DAYANANDA SAGAR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY
KUDLU GATE
BANGALORE – 560068

A MINI PROJECT REPORT

ON

"Get Projects"

SUBMITTED TO THE 7th SEMESTER MACHINE LEARNING LABORATORY
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by

Jatin Yadav - ENG17CS0098
Mayarmui Ruivah - DSU15CS0042
Asfan Ulla – ENG16CS0024

Under the supervision of
Prof. Reeja S R,

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that Mr./Ms. Jatin Yadav, Mayarmui Ruivah and Asfan Ulla bearing USN Eng17cs0098, Dsu15cs0042 & Eng16cs0024 has satisfactorily completed his/her Mini Project as prescribed by the University for the 7th semester B.Tech. programme in Computer Science & Engineering during the year 2020-21 at the School of Engineering, Dayananda Sagar University., Bangalore.

Date: ____________

Signature of the faculty in-charge

Max Marks	Marks Obtained

Signature of Chairman

Department of Computer Science & Engineering

DECLARATION

We hereby declare that the work presented in this mini project entitled
“Get Projects “, has been carried out by us and it has not been submitted for the award of any degree, diploma or the mini project of any other college or university.

Jatin Yadav – ENG17CS0098
Mayarmui Ruivah – DSU15CS0042
Asfan Ulla – ENG16CS0024

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We are especially thankful to our **Chairman, Dr. Sanjay Chitnis**, for providing necessary departmental facilities, moral support and encouragement.

We are very much thankful to **Prof. Reeja SR**, for providing help and suggestions in completion of this mini project successfully.

We have received a great deal of guidance and co-operation from our friends and we wish to thank all that have directly or indirectly helped us in the successful completion of this project work.

Jatin Yadav - ENG17CS0098
Mayarmui Ruivah – DSU15CS0042
Asfan Ulla – ENG16CS0024

TABLE OF CONTENTS

Abstract

Recommender systems usually make personalized recommendation with user-item interaction ratings, implicit feedback and auxiliary information. Matrix factorization is the basic idea to predict a personalized ranking over a set of items for an individual user with the similarities among users and items. In this paper, we propose a novel matrix factorization model. Firstly, we construct a user-item matrix with explicit ratings and non-preference implicit feedback. With this matrix as the input, we present a deep structure learning architecture to learn a common low dimensional space for the representations of users and items. Secondly, we design a new loss function based on binary cross entropy, in which we consider both explicit ratings and implicit feedback for a better optimization. The experimental results show the effectiveness of both our proposed model and the loss function. On several benchmark datasets, our model outperformed other state-of-the-art methods. We also conduct extensive experiments to evaluate the performance within different experimental settings

INTRODUCTION

Machine learning is a branch of Artificial Intelligence which is used to analyse the data more smartly. It automates the process using certain algorithms to minimize human intervention in the process.

In this project we are going to focus on solving the problem of recommending projects to the user. It might be difficult for a user to find the project they need or would like in a large library of free open source projects we are going to analyse the previous project ratings of the user and recommend the projects.

The aim of this project is to recommend projects using machine learning recommendation algorithm, Matrix Factorization. We will use past user project ratings to try to improve our recommendation accuracy.

The primary goal of the project is to implement the project recommendation. Here we are going to use machine learning recommendation algorithm, Matrix Factorization. and recommend projects.

Problem Statement

In this machine learning project, we are going to recommend projects to users using ML.NET. This project will help the users to choose the projects they like from the vast project library and act accordingly.

Here we are going to use matrix factorization to recommend projects. In a nutshell matrix factorization is a class of collaborative filtering algorithms used in recommender systems. The Score, or the predicted rating is used to determine whether we want to recommend a project to the user. The higher the Score, the higher the likelihood of a user liking a particular Project.

Objective

There are several ways to approach recommendation problems, such as recommending a list of related products, but in this case, we will predict what rating (1-5) a user will give to a particular project and recommend that project if it's higher than a defined threshold (the higher the rating, the higher the likelihood of a user liking a particular project).

Matrix Factorization is a common approach to recommendation when we have data on how users have rated products in the past, which is the case for the datasets we are using in this case.

The Matrix Factorization algorithm uses a method called "collaborative filtering", which assumes that if User 1 has the same opinion as User 2 on a certain issue, then User 1 is more likely to feel the same way as User 2 about a different issue. For instance, if User 1 and User 2 rate projects similarly, then User 2 is more likely to use the project User 1 has used and rated highly.

The recommendation ratings data is split into Train and Test datasets. The Train data is used to fit our model. The Test data is used to make predictions with our trained model and evaluate model performance.

Methodology

Data structure used

- Arrays

Flow chart

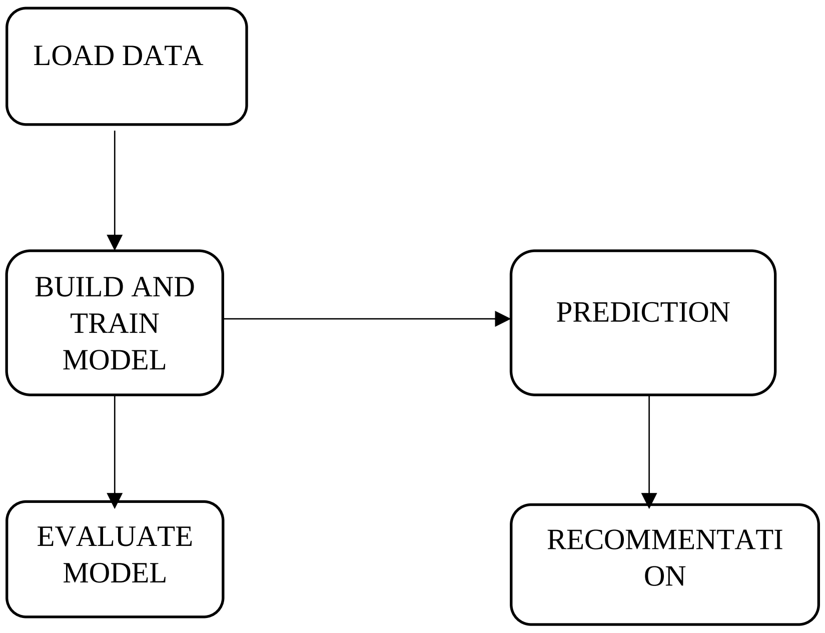


Fig.1 Program Flow Chart

Pseudo code

Define and Initialize MLContext –

```
MLContext mlContext = new MLContext();
```

Load Train and Test Data into IDataView Object –

```
IDataView trainingDataView =
mlContext.Data.LoadFromTextFile<ProjectRating>(trainingDataPath, hasHeader: true,
separatorChar: ',');
IDataView testDataView = mlContext.Data.LoadFromTextFile<ProjectRating>(testDataPath,
hasHeader: true, separatorChar: ',');
```

Build and Train data and load model into ITransformer object –

```
var trainerEstimator =
estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
ITransformer model = trainerEstimator.Fit(trainingDataView);
```

Evaluate the model using test data –

```
var prediction = model.Transform(testDataView);
var metrics = mlContext.Regression.Evaluate(prediction, labelColumnName: "Label",
scoreColumnName: "Score");
```

Make a prediction and recommend the project if the prediction score is greater than 3.5 –

```
var predictionEngine = mlContext.Model.CreatePredictionEngine<ProjectRating,
ProjectRatingPrediction>(model);


var projectRatingPrediction = predictionEngine.Predict(testInput);

            if (Math.Round(projectRatingPrediction.Score, 1) > 3.5)
            {
                Console.WriteLine("Project " + testInput.projectId + " is recommended
for user " + testInput.userId);
            }
            else
            {
                Console.WriteLine("Project " + testInput.projectId + " is not
recommended for user " + testInput.userId);
            }
```

Code Snippets & user defined functions

Main function –

```
        static void Main(string[] args)
        {
            MLContext mlContext = new MLContext();
            (IDataView trainingDataView, IDataView testDataView) = 
LoadData(mlContext);
            ITransformer model = BuildAndTrainModel(mlContext, trainingDataView);
            EvaluateModel(mlContext, testDataView, model);
            ModelPrediction(mlContext, model);
        }
```

Load Data Function –

```
       public static (IDataView training, IDataView test) LoadData(MLContext 
mlContext)
        {
            var trainingDataPath = Path.Combine(Environment.CurrentDirectory, "Data", 
"recommendation-ratings-train.csv");
            var testDataPath = Path.Combine(Environment.CurrentDirectory, "Data", 
"recommendation-ratings-test.csv");

            IDataView trainingDataView = 
mlContext.Data.LoadFromTextFile<ProjectRating>(trainingDataPath, hasHeader: true, 
separatorChar: ',');
            IDataView testDataView = 
mlContext.Data.LoadFromTextFile<ProjectRating>(testDataPath, hasHeader: true, 
separatorChar: ',');

            return (trainingDataView, testDataView);
        }
```

Function to build and Train the model –

```
public static ITransformer BuildAndTrainModel(MLContext mlContext, IDataView
trainingDataView)
        {
            IEstimator<ITransformer> estimator =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "userIdEncoded",
inputColumnName: "userId")

.Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:
"projectIdEncoded", inputColumnName: "projectId"));

            var options = new MatrixFactorizationTrainer.Options
            {
                MatrixColumnIndexColumnName = "userIdEncoded",
                MatrixRowIndexColumnName = "projectIdEncoded",
                LabelColumnName = "Label",
                NumberOfIterations = 20,
                ApproximationRank = 100
            };

            var trainerEstimator =
estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

            Console.WriteLine("=============== Training the model ===============");
            ITransformer model = trainerEstimator.Fit(trainingDataView);

            return model;
        }
```

Function to evaluate the Model –

```
public static void EvaluateModel(MLContext mlContext, IDataView testDataView,
ITransformer model)
        {
            Console.WriteLine("=============== Evaluating the model ===============");
            var prediction = model.Transform(testDataView);
            var metrics = mlContext.Regression.Evaluate(prediction, labelColumnName:
"Label", scoreColumnName: "Score");
            Console.WriteLine("Root Mean Squared Error : " +
metrics.RootMeanSquaredError.ToString());
            Console.WriteLine("RSquared: " + metrics.RSquared.ToString());
        }
```

Function for prediction and recommending project to the user –

```
        public static void ModelPrediction(MLContext mlContext, ITransformer model)
        {
            Console.WriteLine("=============== Making a prediction ===============");
            var predictionEngine = 
mlContext.Model.CreatePredictionEngine<ProjectRating, ProjectRatingPrediction>(model);

            var testInput = new ProjectRating { userId = 6, projectId = 10 };
            var projectRatingPrediction = predictionEngine.Predict(testInput);

            if (Math.Round(projectRatingPrediction.Score, 1) > 3.5)
            {
                Console.WriteLine("Project " + testInput.projectId + " is recommended 
for user " + testInput.userId);
            }
            else
            {
                Console.WriteLine("Project " + testInput.projectId + " is not 
recommended for user " + testInput.userId);
            }

        }
```

Software and hardware Requirement

Hardware Requirements:

- CPU: Intel Pentium or higher (32/64 bit)
- Storage: At least 128 MB free storage space
- RAM: At least 128 MB free memory

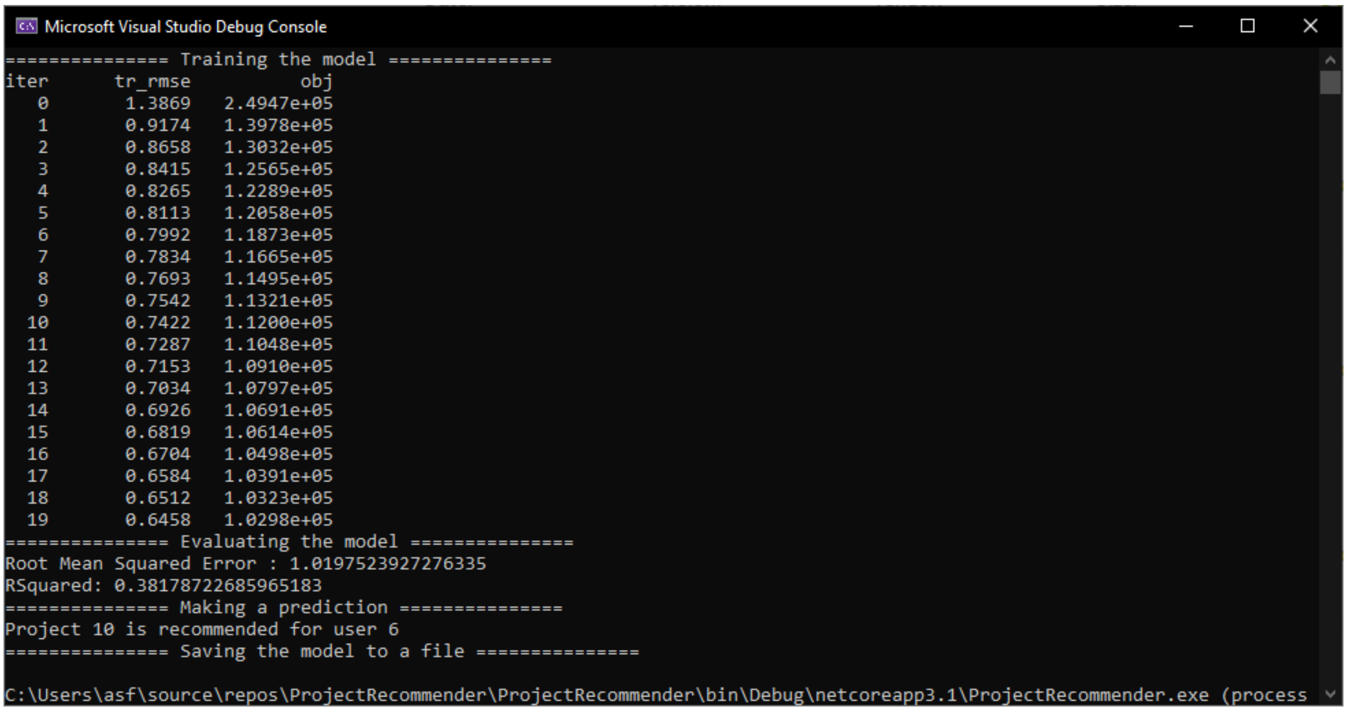
Software Requirements:

- Visual Studio.
- ML.NET
- .NET Core

LANGUAGES:

C#.

Result



```
=============== Training the model ===============
iter      tr_rmse          obj
   0       1.3869   2.4947e+05
   1       0.9174   1.3978e+05
   2       0.8658   1.3032e+05
   3       0.8415   1.2565e+05
   4       0.8265   1.2289e+05
   5       0.8113   1.2058e+05
   6       0.7992   1.1873e+05
   7       0.7834   1.1665e+05
   8       0.7693   1.1495e+05
   9       0.7542   1.1321e+05
  10       0.7422   1.1200e+05
  11       0.7287   1.1048e+05
  12       0.7153   1.0910e+05
  13       0.7034   1.0797e+05
  14       0.6926   1.0691e+05
  15       0.6819   1.0614e+05
  16       0.6704   1.0498e+05
  17       0.6584   1.0391e+05
  18       0.6512   1.0323e+05
  19       0.6458   1.0298e+05
=============== Evaluating the model ===============
Root Mean Squared Error : 1.0197523927276335
RSquared: 0.38178722685965183
=============== Making a prediction ===============
Project 10 is recommended for user 6
=============== Saving the model to a file ===============

C:\Users\asf\source\repos\ProjectRecommender\ProjectRecommender\bin\Debug\netcoreapp3.1\ProjectRecommender.exe (process
```

Fig. 2. Program Output

Conclusion

In our proposed model, we make full use of both explicit ratings and implicit feedback in two ways. The input matrix to our proposed model includes both explicit ratings and non-preference feedback. In another way, we also design a new loss function for training our models in which both explicit and implicit feedback are considered. The experiments on several benchmark datasets demonstrate the effectiveness of our proposed model.

We will verify our model with pairwise objective function. Because of the sparseness and large missing unobserved data, many works try to incorporate auxiliary extra data into recommender systems, such as social relation, review text, browsing history, and so on. This give us another interesting direction to extend our model with extra data.

References

- https://docs.microsoft.com/en-us/dotnet/machine-learning/
- https://developers.google.com/machine learning/recommendation/collaborative/matrix