Data Set Description

This data set is taken from Northeastern University CS6220 Data Mining Techniques course material.

- rating.csv

  Contains 10,000 entry of ratings. Each entry consists of “user_id”,”item_id”,”rating_score”

  “user_id” is the ID of a user; ”item_id” is the ID of a movie; “rating_score” is the rating from the user with ID of “user_id” to the move with ID of “item_id”;

Data Partition

The model can be trained by using K-fold cross validation, which didn’t require divide the rating.csv to get a test set.

Alternatively, we can create a test set by dividing “rating .csv” by using the function **Utility.generateTestAndTrainData**(**String** inputPath, **double** proportionTrainset,**String** trainSetPath,**String** testSetPath)

Parameter selection

we can use the following code to select a best dimension of latent vector

```
String outputPathSelectLatentDimension=inputPath+"_LatentDimSelection.txt";
selectLatentDimension(1, 50,inputPath,outputPathSelectLatentDimension);
```

Baseline

we can get the baseline RMSE and MAE by using the following function

getBaselineRMSE(List<Rating> testSet)

getBaselineMAE(List<Rating> testSet)

Pipline

We can train a model, make prediction and get RMSE and MAE by using the following function

in such oreder,

1. **public void training**(List<Rating> trainSet)
2. call **public double predict**(**int** user_id,**int** item_id) for every data entry in the test set
3. get evaluation results by following functions seperately
   private double getRMSE(List<Rating> testSet)
   private double getMAE(List<Rating> testSet)

Training

we train the model by the following order

1. initialization
   call the function **private void initlization**(List<Rating> trainSet)
2. using the stochastic gradient descend to optimize the cost function
3. check for convergence
   3.1 set the convergeCheckWindowSize, which determines how many data from the first data entry in the training set we want to use to calculate the cost function
   3.2 check if converges every time we iterate for “convergeCheckWindowSize” number of training examples

```
// check if converge
convergeCheckProgress++;
if(convergeCheckProgress>=convergeCheckWindowSize){

    cost_new=getCost(trainSet);
    delta_cost=cost_new-cost_old;
    if(delta_cost<convergeThreshold){
        converge=true;
        break;
    }else{
        cost_old=cost_new;
        convergeCheckProgress=0;
    }
}
```