1 BPR - Bayesian Personalised Ranking

1.1

The **objective** of the BPR model with item bias and l_2 regularization is:

$$\max_{\boldsymbol{\theta}} \sum_{(m,i,j)\in D} \log \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i - \boldsymbol{u}_m^T \boldsymbol{v}_j + b_i - b_j) - \frac{\alpha_{u_m}}{2} \sum_{m=1}^{M} \|\boldsymbol{u}_m\|^2 - \frac{\alpha_{v_n}}{2} \sum_{n=1}^{N} \|\boldsymbol{v}_n\|^2 - \frac{\alpha_{b_n}}{2} \sum_{n=1}^{N} \|\boldsymbol{b}_n\|^2$$

From here and below, we will refer the objective as $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the model's parameters group:

$$\boldsymbol{\theta} = \Big[\{\boldsymbol{u}_m\}_{m=1}^{M}, \{\boldsymbol{v}_n\}_{n=1}^{N}, \{b_n\}_{n=1}^{N} \Big]$$

This objective function is the log-Posterior function which is built from two parts:

1. **The log-likelihood:** $\log \Big(Pr(D|\boldsymbol{\theta}) \Big)$ - the first sum is the probability to get the data with the given model parameters $\boldsymbol{\theta}$. With an assumption that the records in the data are independent, this probability is just the multiplication of the probabilities to receive each observation record. After using the log function, that does not change the extremum points of a function, the multiplication changes to sum.
2. **The log-priors:** $\log \Big(Pr(\boldsymbol{\theta}) \Big)$ - the other parts of the function are the initial assumption about the distribution of the model's parameters when nothing is known. We assume the parameters are distributed normally (multivariate one for the vector parameters) with a mean of 0 the α_k as the variance parameter for the k group of the parameters. We get these nice sum expressions from using the normal density function and the log function while omitting constants respect the models' parameters.

1.2

We used negative examples for training the BPR model. We have employed two schemes, each matches to other method that the test data was created by. Both schemes are based on the users negative items - for each user, we found her negative items by subtracting the items she consumed (positive items) from the items catalog. Afterwards, we used the following schemes:

1. **Uniform Sampling:**
   For each positive item in the training data and each epoch, we have sampled (with replacement) a negative item, using uniform distribution over the entire negative examples of that user. That is, for each user we have sampled `number_of_positive*number_of_epochs` negative items.
2. **Popularity Sampling:**
   Firstly, for each user, we have calculated the frequencies of her negative items. Secondly, for each positive item in the training data and each epoch, we have sampled (with replacement) a negative item, using the calculated frequencies as probabilities. It's worth mentioning that we have used different popularity distributions over the negative items for each user, according to her positive items. As the uniform sampling above, we have sampled `number_of_positive*number_of_epochs` negative items for each user.

1.3

1.3.i

The derivative of the objective by u_m is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{u}_m} = \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](\boldsymbol{v}_i - \boldsymbol{v}_j) - \alpha_{u_m} u_m$$

Therefore, the update step is:

$$\boxed{u_m^{new} \leftarrow u_m^{old} + \eta \left\{ \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](v_i - v_j) - \alpha_{u_m} u_m \right\}}$$

Where η is the learning rate hyper parameter.

1.3.ii

The derivative of the objective by v_i is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial v_i} = \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](u_m) - \alpha_{v_n} v_i$$

Therefore, the update step is:

$$\boxed{v_i^{new} \leftarrow v_i^{old} + \eta \left\{ \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](u_m) - \alpha_{v_n} v_i \right\}}$$

Where η is the learning rate hyper parameter.

1.3.iii

The derivative of the objective by v_j is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial v_j} = \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](-u_m) - \alpha_{v_n} v_j$$

Therefore, the update step is:

$$\boxed{v_j^{new} \leftarrow v_j^{old} + \eta \left\{ \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right](-u_m) - \alpha_{v_n} v_j \right\}}$$

Where η is the learning rate hyper parameter.

1.3.iv

The derivative of the objective by b_i is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial b_i} = \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right] - \alpha_{b_n} b_i$$

Therefore, the update step is:

$$\boxed{b_i^{new} \leftarrow b_i^{old} + \eta \left\{ \left[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\right] - \alpha_{b_n} b_i \right\}}$$

Where η is the learning rate hyper parameter.

1.3.v

The derivative of the objective by b_j is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial b_j} = \big[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\big](-1) - \alpha_{b_n} b_j$$

Therefore, the update step is:

$$\boxed{b_j^{new} \leftarrow b_j^{old} + \eta \Big\{ -[1 - \sigma(\boldsymbol{u}_m^T \boldsymbol{v}_i + b_i - \boldsymbol{u}_m^T \boldsymbol{v}_j - b_j)\big] - \alpha_{b_n} b_j \Big\}}$$

Where η is the learning rate hyper parameter.

1.4

Below is a pseudo code for the SGD algorithm for the BPR model. Notice that the latent vectors dimension d, the variance of the normal distribution σ^2, the coefficients α_k and the learning rate η are part of the model hyper parameters set. As such, their values are set as part of hyper-parameters tuning stage. It worth mention that in our code we first did some epoch with only the bias model of the BPR.

Algorithm 1 Matrix Factorization BPR Model using Stochastic Gradient Descent Algorithm

```
    Input: A training set $(m,i,j) \in D^t$, a validation set $(m,i,j) \in D^v$, an hyper-parameters set $H$
    Output: Model parameters: $\forall m,n : \boldsymbol{u}_m, \boldsymbol{v}_n \in \mathbb{R}^d$, $b_n \in \mathbb{R}$
 1: Initialize Model Parameters:
    $\forall m : \boldsymbol{u}_m \sim \mathcal{N}(0,\, \sigma^2 I)$
    $\forall n : \boldsymbol{v}_n \sim \mathcal{N}(0,\, \sigma^2 I)$; $b_n \sim \mathcal{N}(0,\, \sigma^2)$
 2: while model not converge do
 3:     for each $(m,i,j) \in D_t$ do
 4:         $e_{mij} \leftarrow 1 - \sigma\big(\boldsymbol{u}_m^T(\boldsymbol{v}_i - \boldsymbol{v}_j) + (b_i - b_j)\big)$
 5:         $b_i \leftarrow b_i + \eta\big[e_{mij} - \alpha_{b_n} b_i\big]$
 6:         $b_j \leftarrow b_j + \eta\big[-e_{mij} - \alpha_{b_n} b_j\big]$
 7:         $\boldsymbol{u}_m \leftarrow \boldsymbol{u}_m + \eta\big[e_{mij}(\boldsymbol{v}_i - \boldsymbol{v}_j) - \alpha_{u_m} \boldsymbol{u}_m\big]$
 8:         $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \eta\big[e_{mij} \boldsymbol{u}_m - \alpha_{v_n} \boldsymbol{v}_i\big]$
 9:         $\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j + \eta\big[e_{mij}(-\boldsymbol{u}_m) - \alpha_{v_n} \boldsymbol{v}_j\big]$
10:     $\eta \leftarrow 0.9\eta$
```

For details on how we have judged whether the model has converged, please follow Section 1.6.

1.5

The hyper-parameters for BPR algorithm are:

1) **Learning rate:** η - determines the step size to be taken in the direction of the parameters gradient for each update step, in the way towards a maximum of the log-Posterior function.

2) **The vector dimension:** d - the dimension of the latent vectors $\boldsymbol{u}_m$ and $\boldsymbol{v}_n$, which represent each of the users and items in the data.

3) **Regularization coefficients:** α_{u_m}, α_{v_n}, α_{b_n} - three regularization coefficients for each group of the model parameters. These coefficients help to prevent overfitting of model to the train set.

4) **Standard deviation of parameters initialization:** σ - supposed to be the inverse of the alpha coefficients. But when trying to work with them it took too much time to converge and resulted in inferior model so we used separately hyper-parameter instead.

As the search method for finding the best hyper-parameters set, we have used the **Randomized Grid Search.** In this method we **control the number of hyper-parameters sets we examine,** and with only 100 samples we can gain one of the **top 3% sets of hyper-parameters with 95% confidence[1],** and thus reducing dramatically the search space.

The best hyper-parameter set we have found was the same for both of the models - the Popularity Model and the Uniform Model is:

η	d	α_{u_m}	α_{v_n}	α_{b_n}	σ
0.05	60	0.01	0.01	0.1	0.1

1.6

We do need validation data for this learning task. With this data, we examined whether the model has converged yet or not - one criterion out of our three convergence criteria (more to follow). We wanted to keep the validation data as similar as possible to the test data we were provided. So, we created two validation data sets, one for each sampling scheme of the test sets. Each validation set was being in use for the convergence check of the matching model. Moreover, in each validation set, there was one record for each user, as in the test sets. This record was a triplet that included a user index m, a hidden positive item index i (hidden from the training data, so the model never trained on it), and a negative item index j, that was sampled with the correspondent sampling scheme - according to the items popularity or uniformly.

We used three convergence criteria that were checked at the end of each epoch. The criteria are:

1) **A threshold number allowing the objective function to increase on the validation data:**
We calculated the objective function value of the current model (with the current parameters) on the validation data and compared it to the objective function value from the previous epoch. We allowed this value to increase three times, before stopping the parameters' updating steps (after the third time of increment).

2) **A threshold value for the change of the objective function on the training data:**
If there wasn't sufficient improvement (increasing of at least ϵ) of the objective function on the training data in adjacent epochs, the update steps would stop. Eventually, we set aside this criterion and never used it.

3) **Reaching max epoch threshold number:**
After conducting 50 epochs of the SGD algorithm in which no other criterion was met, we have stopped the parameters' update steps after 50 epochs. That is, after going through the entire training data 50 times.

[1] `https://towardsdatascience.com/hyper-parameter-tuning-with-randomised-grid-search-54f865d27926`

We determined that **the model has converged if at least one criterion is met.** It is worth mention that the returned model is the one that got the lowest value of the objective function on the validation set before the early stop, which might not be the last one.

1.7

After **finding the best hyper-parameters set in the tuning step** (the one that has balanced between good results for both the MPR and error rate, see the next section for more details), we used it for training the final models - the Popular Model and the Uniform Model. These final models, which were used for the prediction of each test set, were **trained over the whole training set, without validation data held out.** Thus, also including the hidden positive item that was left out for each user. Moreover, as part of the training data, we sampled negative items, with the corresponding sampling scheme for each test set. The third criterion from Section 1.6 was used as for the convergence check, with the maximum number of epochs defined as the number of epochs required when the validation data was available (found in the tuning step as well). Finally, we used each trained model to predict the positive item of a user from a pair of items in the test set.

1.8

Our **main work items** in the search for the final model were:

1) **Running the Bias model** for 2 epochs and return the best biases found on validation data. We don't continue with the epochs until the bias model converges since these parameters keep to optimize with the full model updates anyway.
2) **Running the Full model** until convergence, with the biases found from the Bias model.
3) **Tuning of the hyper-parameters** using **Randomized Grid Search** as the search method, while trying to optimize the MPR on validation set. For more details about this method, follow Section 1.5. We randomly drew one set of hyper-parameters from the full grid to compute the MPR at every iteration. In addition, in order to get the optimal hyper-parameters, we examined the error rate as well, and chose the hyper-parameters set that got the good values for these two metrics. See figures 1 and 2.
4) Lastly, we **trained the final model** on the train and validation sets with the **best hyper parameters values set** found on the previous part.

1.9

Results on validation set: One can see that the Uniform model performs better on all the metric. It is not surprising since it learns easier task - in most cases, the more popular item of the pair is the positive one. This effect is weaken in the other task using the popularity distribution.

Evaluation	MPR	Hit Rate@$K = 1$	Hit Rate@$K = 10$	Hit Rate@$K = 50$
Uniform Model	0.082	0.0079	0.074	0.27
Popularity Model	0.12	0.0071	0.065	0.24

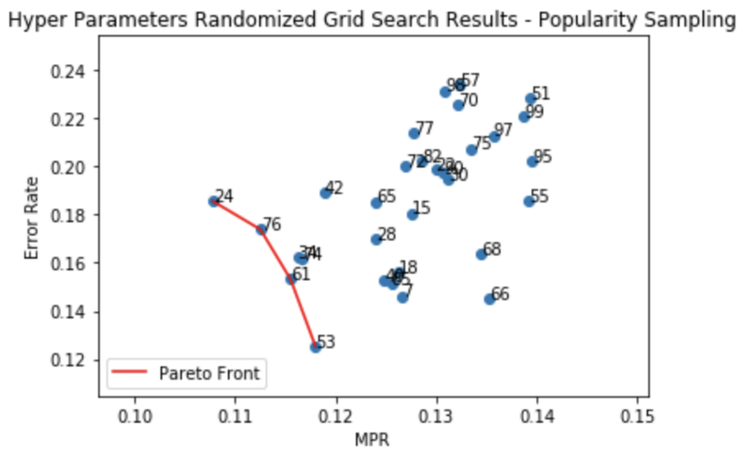

Figure 1: Hyper parameters randomized grid search results of the popularity sampling model. The red line demonstrate the hyper-parameters sets that there is no other set that dominants on its value of either the MPR or error rate. We chose HP set number 53 for the final model.

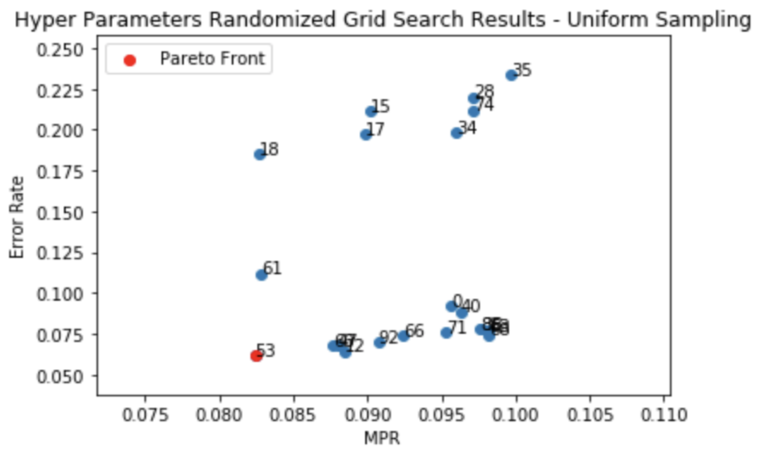

Figure 2: Hyper parameters randomized grid search results of the uniform sampling model. The red dot demonstrate the hyper-parameters sets that there is no other set that dominants on its value of either the MPR or error rate. We chose HP set number 53 for the final model, which get both the best error rate and MPR values.