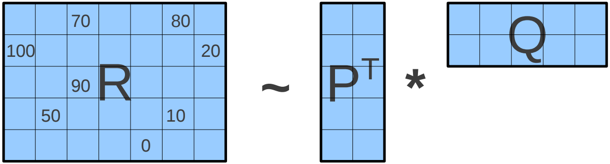
The sparse matrix



- Small part of $\mathbf{R}$
- Fill rate: 0.04%
- Green dots: ratings
- users: 10100..10600
- items: 10100..10600

Basic SVD

- $\mathbf{R}$ has the size of 1,000,990 x 624,961
- $\mathbf{R} \approx \mathbf{P}^T \cdot \mathbf{Q}$
- $\mathbf{P}$ are the F-dim. user features $\mathbf{p}_u$
- $\mathbf{Q}$ are the F-dim. item features $\mathbf{q}_i$
- user/item prediction is a dot product $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$



Basic SVD Training: SGD

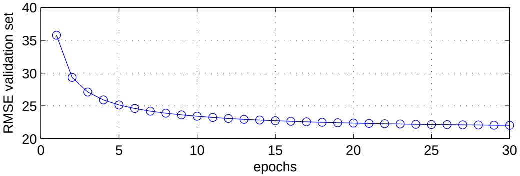
Input: Sparse rating matrix $\mathbf{R} \in \mathbb{R}^{|U|\times|I|} = [r_{ui}]$
Tunable: Learning rate η, Regularization λ, feature size F

Initialize user weights $\mathbf{p} \in \mathbb{R}^{F\times|U|}$ and item weights $\mathbf{q} \in \mathbb{R}^{F\times|I|}$ with small random values
while *error on validation set decreases* **do**
    forall $u, i \in \mathbf{R}$ **do**
        $\hat{r}_{ui} \leftarrow \mathbf{p}_u^T \mathbf{q}_i$
        $e \leftarrow \hat{r}_{ui} - r_{ui}$
        for $k = 1 ... F$ **do**
            $c \leftarrow p_{uk}$
            $p_{uk} \leftarrow p_{uk} - \eta \cdot (e \cdot q_{ik} + \lambda \cdot p_{uk})$
            $q_{ik} \leftarrow q_{ik} - \eta \cdot (e \cdot c + \lambda \cdot q_{ik})$
        end
    end
end

Algorithm 1: Pseudo code for training a SVD on rating data.

Basic SVD: Performance

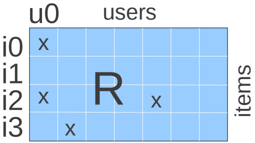
- 100 features
- small learning rate $\eta = 0.00025$
- large regularization $\lambda = 0.5$
- 30 epochs
- 300[s] per epoch $\rightarrow$ 5[h] total time
- RMSE=22.03 on the validation set
- RMSE=23.92 on the leaderboard



AFM: The asymmetric idea

- The user is represented by its rated items
- The user feature is a sum of rated item features
- $\rightarrow$ "virtual user feature" $\mathbf{v}_u$

$$v_u = \sqrt{\frac{1}{|I(u)|}} \cdot \sum_{j \in I(u)} q_j^{(0)}$$



$$\boxed{v_0} = \sqrt{\frac{1}{2}} \left(\boxed{q_0^{(0)}} + \boxed{q_2^{(0)}} \right)$$

$\leftarrow$ virtual user feature for user 0
bag of i0 and i1

This is used for normalization
Assumption: values in $Q^{(0)}$ are normal distributed

AFM: The asymmetric idea

- "AFM" model
- Add item and user biases
- Trained with gradient descent

$$\hat{r}_{ui} = \mu_u + \mu_i + \overbrace{\mathbf{q}_i^T}^{item feature} \overbrace{\left(\frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)} \right)}^{user feature}$$

- RMSE=23.54 on the validation set (30 features)

ASVD: Combines AFM and SVD

- “ASVD” model
- user is represented by a user feature and the sum of rated item features

$$\hat{r}_{ui} = \mu_u + \mu_i + \overbrace{\mathbf{q}_i^T}^{item feature} \overbrace{\left(\mathbf{p}_u + \frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)}\right)}^{user feature}$$

- RMSE=21.92 on the validation set (30 features)

Enhance the SVD apporach

- All factor models are trained with gradient descent
- Easy to add different feature parts together
- Idea: Combine SVD + Baseline model

$$\hat{r}_{ui}^{(2)} = \hat{r}_{ui}^{(0)} + \hat{r}_{ui}^{(1)}$$

- RMSE=20.95 on the validation set (30 features)
  $\rightarrow$ RMSE=22.97 on leaderboard
  $\rightarrow$ our best single model

Thinking different

- Flip the asymmetric idea
- Works well
- Approach: item = bag of rated users
  - "AFM flipped" model

$$\hat{r}_{ui} = \mu_u + \mu_i + \overbrace{\mathbf{p}_u^T}^{userfeature} \overbrace{\left(\frac{1}{\sqrt{|\mathbb{U}(i)|}} \sum_{j \in \mathbb{U}(i)} \mathbf{p}_j^{(0)} \right)}^{itemfeature}$$

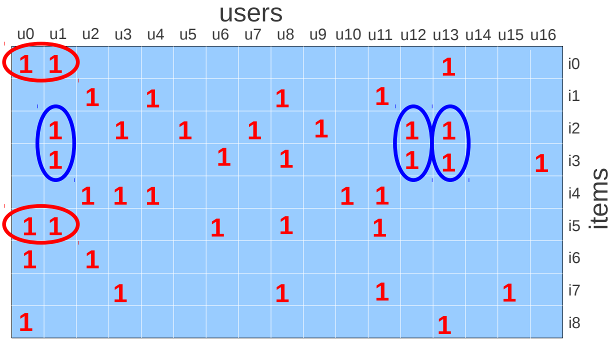
  - "ASVD flipped" model

$$\hat{r}_{ui} = \mu_u + \mu_i + \mathbf{p}_u^T \left(\mathbf{q}_i + \frac{1}{\sqrt{|\mathbb{U}(i)|}} \sum_{j \in \mathbb{U}(i)} \mathbf{p}_j^{(0)} \right)$$

- This models were helpful within the ensemble

Neighborhood models

- Based on similarity measure
- Item-Item similarity
- Precalculation ? $\rightarrow$ 781GB RAM ☹



1 ... Rating

Item-Item similarity

- Precalculation: Would not fit in RAM
- On-the-Fly: Too slow (would take days)
- Solution: use the inverse euclidean feature distance
  sim(i,j) = ? (i,j are items)
  Item features: $\mathbf{q}_0$ and $\mathbf{q}_1$

$$c_{ij} = \left(\frac{\sum_{k=1}^{F} (q_{ik} - q_{jk})^2}{\sqrt{\sum_{k=1}^{F} q_{ik}^2}\sqrt{\sum_{k=1}^{F} q_{jk}^2}} \right)^{-2}$$

- Constant calculation time O(1)

Neighborhood model

- "Item-Item KNN with SVD Features" model

$$\hat{c}_{ij} = 0.5 \cdot tanh(\sigma \cdot c_{ij} + \gamma) + 0.5$$

$$\hat{r}_{ui}^{(12)} = \frac{\sum_{j \in \mathbb{I}(u,i,K)} r_{uj} \hat{c}_{ij}}{\sum_{j \in \mathbb{I}(u,i,K)} |\hat{c}_{ij}|}$$

- 3 meta parameters: K, σ and γ
- RMSE=21.6743 on the validation set (on residuals of SVD)

User-User similarity

- Flipped to the user side
- "User-User KNN" model
- Not helpful in the ensemble
- Neighborhood information is already covered by the Item-Item model

$$c_{uv} = \left(\frac{\sum_{k=1}^{F} (p_{uk} - p_{vk})^2}{\sqrt{\sum_{k=1}^{F} p_{uk}^2} \sqrt{\sum_{k=1}^{F} p_{vk}^2}} \right)^{-2}$$

Restricted Boltzmann Machines

- Algorithm template from: “Restricted Boltzmann Machines for Collaborative Filtering” from R.Salakhutdinov, A.Mnih, G.Hinton. In ICML 2007
- In the data: 101 discrete ratings
- We compress them to 11 and 2 ratings
- RBMs were helpful in the ensemble

Blending

Blending

- Huge difference in validation vs. leaderboard RMSE !
- Approx. $\approx$ 1.9 of the RMSE
- Why they are so far apart?
- Code bug?
- Maybe the leaderboard set is very small?

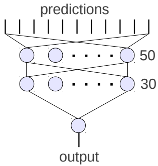
model	RMSE valid	RMSE leaderboard	leaderboard - valid
AFM	23.5492	25.1584	**1.6092**
AFM flipped	22.3234	24.2555	**1.9321**
ASVD	21.9202	23.8580	**1.9378**
ASVD + Baseline	21.1875	23.119	**1.9315**
ASVD flipped	22.817	24.6345	**1.8175**
Baseline	23.3766	25.2068	**1.8302**
Item-Item KNN SVD feat	21.6743	23.6730	**1.9987**
SVD + Baseline	20.9507	22.9765	**2.0258**
...	...	...	...

Blending Approaches

- Standard way: Use validation set to build the ensemble
- Netflix Prize Trick: Use leaderboard feedback to make a linear regression
  - We rejected it because of the huge difference of $RMSE_{leaderboard} - RMSE_{validation}$
  - Also impractical in real environments, no leaderboard available

Blending

- 47 predictions in the ensemble
- Neural Network
- 2 hidden layers
- Gradient descent learning on validation set
- We do not clip the predictions before blending
    - Best single: RMSE=20.68 (leaderboard 22.80)
    - Blending: RMSE=18.90 (leaderboard 21.08)



Summary

- Apply different collaborative filtering models
- Train some of them on residuals of others
  - KNN for postprocessing
- Use the time for data cleaning
- Add basic statistics of the taxonomy
- Not explored: “taxonomy info” in models
  - Reason for 2nd place ☺?!

and now ... Track 2

commendo research & consulting GmbH
AUSTRIA