Recommender System — Bayesian personalized ranking from implicit feedback

Walk Through Recommender System of Bayesian Personalized Ranking (BPR) & Adaptive K-nearest-neighbor

Denise Chen

Photo by Jeff Sheldon on Unsplash

It’s more prevalent to see the companies use the recommended system algorithm to produce users’ favorite items based on the previous shopping experience. Online customers would get recommended items while shopping online from stores like eBay, Amazon, Walmart, etc. The article is focused on item recommendations. The item recommendation system method provides a user-specific ranking for a set of items learned from users’ past datasets such as buying history, viewing history, etc. Nowadays, the recommendation system varies a lot from the input of explicit dataset like ratings to an implicit dataset such as monitoring clicks, view times, purchases, etc. This information is easier to collect, but hard to recommend the users’ favorite items.

In this article, you will learn the singular value decomposition and truncated SVD of the recommender system:

(1) Personalized Ranking System

(2) Problem Statement

(3) Bayesian Personalized Ranking (BPR)

(4) BPR Optimization Criterion

(5) BPR Learning Algorithm

(6) Matrix Factorization

(7) Adaptive K-nearest-neighbor

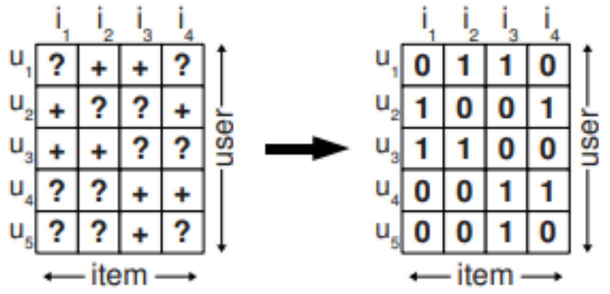
Personalized Ranking System

The personalized ranking provides customers with item recommendations of a ranked list of items. The article would focus on recommending customers with a personalized ranked list of items from users’ implicit behavior derived from the past purchase data. Observed from the purchase data, it is available to get the positive observations like users' bought history, whereas non-observed user-item pairs data is difficult for the model input like the unbought item, non-interested items, or the interesting items for their future purchase.

Problem Statement

For the implicit feedback systems, it is able to detect the positive dataset like bought history. For the remaining data, it is a mixture of actually negative and missing values. Nevertheless, machine learning models are unable to learn the missing data. Typically, the item recommenders output the personalized score X_ui based on the preference of the user for the item, and items are sorted form the predicted score. The machine learning model of item recommenders provides the training data with giving pairs (u, i) ∈ S as a positive class label and all other combinations in (U × I) \ S a negative one.

The model is fit to predict the positive class with value 1 and 0 for the rest. The problem would occur when the model is unable to rank the items ((U × I) \ S), which have been given as negative feedback during the training. An alternative would be to add regularization to the model to prevent overfitting. Another method is to create item pairs as training data and optimize for correctly ranking item pairs instead of scoring single item while the missing values are taken care of. From the photo below, it's hard for the model to only learn from the observed data. Therefore, all the negative data is replaced with 0.



	i_1	i_2	i_3	i_4
u_1	?	+	+	?
u_2	+	?	?	+
u_3	+	+	?	?
u_4	?	?	+	+
u_5	?	?	+	?

	i_1	i_2	i_3	i_4
u_1	0	1	1	0
u_2	1	0	0	1
u_3	1	1	0	0
u_4	0	0	1	1
u_5	0	0	1	0

Photo0: The observed 0 value and unobserved 1 value.

Bayesian Personalized Ranking (BPR)

To overcome the personalized ranking task, the Bayesian personalized ranking incorporates the Bayesian analysis of the problem using the likelihood function for p(i >u j|Θ) and the prior probability for the model parameter p(Θ).

In this section, we derive a generic method for solving the personalized ranking task. It consists of the general optimization criterion for personalized ranking, BPR-Opt, which will be derived by a Bayesian analysis of the problem using the likelihood function for p(i >u j|Θ) and the prior probability for the model parameter p(Θ)

BPR Optimization Criterion

The Bayesian approach is to produce the rankings for all items i ∊ I to maximize the following posterior probability where Θ represents the parameter vector of an arbitrary model class

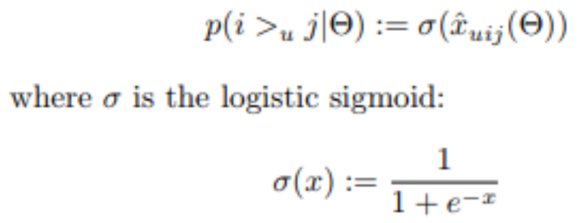
> $p(\Theta| >u) \propto p(>u |\Theta) p(\Theta)$

All user factors are independent with each other, and the ordering of each pair of items (i, j) for a specific user is unique.

Hence, the above user-specific likelihood function can be reproduced with the formula below.

User’s likelihood function

We define the individual probability that a user really prefers item i to item j as:



$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

where σ is the logistic sigmoid:

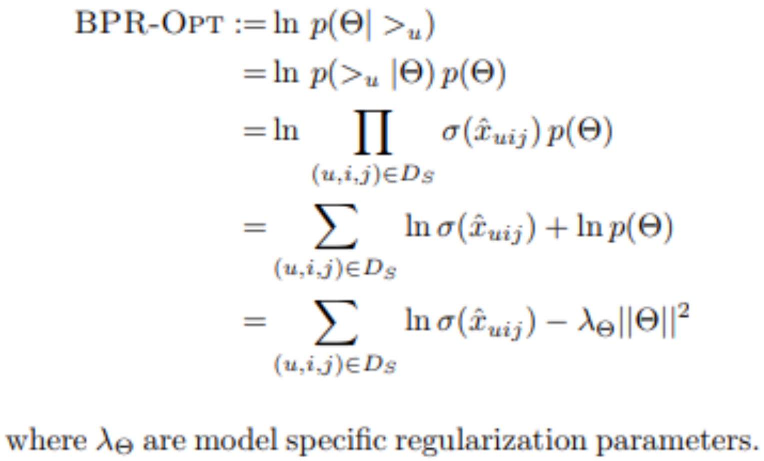
$$\sigma(x) := \frac{1}{1+e^{-x}}$$

User’s probability on the preference over item i to item j

For the Bayesian modeling approach of the personalized ranking task, a general prior density p(Θ) is introduced, which is a normal distribution with zero mean and variance-covariance matrix ΣΘ.

> *p(Θ) ~ N(0, ΣΘ)*

By setting ΣΘ = λΘI, we reduce the number of unknown hyper-parameters. The maximum posterior estimator is set to derive the optimization criterion for personalized ranking BPR-Opt.



$$\begin{aligned}
\text{BPR-OPT} &:= \ln p(\Theta| >_u) \\
&= \ln p(>_u |\Theta)\, p(\Theta) \\
&= \ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij})\, p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}$$

where λ_Θ are model specific regularization parameters.

BPR-Opt mathematic formula

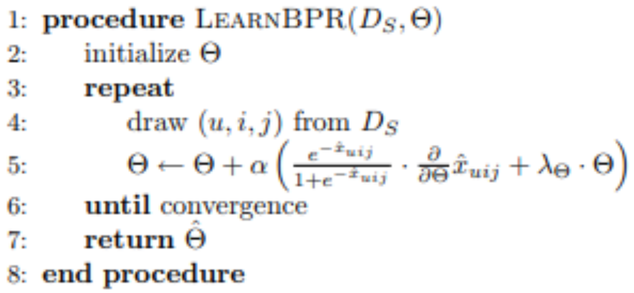
BPR Learning Algorithm

From the section above, the criterion is derived from personalized ranking, and standard gradient descent is not proper to cope with the problem. Then, LearnBPR is introduced as a stochastic gradient-descent algorithm to optimize model performance.

The gradient of BPR-Opt with respect to the model parameters is:

Gradient BPR-Opt mathematic formula

The parameters in the optimized model are specified as the learning rate α and regularization λΘ. The model is a bootstrapping based stochastic gradient descent.



```
procedure LearnBPR($D_S, \Theta$)
    initialize $\Theta$
    repeat
        draw $(u, i, j)$ from $D_S$
        $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
    until convergence
    return $\hat{\Theta}$
end procedure
```

Photo2: bootstrapping based stochastic gradient descent algorithm

We will introduce a popular stochastic gradient descent approach. From the equation below, an update is performed for each triple (u, i, j) ∊ D_s.

stochastic gradient descent mathematic formula

The order for the approach over the training pairs is crucial. Traversing over the data item-wise or user-wise results in poor convergence while the same user-item pairs would have many updates. The stochastic gradient descent algorithm is introduced to choose the triples randomly (uniformly distributed). Such an approach can reduce the chances to choose the same user-item combination in consecutive update steps. Bootstrap sampling approach with replacement is able to stop the random choice of the user-item pair. The number of a single step is selected linearly in our evaluation based on the number of observed positive feedback S.

Learning models with BPR

Both two diverse model classes of matrix factorization and learned k-nearest-neighbor would model the hidden preferences of a user on an item. First, we decompose the estimator ˆx_uij and define it as:

> *xˆuij := ˆxui − xˆuj*

Next, the standard collaborative filtering model is applied to predict ˆx_ul. Such an approach is better than the original one since it classifies the difference of two predictions ˆx_ui – xˆ_uj rather than regress a single predictor ˆx_ul to a single number.

Matrix Factorization

The estimate of the matrix X : U × I is used to coping with the problem of predicting ˆx_ui. The target matrix is produced with a matrix product of two low-rank matrices W : |U| × k and H : |I| × k from the matrix factorization.

> *Xˆ := W* H^t*

From the equation below, there are parameters specified.

K: the dimensionality/rank of the approximation
W_u: a feature vector in W, indicating a user u and similarly each row hi of H describes an item i.

prediction formula

Θ = (W, H): the model parameters for matrix factorization. The parameters are latent variables to model unobserved user and item pairs. Singular value decomposition (SVD) is achieved to approximate Xˆ to X for the least-square. Usually, the SVD approach would overfit the data. Then, there are other recommended methods like regularized least-square optimization, non-negative factorization, maximum margin factorization.

For the ranking task, we use LearnBPR to estimate whether a user prefers one item over another. Through the LearnBPR approach, we need to know the gradient of ˆx_uij with respect to every model parameter θ. Below, it shows the derivatives from the matrix factorization model.

For the task of ranking, i.e. estimating whether a user prefers one item over another, a better approach is to optimize against the BPR-Opt criterion. This can be achieved by using our proposed algorithm LearnBPR. As stated before for optimizing with LearnBPR, only the gradient of ˆx_uij with respect to every model parameter θ has to be known. For the matrix factorization model the derivatives are:

the derivatives of matrix factorization model

Besides, there are three other regularization terms added.

λW: user features W

λH+: positive updates on hif for the item features H

λH−: negative updates on hjf for the item features H

Adaptive K-nearest-neighbor

There's another popular method in collaborative filtering called K-nearest-neighbor method, derived from a similarity measure between either item (item-based) or users (user-based). For the K-nearest-neighbor, we generate the prediction of the recommend item i from the user u based on the past similarity of i to all other items the user has seen in the past of history data — i.e. I + u. The k-nearest neighbors are produced from the k most similar items of I + u. The model of item-based K-nearest-neighbor is listed below for the item prediction. C is the symmetric item-correlation/ item-similarity matrix.

Item prediction of K-nearest-neighbor

Model parameters in kNN are Θ = C. C is determined by applying a heuristic similarity measure, e.g. cosine vector similarity:i.

cosine vector similarity of K-nearest-neighbor

Similarity measure C would be updated when the model is learning. Parameter C can be applied directly. On the other hand, when the parameter

C is too large, the model produces C from the factorization HHt with H : I × k. Later on, we choose to adapt to C without the factorization. To optimize the KNN model for ranking, the LearnBPR algorithm is used to update the gradient of ˆx_uij with respect to the model parameters C.

KNN gradient of LearnBPR algorithm mathematic formula

There are two regularization constants, λ+ for updates on cil, and λ− for updates on cjl.:

Hands-on experience of python code

Data Description:

There are ratings, users, and movies dataset extracted from grouplens website. These files contain around 1 million anonymous ratings of approximately 4,000 movies from 6,000 MovieLens users. The rating data includes UserID, MovieID, and ratings. The user file contains demographic information from users like gender, age, occupation, and zip-code. The movie dataset has basic information like MovieID, title, and genres.

Bayesian Personalized Ranking from Implicit Feedback

For the modeling approach, the personalized ranking system, maximum posterior estimator derived from a Bayesian analysis, is implemented in Pytorch model. Besides, the optimization of the model is done through a generic learning algorithm based on stochastic gradient descent with bootstrap sampling.

Data Preprocessing

From the users and rating dataset, we create the unique userid mapping with the movies’ ratings given by the unique users. Also, the unique item mapping from the rating dataset. There are two approaches applied when creating train_user_matrix and test_user_matrix. One is randomly selected of the users, and another is considered with the time order factor. With the train and test setlist, we take them to create the user and item pair for the matrix input.

For the model input data, we use the DataLoader object from Pytorch library. DataLoader iterates through the dataset from the combination of a dataset and a sampler. DataLoader enables data to create in batches via arguments `batch_size.` The parameter determines how many samples per batch to load. There's another `num_worker` parameter in the DataLoader class that assigns how many subprocesses to use for data loading.

Modelling

First, we create a user matrix (u) with user index, the item index (i) with all the items preferred by users, and item index (j) not preferred by users. Then, the matrix x_ui is generated with the multiplication of u and i, and sums up with each row. And, the matrix x_uj is generated with the multiplication of u and j, and sums up with each row. Matrix x_uij: matrix x_ui — matrix x_uj. The probability is rendered by the sigmoid function of the matrix x_uij and sums up the value. Regularization is added with the weight_decay parameter multiplied by the normalization of the user and item matrix and sums up the square function. The mathematical formula is produced below.

[illegible]

Later on, the prediction outputs the highest probability of the recommended items and is stored in the list.

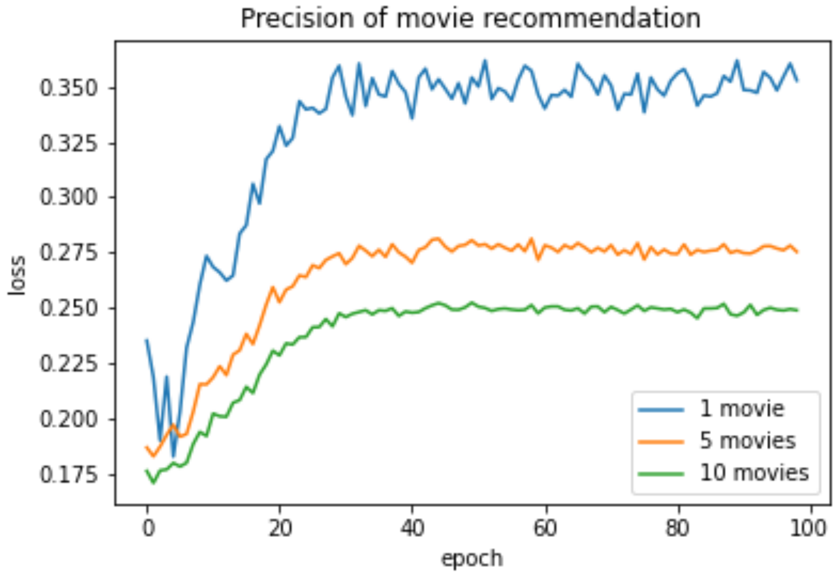
DataLoader and lost function

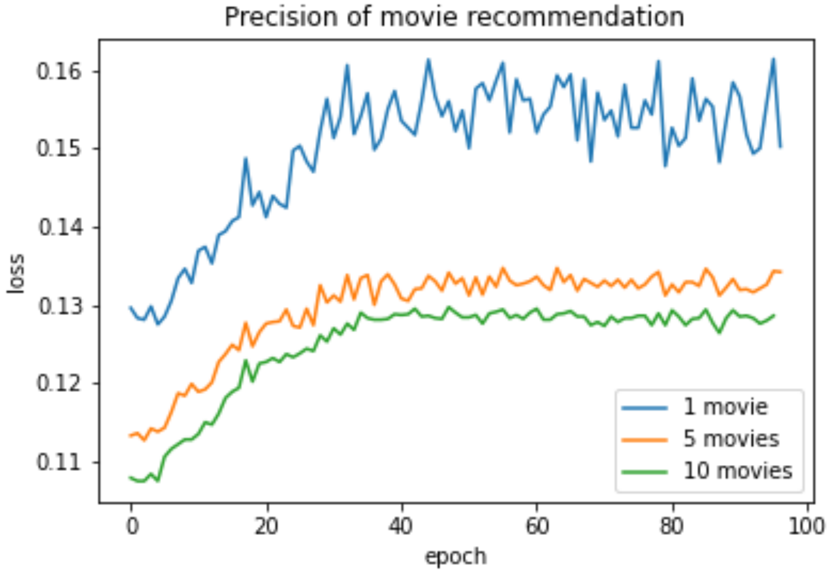
We generate a DataLoader Class with a batch size of 4096 and assigns 16 numbers of workers. Since the dataset is quite large, we assign 16 subprocesses to parallel the data processing period. Adam optimizer is set with a learning rate of 0.0001. There’s stochastic gradient descent for the backward propagation to optimize the model. For each batch size, we record the loss, precision rate, and recall.

Result

There are two different datasets for model input. One is the random data with userid and itemid, and the other is the time-order dataset.

The precision is produced below. The left-hand side is random data. From the left plot, the 1 item recommendation produces a better result than the 5 and 10 movies recommendation. From the right plot, the time order input dataset has a slightly worse result than the random data since the model is hard to catch the time sequence factor into the model.





Future work :

- Cope with the sparse matrix. From the data preprocessing method, filter the movies, not from the user's favorite category.
- Add more features of users matrix-like users' friend relationships, membership, click rate, etc.
- Incorporate the time factor into the model structure

In Conclusion

- Bayesian Personalized Ranking uses the likelihood function for p(i >u j|

Θ) and the prior probability for the model parameter $p(\Theta)$. The Bayesian approach is to produce the rankings for all items $i \in I$ to maximize the following posterior probability

- For the ranking task, we use LearnBPR to estimate whether a user prefers one item over another from matrix factorization. The optimized of the model would be achieved by calculating the gradient of ˆx_uij with respect to every model parameter θ.
- K-nearest-neighbor method, derived from a similarity measure between either item (item-based) or users (user-based), generates the prediction of the recommend item i from the user u based on the past similarity of i to all other items the user has seen in the past of history data.

Reference

- BPR: Bayesian Personalized Ranking from Implicit Feedback https://arxiv.org/pdf/1205.2618.pdf