# SGD

November 2019

## 1 Objective

minimize

$$L = \sum_{D}(r_{m,n} - \mathrm{U_m}^{\mathrm{T}}\mathrm{V_n} - b_m - b_n - \mu)^2 + \lambda_u \sum_{m} \|U_m\| + \lambda_v \sum_{n} \|V_n\| + \lambda_{bu} \sum_{m} {b_m}^2 + \lambda_{bv} \sum_{n} {b_n}^2$$

where $r_{m,n}$ is the explicit feedback from user $m$ on item $n$, $D$ is the data set, each sample is the triplet ($m$, $n$, $r_{m,n}$). $\mathrm{U_m}$ is user $m$ latent vector and $\mathrm{V_n}$ is item $n$ latent vector. $b_m$ and $b_n$ are the biases of user $m$ and item $n$ respectively, and $\mu$ is the mean of $r_{m,n}$ on the entire dataset. $\lambda_u$, $\lambda_v$, $\lambda_{bm}$, $\lambda_{bn}$ are regularization hyper-parameters.

## 2 Update Steps

SGD

$$\mathrm{e_{m,n}} = (r_{m_n} - \mathrm{U_m}^{\mathrm{T}}\mathrm{V_n} - b_m - b_n - \mu)$$

Update steps are:

$$\mathrm{U_m} = \mathrm{U_m} + \alpha(e_{m,n}\mathrm{V_n} - \lambda \mathrm{U_m})$$

$$\mathrm{V_n} = \mathrm{V_n} + \alpha(e_{m,n}\mathrm{U_m} - \lambda \mathrm{V_n})$$

$$\mathrm{b_m} = \mathrm{b_m} + \alpha(e_{m,n} - \lambda \mathrm{b_m})$$

$$\mathrm{b_n} = \mathrm{b_n} + \alpha(e_{m,n} - \lambda \mathrm{b_n})$$

# 3 Pseudo Code

**Algorithm 1:** SGD pseudo code

**Result:** Write here the result

init $\mathrm{U_m}$, $\mathrm{V_n}$, $\mathrm{b_m}$, $\mathrm{b_n} \sim N(0, 0.1)$ ;

**while** *true* **do**

  print (iteration, train error, validation error);

  **for** $\forall m, n, r_{m,n} \in D$ **do**

    Update

$$\mathrm{e_{m,n}} = r_{m_n} - \mathrm{U_m}^{\mathrm{T}}\mathrm{V_n} - b_m - b_n - \mu$$

    Update

$$\mathrm{U_m} = \mathrm{U_m} + \alpha(e_{m,n}\mathrm{V_n} - \lambda\mathrm{U_m})$$

    Update

$$\mathrm{V_n} = \mathrm{V_n} + \alpha(e_{m,n}\mathrm{U_m} - \lambda\mathrm{V_n})$$

    Update

$$\mathrm{b_m} = \mathrm{b_m} + \alpha(e_{m,n} - \lambda\mathrm{b_m})$$

    Update

$$\mathrm{b_n} = \mathrm{b_n} + \alpha(e_{m,n} - \lambda\mathrm{b_n})$$

  **end**

  $a$ = 0.9$a$

  **if** *validation error did not improve for the last $k$ iterations* **then**

    break;

  **end**

**end**

# 4 Hyper-parameters

The hyper-parameters we need to tune are: $a$ which is the learning rate, $\lambda_u$ (user regularization), $\lambda_v$ (item regularization), $\lambda_{bm}$ (user bias regularization), $\lambda_{bn}$(item bias regularization) and also $d$ the embedding dimension or the latent dimension of the user and item matrices.

# 5 Validation set

The validation set is given to us (we assume it contains ($m$, $n$, $r_{m,n}$) triplets that were recorded AFTER the training set). We will decide on $RMSE$ (We could have chosen $R^2$ as well) as our key evaluation metric and init $prevRMSE_v = 1000000$ and $noimprovecounter = 0$. Each iteration on the training set (after all user,item and biases vectors were updated), we will generate predictions on the entire validation set. We will decrease the learning rate $a$ to be $a = 0.9a$, and evaluate the $RMSE$: if it is lower than $prevRMSE_v$ we will update $prevRMSE_v = RMSE$ and set continue to another iteration, otherwise we set $noimprovecounter+ = 1$. Once $noimprovecounter$ reaches a fixed value (say 10) we can stop the iterations and figure out the the model is not improving (and we reached to convergence). In addition, we could print the $R^2$ for the training set to check if our model is learning and to see we are not over-fitting (and if so, tune the regularization hyper-parameters accordingly). Providing further metrics such as $R^2$ and MAE will also help us evaluate the model better.

# 6 Train last/best model

First we trained the model only with biased parameters to get $b_m$ and $b_n$. Then, we trained the entire model again with different hyper-parameters that were selected randomly from a uniform distribution. After running a few iterations we took the hyper-parameters that gave the best results and optimized them again with 'Nelder Mead' method. We took the best result from 'Nelder Mead', and trained again on different embedding dimensions($d$). Finally, we chose the best $d$ with the best hyper - parameters that gave the best results.

# 7 Implementation

The implementation is provided in the attached .zip file. Please follow the instructions for running the code. the main modules in the code are:

1. "data-preprocess" which contains the DataSet() class that holds the training,validation and test data sets along with the ratings in matrix form and other utilities.
2. "resources" which contains the original data sets and configurations.
3. "SGD" folder which contains the "sgd-utils" module. This module contains the train, evaluate, predict and tune methods.

Main work items:

1. read the training and validation sets and mapping them to continuous indexes
2. deal with cold users or items in the validation set
3. training and evaluating the model (already explained in previous sections)
4. giving predictions for the test set include cold users/items.

# 8 Results

The best results on the validation set were achieved with the following hyper-parameters $a = 0.2449$ , $\lambda_u = 0.1045$, $\lambda_v = 0.0389$, $\lambda_{bm} = 0.1183$, $\lambda_{bn} = 0.1058$ and $d = 15$. The results on validation set showed $R^2 = 0.364$ , MAE=$0.695$, RMSE=$0.889$