데이터사이언스

Recommender

2013011800 구장회

순서

1. **data structure - recommender,BPRRecommender,SparseMatrix,DenseMatrix,Node**
2. **summary of algorithm(flow)**
3. **summary of algorithm(other crucial method)**
4. **instruction for compiling**
5. **any other specification**

data structure

[BPRRecommender]

* 목적
- BPR-MF를 통해서 all unrated user-item pair의 pre-use preference를 예측한다.
- 예측된 pre-use preference의 상위 20%를 neutral item으로 분류한다.
- 이를 netrainMatrix로 만들어서 return한다.

```
/* BPRRecommender Class
 *
 * float learnRate                : learning rate
 * float regUser,regItem          : regularization term for user,item
 * int numIterations              : # of iteration
 * int numFactors                 : # of latent factor
 * DenseMatrix userFactors        : user-factor matrix
 * DenseMatrix itemFactors        : item-factor matrix
 */
public class BPRRecommender {
    float learnRate, regUser, regItem;
    int numIterations, numFactors;
    DenseMatrix userFactors, itemFactors;
```

[Node]

* 목적
- user, item ,rating을 묶어서 저장하고, rating의 내림차순으로 정렬하기 위함

```
/* Node Class
 *
 * int user,item                  : user , item
 * double rating                  : rating
 */
public class Node implements Comparable<Node> {
    int user, item;
    double rating;
```

[recommender]

*** 목적**

- item 분류

- **observed rating으로 만들어진 trainMatrix를 interesting item**
- **BPRRecommender로부터 만들어진 netrainMatrix를 neutral item**
- **나머지 item은 uninteresting item**

- BPR concept에서 interesting > neutral > uninteresting 을 각각 학습하여 all unrated user-item pair에 대해서 post-use preference를 예측한다.

```
/* Recommender Class
 *
 * final Random r                  : Random instance
 * float learnRate                 : learning rate
 * float regUser,regItem           : regularization term for user,item
 * int numIterations               : # of iteration
 * int numFactors                  : # of latent factor
 * int numUsers,numItems           : # of user,item
 * String input                    : inputfile name
 * SparseMatrix trainMatrix        : rating matrix(observed user-item pair)
 * SparseMatrix netrainMatrix      : rating matrix(neutral user-item pair)
 * DenseMatrix userFactors         : user-factor matrix
 * DenseMatrix itemFactors         : item-factor matrix
 * DenseMatrix ratingMatrix        : rating matrix(result. i.e, all user-item pair)
 */

public class recommender {
    final Random r = new Random(System.currentTimeMillis());
    float learnRate, regUser, regItem;
    int numIterations, numFactors, numUsers, numItems;
    String input;
    SparseMatrix trainMatrix, netrainMatrix; //
    DenseMatrix userFactors, itemFactors, ratingMatrix;

```

[SparseMatrix class]

*** 목적**
- user-item pair의 rating matrix를 구성하기 위함

```java
/* SparseMatrix Class
 *
 * int numRows,numColumns          : # row , col
 * double[][] matrix               : rating matrix
 */
public class SparseMatrix {
    int numRows, numColumns;
    double[][] matrix;
```

[DenseMatrix class]

*** 목적**
- user, item factor matrix를 구성하기 위함

```java
/* DenseMatrix Class
 *
 * int numRows,numColumns          : # row , col
 * double[][] matrix               : rating matrix
 */
public class DenseMatrix {
    int numRows, numColumns;
    double[][] matrix;
```

summary of algorithms(flow)

[main method]

- 인자로 training, test filename을 받고,
- recommender class의 instance를 만들고, work() method에 training file-name을 인자로 주면서 호출한다.

```java
    /* main method
     *
     * store all parameters as training,test filename
     * and call work method after create recommender instance.
     *
     */
    public static void main(String[] args) {
        if (args.length != 2)
            System.out.println("args error");
        String base = args[0];
        new recommender().work(base);
    }
```

[main workflow]

- BuildTrainMatrix()
    - training file을 읽어서 observed user-item pair를 rating matrix로 구성한다.
- BPRRecommender class의 instance(bprrecommender)를 생성
- bprrecommender.recommend()
    - BPR-MF를 통해 all unrated user-item pair의 pre-use preference를 예측하고, 유사도 상위 20%의 pair로 구성된 netrainMatrix를 return한다.
- setup() : 변수들의 초기화
- trainModel() : 모델을 training
- buildRatingMatrix() : 예측된 all-user item pair를 output file에 쓴다.

```java

    void work(String _input) {
        input = _input;
        BuildTrainMatrix();
        BPRRecommender bprrecommender = new BPRRecommender();
        netrainMatrix = bprrecommender.recommend();
        setup();
        trainModel();
        buildRatingMatrix();
    }

```

summary of algorithms(other crucial method)

[BPRRecommender]

- recommend()
    - setup() method를 호출하고
    - trainModel() method를 호출한 뒤 리턴된 netrainMatrix를 return한다.
- setup()
    - 변수들의 초기화
    - userFactors,itemFactors matrix의 선언과 초기화

- trainModel() : BPR-MF를 그대로 사용. 추가적으로 netrainMatrix 구성
    - intuition : observed rating > unrated rating

```
362     */
363     public class BPRRecommender {
364         float learnRate, regUser, regItem;
365         int numIterations, numFactors;
366         DenseMatrix userFactors, itemFactors;
367
368         SparseMatrix recommend() {
369             setup();
370             return trainModel();
371         }
372
373         void setup() {...
384
385         SparseMatrix trainModel() {...
449     }
```

[recommender]

- **work() : main flow**
- **BuildTrainMatrix() : input file을 읽어 observed rating matrix구성**
- **setup() : 변수들의 초기화**
- **trainModel() : BPR컨셉에서 변형된 objective-function을 사용하여 학습**
    - **intuition : interesting > neutral > uninteresting item**
- **buildRatingMatrix() : user factor 와 item factor를 mult한 matrix를 생성하여 output file에 씀.**

```
26
27  public class recommender {
28      final Random r = new Random(System.currentTimeMillis());
29      float learnRate, regUser, regItem;
30      int numIterations, numFactors, numUsers, numItems;
31      String input;
32      SparseMatrix trainMatrix, netrainMatrix; //
33      DenseMatrix userFactors, itemFactors, ratingMatrix;
34
35      void work(String _input) {
44
45      void BuildTrainMatrix() {
82
83      void setup() {
94
95      void trainModel() {
164
165     void buildRatingMatrix() {
193
194     double logistic(double x) {
197
198     int uniform(int range) {
201
202     double rowMult(DenseMatrix m, int mrow, DenseMatrix n, int nrow) {
209
```

instruction for compiling

[Envoirments]

OS : Mac OS
Language : java

[Screenshot-실행전]

src — -bash — 88×53

```
koo:src KJH$ ls -al
total 19416
drwxr-xr-x  14 KJH  staff   448B Jun  1 15:41 .
drwxr-xr-x   8 KJH  staff   256B Jun  1 00:50 ..
-rw-r--r--@  1 KJH  staff   6.0K Jun  1 15:41 .DS_Store
-rw-r--r--   1 KJH  staff    14K Jun  1 15:03 recommender.java
-rwxr-xr-x@  1 KJH  staff   1.5M Mar  8  2001 u1.base
-rwxr-xr-x@  1 KJH  staff   383K Mar  8  2001 u1.test
-rwxr-xr-x@  1 KJH  staff   1.5M Mar  8  2001 u2.base
-rwxr-xr-x@  1 KJH  staff   386K Mar  8  2001 u2.test
-rwxr-xr-x@  1 KJH  staff   1.5M Mar  8  2001 u3.base
-rwxr-xr-x@  1 KJH  staff   387K Mar  8  2001 u3.test
-rwxr-xr-x@  1 KJH  staff   1.5M Mar  8  2001 u4.base
-rwxr-xr-x@  1 KJH  staff   388K Mar  8  2001 u4.test
-rwxr-xr-x@  1 KJH  staff   1.5M Mar  8  2001 u5.base
-rwxr-xr-x@  1 KJH  staff   388K Mar  8  2001 u5.test
koo:src KJH$
```

[screenshot-컴파일,실행]

컴파일 **: $ javac recommender.java**
실행 **: $ java recommender [basefile] [testfile]**

src — java recommender u1.base u1.test — 88×53

```
koo:src KJH$ javac recommender.java;
koo:src KJH$ java recommender u1.base u1.test
BPRMF:training
[iter #1][0.0010167866047530107][-7.817747414938982E-5]
[iter #2][0.0036015872620539418][-5.4682296313192654E-5]
[iter #3][0.006903412806176068][-3.1511793907586884E-4]
[iter #4][0.013748655240844418][-5.274679509867676E-4]
[iter #5][0.03378407257532971][-8.052496799899929E-4]
[iter #6][0.09254552719219171][-0.002153318941822268]
[iter #7][0.22529998010752464][-0.006532799616561083]
[iter #8][0.4320999537086485][-0.013813754313649028]
[iter #9][0.6867584242088937][-0.025126949941470956]
[iter #10][0.9540736851462366][-0.039354493524642736]
[iter #11][1.1996839198741933][-0.058131752576270415]
[iter #12][1.4255900685232847][-0.07329985159291]
```

 # any other specification

[참고]
- 모든 스크린샷은 첨부되어있음
- **recommender.java는 default package**로 **되어있음**

[아이디어]

김상욱 교수님에게 추천시스템 관련하여 졸업프로젝트를 진행하고 있습니다.
기존 추천기술에 대한 공부를 하고있고, 새로운 추천모델을 고안하고 구현하는 시행착오를 통해 성능 향상이 있었던 방법을 과제에 사용하였습니다.

[구현]

기본적인 자료구조(SparseMatrix,DenseMatrix) 와 BPR-MF는 추천라이브러리 Librec에서 참고하였습니다.