Taxonomy-informed latent factor models for implicit feedback

Andriy Mnih

Gatsby Computational Neuroscience Unit
University College London

KDD Cup Workshop
August 21, 2011

Introduction

- Track 2 task: discriminate between the highly rated and the unrated tracks for each user.
- Classic implicit feedback modelling problem
  - Discriminate between the observed positive items and the unobserved negative items
- Some twists:
  - Negative items are sampled from the empirical distribution of positive items
  - Taxonomy information is available
  - Item ratings are known

Overview

- Approach: Latent factor models for ranking items
- Learn a scoring function $s_u(i)$ for each user u so that

$$s_u(\text{highly rated item}) > s_u(\text{unrated item})$$

- Use item taxonomy information in two ways:
  - To define a hierarchical parameterization of item representations
  - As a guide for extracting user history features relevant to an item
- Training algorithm that takes advantage of
  - the known distribution of unrated items and
  - the known number of unrated items per user in the test set.
- Build on the matrix factorization version of Bayesian[1] Personalized Ranking (Rendle et al., 2010).

[1] In spite of its name BPR is not actually a Bayesian method.

Learning a scoring function (BPR)

- Represent users and items using vectors of latent factors
- Parameterize the score of item i for user u in terms of their latent vectors:

$$s_u(i) = U_u^\top V_i + b_i,$$

  - U_u user factor vector; V_i item factor vector
  - b_i item bias (captures popularity)
- Model the probability of ranking a positive example i above a negative example j as

$$P_u(i > j) = \text{logistic}(s_u(i) - s_u(j)).$$

- Train the model by maximizing the log-likelihood using stochastic gradient ascent.
  - For each positive example in the training data, sample a negative example from some distribution.
  - BPR generates negative examples from the uniform distribution.

Generating negative examples

- The distribution of negative examples has a strong effect on model performance.
- Track 2 evaluation protocol:
  - Separate three highly rated from three unrated items per user
  - Unrated items come from the empirical distribution of the highly rated items.
- Thus want to score each highly rated item higher than three unrated items sampled from the distribution of highly rated items.
- Negative item generation: sample three items from the distribution of the highly rated items and keep the one with the highest score.
- Sampling negative items from the empirical distribution efficiently:
  1. Pick a training case (user, positive item, rating) uniformly at random
  2. Return the item from the training case as the sample

Taxonomy-based hierarchical item parameterization

- Taxonomy can serve as a source of track similarity information.
- **Idea:** Ensure that tracks by the same artist or from the same album have similar representations unless the data suggests otherwise.
- New track representation: a linear combination of the old unconstrained representation and the representations of its album and artist.

$$V_i = V_i^{old} + w_{album} V_{album(i)} + w_{artist} V_{artist(i)}.$$

- Enables information transfer between
  - tracks by same the artist / off the same the album
  - tracks and their artists
  - tracks and their albums
- Incorporating genre information in a similar manner did not seem to help.

Taxonomy-based features

- Can also use item similarity information provided by the taxonomy to extract features from a user's rating history.
  - A more direct use of the information than the taxonomy-based parameterization.
- **Idea:** Summarize the user's rating history as it relates to the item of interest using a vector of features $f(i, u)$.
- Use features as inputs to the scoring function:

$$s_u(i) = U_u^\top V_i + b_i + (W_u + W_i + W)^\top f(i, u).$$

  - W_u, W_i, and W are user-specific, item-specific, and global feature weights respectively.
- Since $f(i, u)$ has to be computed for randomly sampled negative items, cannot precompute all features before training.
  - Store user rating information in data structures that allow efficient search (e.g. hash tables).
  - Precompute as much as possible.

Best features

Features that helped all models regardless of the number of latent factors:

1. The rating given to the track's album by the user (divided by 100)
2. Is the album's rating 80 or higher?
3. The rating given to the track's artist by the user (divided by 100)
4. Is the artist's rating 80 or higher?

Potentially helpful features

Features that helped models with few latent factors but hurt or did not help models with many:

5. The fraction of the track's genres rated by the user (either directly or by rating an album or a track of that genre)
6. Has the user rated (directly or indirectly) any of the track's genres?
7. The fraction of user's genre ratings the track's genres account for
8. Has the user rated any other track from the album?
9. Has the user rated any other track by this artist?
10. The mean rating of the track's genres that have been rated by the user (divided by 100)

Training procedure

- Train on *both* tracks and non-tracks rated 50 and above.
  - Training on tracks alone does not work as well.
  - Downweight the contribution of items rated below 80 by 0.25.
- Early stopping on the validation set as well as *L2* regularization to avoid overfitting.
- Unless stated otherwise, models have 100 latent factors.

Validation set

- A validation set was used for monitoring model performance during training as well as for quick model evaluation without using the KDD Cup website.
- Creating a validation set:
  - For each user in the test set:
    - Positive tracks: Three highly rated tracks from the training set.
    - Negative tracks: Three tracks sampled from the empirical distribution of tracks rated 80 or higher in the training set.
- Validation scores were highly correlated with the leaderboard scores.
  - Validation scores were about 0.45% higher.
  - A model with a validation error rate (ERR) of 5.45% will have a leaderboard ERR of about 5%.
- All scores reported in this talk are validation set scores.

Effect of the negative item distribution

How important is it to match the distribution of negative items used for training to the one for the test set?

Distribution of negative items	Validation ERR (%)
Uniform	9.274
Same as for positive items	5.350

Conclusion: Sampling negative items from the right distribution is crucial.

Ranking vs. classification

Is the ranking objective function superior to the classification objective for learning scoring functions?

Objective function	Validation ERR (%)
Ranking	5.350
Classification	5.556

Conclusion: Rankers outperform classifiers, but not by much.

Using taxonomy information

Which way of using taxonomy information is more effective?

Features used	Parameterization	Validation ERR (%)
—	Flat	6.017
—	Taxonomy-based	5.350
Best 4 (1-4)	Flat	3.916
Best 4 (1-4)	Taxonomy-based	3.708

Conclusion: Carefully selected taxonomy-based features give a larger performance boost than the taxonomy-based parameterization. Combining the two leads to the best-performing model with the test set ERR of 3.3142.

Number of latent factors vs. usefulness features

Can features be harmful?

Features used	Number of latent factors	Validation ERR (%)
Best 4 (1-4)	0	11.654
All	0	7.186
Best 4 (1-4)	100	3.708
All	100	3.977

Conclusion: Using too many features can easily hurt generalization, especially if the model has many latent factors.

Contribution of feature weights

Are user-specific and item-specific feature weights actually useful?

Feature weights used	Validation ERR (%)
All	3.708
User only	3.830
Item only	4.003
Global only	4.051
None	5.350

Conclusion: Global feature weights do most of the work, but user-specific weights are still helpful.

Discussion

- Latent factor models can be very effective at modelling implicit feedback.
- Using the right distribution for negative items is crucial.
- Using taxonomy information:
  - Taxonomy-based parameterization of item representations is an easy way to improve model performance without considerably increasing running time.
  - Taxonomy-based features can improve model performance significantly, but care must be taken to avoid overfitting.
  - Artist and album information is much more useful than genre information.
  - Price to pay for using features: considerably slower training.
- Latent factors beat (my) feature engineering but the combination of the two works best.

The End