Lessons from the Netflix prize

Nisheeth

The road to the Netflix Prize

- SVD
- SVD++
- Temporal SVD
- Temporal SVD++
- Highly recommended reading: Madrigal (2014) How Netflix reverse-engineered Hollywood

Netflix challenge

- Ratings on 1-5 scale
  - 480000 users
  - 17770 movies
- Metric $\sqrt{\frac{\sum_{u,i}(r_{ui} - \hat{r}_{ui})^2}{N}}$
- Reduce baseline RMSE of 0.9514 by 10% - win a million $

Koren-Bell approach

- Basic insight
  - SVD captures user-item interaction
  - Model user and item biases separately
  - Include information from temporal dynamics
  - Include role of implicit feedback
    - People rate some movies for a reason, no matter what rating they eventually assign

Modeling user and item biases

- Using simple linear models
- $b_{ui} = \mu + b_i + b_u$
- Learn the parameters using a least squares loss function
- $min_{b*} \sum_{ui}(r_{ui} - \mu - b_i - b_u)^2 + \gamma \left(\sum_i {b_i}^2 + \sum_u {b_u}^2\right)$
- What is the second term here doing? Is it necessary?
- Typically fit using alternative least squares or stochastic gradient descent
- SGD most popular
  - Iterate across all parameters
  - Calculate prediction error
  - Shift new parameter estimates in opposite direction of error, multiplied by error magnitude

Adding SVD to the baseline

- Added by simply adding
- $min_{b*,p,q} \sum_{ui}(r_{ui} - \mu - b_i - b_u - p_u q_i)^2 + \gamma\left(\sum_i {b_i}^2 + \sum_u {b_u}^2 + \|p_u\|^2 + \|q_i\|^2\right)$
- Parameters continue to be learned via SGD

Incorporating implicit feedback

- User trait factors learned by SVD live in p_u
- Augment with implicit feedback information
- Implicit feedback = which items were rated at all
- Revised user model
- $p_u + \frac{\sum_{j \in R(u)} y_j}{\sqrt{|R(u)|}}$
- The y_j are parameters learned from data also – interpret as item-specific traits

Incorporating time dynamics

- User preferences shift over time
- Assume this in the baseline predictor

$$b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui})$$

- Design an item time drift model
  - Simple, just fit a histogram

$$b_i(t) = b_i + b_{i,\text{Bin}(t)}$$

- Design a user time drift model
  - Not as straightforward, have to deal with psychology

User time drift model

- If a user rated a movie at time t, model the time drift of the rating with respect to the mean rating time t_u as
- $dev_u(t) = sgn(t - t_u)|t - t_u|^{\beta}$
- User model becomes
- $b_u(t) = b_u + b_{u,t} + \alpha_u \, dev_u(t),$
  - $b_{u,t}$ is meant to fit rating spikes in time

Combining dynamic baseline with SVD

- Make the user trait model dynamic also

$$p_{uk}(t) = p_{uk} + \alpha_{uk} \cdot \text{dev}_u(t) + p_{uk,t}$$

- Put the pieces together to get the final prediction rule
- $\hat{r}_{ui} =$
  $\mu + b_i(t_{ui}) + b_u(t_{ui}) +$
  $q_i \left(p_u(t_{ui}) + \frac{\sum_{j \in R(u)} y_j}{\sqrt{|R(u)|}} \right)$
- Learn all the parameters using SGD

Outcome

Model	f=10	f=20	f=50	f=100	f=200
SVD	.9140	.9074	.9046	.9025	.9009
SVD++	.9131	.9032	.8952	.8924	.8911
timeSVD++	.8971	.8891	.8824	.8805	.8799

Micro-genres instead of parameters

Emotional Independent Sports Movies
Spy Action & Adventure from the 1930s
Cult Evil Kid Horror Movies
Cult Sports Movies
Sentimental set in Europe Dramas from the 1970s
Visually-striking Foreign Nostalgic Dramas
Japanese Sports Movies
Gritty Discovery Channel Reality TV
Romantic Chinese Crime Movies
Mind-bending Cult Horror Movies from the 1980s
Dark Suspenseful Sci-Fi Horror Movies
Gritty Suspenseful Revenge Westerns
Violent Suspenseful Action & Adventure from the 1980s
Time Travel Movies starring William Hartnell
Romantic Indian Crime Dramas
Evil Kid Horror Movies
Visually-striking Goofy Action & Adventure
British set in Europe Sci-Fi & Fantasy from the 1960s
Dark Suspenseful Gangster Dramas
Critically-acclaimed Emotional Underdog Movies

Hundreds of raters

36 page rating manual

Template:

Region + Adjectives + Noun Genre + Based On... + Set In... + From the... + About... + For Age X to Y

This is what a real-world recommender has to do

Netflix recommender workflow

Netflix Quantum Theory

Syntactic combination

Content-based CF

Content-boosted collaborative filtering

Based on content features additional ratings are created

E.g. Alice likes Items 1 and 3 (unary ratings)

Item7 is similar to 1 and 3 with sim rating 0.75

Thus Alice likes Item7 weighted by 0.75

Item matrices become less sparse