世纪佳缘会员推荐

周巍然 weiran.chow@gmail.com 20120723

0. 前言

本文主要介绍如何为世纪佳缘网站的进行会员推荐的算法流程，评测指标和探讨主要的方法思路。本文适合作为初学者对推荐系统的入门实践练习。本文档数据可以通过 e-mail 联系本人获取。

1. 问题定义

通过构造有效的统计评分模型，评估给定的候选会员集合中哪些会员更容易获得特定会员 A 的青睐。例如，如果需要给会员 A 推荐（rec）某 10 名指定的候选会员，则构造的模型应该能够将 1-10 号候选会员排序，排在前面的会员被认为更容易获得 A 的喜爱，从而引起 click 或 msg 的行为，其中 msg > click > rec。

2. 数据说明

本文提供了世纪佳缘网站中某城市会员在最近三个月内完整的交互行为数据以及相关会员资料。共包含 4 个数据集：train.txt，profile_f.txt，profile_m.txt 和 test.txt。

train.txt 包含约 860 万条交互记录，每条记录包括 4 个属性，涉及近 6 万名会员。格式如下：

USER_ID_A	USER_ID_B	ROUND	ACTION
100033	375879	1	rec
100033	381720	1	rec
...	...	...	...
100033	381720	18	rec
100033	417848	18	click
...	...	...	...

100033	327685	19	click
100033	327685	19	msg

test.txt 文件中包含了用来在线验证推荐算法效果的会员配对（interaction），及每对会员在三个月内的推荐次数。男女会员资料（包括部分择偶要求）分别记录在 profile_m.txt 和 profile_f.txt 中。每位会员包含 34 个特征变量（feature），我们提供了字段列表来说明不同特征变量的含义。

数据集信息汇总如下：

	Training	Validation*	Test
# USER_ID_A	15,000	4,200	10,433
# USER_ID_B	55,871	50,459	54,409
# INTERACTION	8,599,012	2,247,217	5,509,312
# FEATURE	34		

3. 评测指标

NDCG 在 Learning to Rank 问题中被广为采用。与经常用来比较两个序之间的相关程度的肯德尔和谐系数相比，NDCG 的不同之处在于，它更强调和关注两个序中排名靠前的部分的相关程度。这与我们的实际需求相符，因为我们更希望将与会员 A 有可能发生 msg 行为的候选会员排名靠前。

Normalized Discounted Cumulative Gain（NDCG），定义如下：

$$NDCG = \frac{DCG}{Ideal\ DCG}$$

这里 $DCG = \sum_{i=1}^{\min(10,n)} \frac{2^{y_i} - 1}{\log_2(1+i)}$。$y_i$ 表示模型给出的排序中，排名为 i 的候选会员的实际 ACTION 值（msg=2，click=1，rec=0）。对每一位获得推荐建议的会员 A，都需要计算一个相应的 NDCG。所有获得推荐建议的会员对应的 NDCG 的平均值，作为排名的主要依据。

$\min(10,n)$ 表示计算 NDCG 时仅采用排序前 10 的候选会员的 ACTION 进行计算，因此将尽可能多的 msg 或 click 排在前面至关重要。指数变换 2^y 是为了增大 ACTION 间的差异以凸显 msg 和 click 的重要性。折扣因子 $\log_2(1+i)$ 用来强调越

能将 msg 会员排名靠前的算法越好。例如，两种不同的推荐算法给出的排序对应的真实 ACTION 如下表所示，由于 RANK 1 算出的 NDCG 为 0.8045，而 RANK 2 算出的 NDCG 仅有 0.7579，我们认为 RANK 1 对应的算法更好。

RANK 1	click	msg	rec	click	rec
RANK 2	click	click	msg	rec	rec

这里给出一个计算 NDCG 的例子。假设某统计评分模型对 5 位会员进行了评分，以确定哪位会员更可能获得会员 A 的青睐（评分越高表示兴趣越大）：

USER_ID_B	1	2	3	4	5
模型评分	1.2	0.7	-2.5	0.2	4.0
按评分排序	2	3	5	4	1
ACTION (y)	msg (y2=2)	click (y3=1)	rec (y5=0)	rec (y4=0)	rec (y1=0)

因此对于会员 A，

$$DCG = \frac{2^0-1}{\log_2(1+1)} + \frac{2^2-1}{\log_2(1+2)} + \frac{2^1-1}{\log_2(1+3)} + \frac{2^0-1}{\log_2(1+4)} + \frac{2^0-1}{\log_2(1+5)} = 2.392789$$

如果能够获得的评分足够理想，从而能够完美地预测出会员 A 关于 5 位会员的兴趣排序，则此时相应的 DCG 称为 Ideal DCG：

$$\text{Ideal DCG} = \frac{2^2-1}{\log_2(1+1)} + \frac{2^1-1}{\log_2(1+2)} + \frac{2^0-1}{\log_2(1+3)} + \frac{2^0-1}{\log_2(1+4)} + \frac{2^0-1}{\log_2(1+5)} = 3.63093$$

从而对会员 A，$NDCG = \frac{2.392789}{3.63093} = 0.659$。

NDCG 指标的具体使用，参考附录代码 Splitedata.py，labelstrain.py 和 evaluate.py。其中 Splitedata.py 可设置参数 M 和 K 并训练集 train.txt 随机分为 M 份，取其中 1 份作为测试集，M-1 份作为训练集，每次实验选用不同的 K，这样做的主要目的是防止结果的过拟合(Over fitting)；而 labelstrain.py 可以将 train.txt 转化为每一行按 userA 排序的 userB 的 action 列表，而每一个 action，根据 msg->2,click->1,rec->0 进行转换；evaluate.py 可以根据 labelstrain.py 生成的 train.txt.test.labels 文件和 DODM.java 文件生成的 train.txt.test.rank 文件，采用 NDCG 指标进行评测。得到最终的评测结果（以本文算法为例，M=5,K=2）：

python evaluate.py ./data/train.txt.test.labels ./data/train.txt.test.rank

(0.50575013168322669, 0.53115324890919724)

而随机算法的结果为：(0.08659561709415893, 0.11637114804038115)

4. 解决方案

由于数据存在稀疏性及冷启动问题，也就是对新注册的用户进行推荐，这是本问题最大的难点。 针对稀疏数据和冷启动的数据特性，经典的算法诸如，最近邻的协同过滤算法、PageRank 排序算法、E-Greedy 排序算法、关联规则挖掘等并不太奏效，因为对于新注册的用户没有用户行为可分析。

而最直接的想法是，根据分别记录在 profile_m.txt 和 profile_f.txt 中男女会员资料（包括部分择偶要求），其中每位会员包含 34 个特征变量（feature）。在考虑到特征之间的择偶要求是否匹配，建立一个回归模型，根据会员 A 和会员 B 的交互行为进行评分，如 msg 给出 2 分，click 给出 1 分，rec 给出 0 分。然而这样做收效甚微。事实上，因为人的行为太随机了，而且人往往重相貌高于其他，不会仅仅因为你年龄、身高符合要求就和你 msg。profile 可能并不奏效，用户行为才真正值得我们挖掘。职业，房、车等因素都会有价值，可是，这些价值也能在用户行为中得到体现。

大众受欢迎度+少量的 profile 信息也许是解决冷启动问题的最佳方式。根据大众受欢迎度来给出推荐的做法很简单。具体参见 dodm.java。其中，大众欢迎度的定义函数如下：

```
  /**
    * caculate the popularity of the userB in trainFile
(userA,userB,times, rec|click|msg),\n
    * p(userB)= rec*(1)+click*10+msg*100;
    *
    * @param trainFile input train.txt data format like (userA, userB,
times,rec|click|msg)
    * @throws IOException
    * @return HashMap<userB,p>
    */
   private static HashMap<String,Integer> popularity(String trainFile)
throws IOException{
       HashMap<String,Integer> popMap = new HashMap<String,Integer>();
       BufferedReader reader = null;
       try {
           reader = new BufferedReader(new FileReader(trainFile));
           String line = reader.readLine(); // abandon first line
```

```
            while ((line = reader.readLine()) != null) {
                String[] fields = line.split(" ");
                if (fields.length == 4) {
                    String userB = fields[1];
                    String action = fields[3];
                    if (popMap.containsKey(userB)) {
                        Integer cur_pop = popMap.get(userB);
                      popMap.put(userB, cur_pop + getPopularity(action));
                    } else {
                        popMap.put(userB, getPopularity(action));
                    }
                }
            }
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return popMap;
    }
    /**
     * calculate the points of action.
     * @param action like (rec|click|msg)
     * @return rec=1,click=10,msg=100
     */
    private static Integer getPopularity(String action) {
        Integer pop = 0;
        if (action.equals("rec")) {
            pop = 1;
        } else if (action.equals("click")) {
            pop = 10;
        } else if (action.equals("msg")) {
            pop = 100;
        }
        return pop;
    }
```

我们将训练集中 userB 的大众欢迎度存入 HashMap 中，由于 userA 大多是新注册用户，而 userB 大多是老用户。因此，我们可以用训练集中的 userB 的大众欢迎度对测试集的 userB 进行打分，若测试集的 userB 不在 HashMap 中，则得分

为 0（这部分评分可以通过 profile 信息进行修正，具体做法本文不再涉及）。于是，我们使用附件的 DODM.java 程序对 train.txt.train 进行训练并对 train.txt.test 进行打分排序，得到 train.txt.test.rank 文件。再参见第 3 节使用 evaluate.py 对 train.txt.test.rank 进行评测，为防止过拟合，通常用进行 M 此实验再去平均值作为最终结果。与随机算法的结果相比，采用大众欢迎度来进行会员推荐，已经大大提升了 NDCG 指标。而对于无法计算大众欢迎度的 userB，若再通过其会员资料信息，预测一个权值较小的得分，则可以进一步改善预测精度。

5. 参考资料

1. 《集体智慧编程》，TOBY SEGARAN，2009
2. 《推荐系统实践》，项亮，2012
3. http://en.wikipedia.org/wiki/Discounted_Cumulative_Gain
4. http://en.wikipedia.org/wiki/Overfitting

附录

Evaluate.py

```
#说明：NDCG 指标评测
#使用：python evaluate.py ./data/train.txt.test.labels ./data/train.txt.test.rank
from math import log
def evaluate_submission(labels, ranks, k = 10):
    nq = len(labels) # Number of user_id_a
    assert len(ranks) == nq, 'Expected %d lines, but got %d.'%(nq, len(ranks))
    ndcg10 = 0.0
    ndcg20 = 0.0
    for i in range(nq):
        l = [int(x) for x in labels[i].split(' ')]
        try:
            r = [int(x) for x in ranks[i].split(' ')]
        except ValueError:
            raise ValueError('Non integer value on line %d'%(i+1))
        nd = len(l)
        gains = [-1]*nd # must be initialized as -1
```

```
            for j in range(nd):
                gains[r[j]-1] = 2**l[j] - 1.0
            assert min(gains) >= 0, 'Not all ranks present at line %d.'%(i+1)
            dcg10 = sum([g/log(j+2) for (j, g) in enumerate(gains[:k])])
            dcg20 = sum([g/log(j+2) for (j, g) in enumerate(gains[:(2*k)])])
            gains.sort()
            gains = gains[::-1]
            ideal_dcg10 = sum([g/log(j+2) for (j, g) in enumerate(gains[:k])])
            ideal_dcg20 = sum([g/log(j+2) for (j, g) in enumerate(gains[:(2*k)])])
            if ideal_dcg10 != 0.0:
                ndcg10 += dcg10/ideal_dcg10
            else:
                ndcg10 += 1.0
            if ideal_dcg20 != 0.0:
                ndcg20 += dcg20/ideal_dcg20
            else:
                ndcg20 += 1.0
    return (ndcg10/nq, ndcg20/nq)
import sys
labels = []
f_labels = open(sys.argv[1])
while True:
    newline = str(f_labels.readline())
    if len(newline) == 0:
        break
    else:
        labels.append(newline)

ranks = []
f_ranks = open(sys.argv[2])
while True:
    newline = str(f_ranks.readline())
    if len(newline) == 0:
        break
    else:
        ranks.append(newline)
ndcg10, ndcg20 = evaluate_submission(labels, ranks, k = 10)
print(ndcg10, ndcg20)
```

Labelstrain.py

```
# coding=gb2312
# labels test data by ACTION：  msg = 2 > click = 1 > rec = 0
# test<userA,userB,times,aciton> -> userBList<2,1,0,0>
```

```
# @author wrchow
# @date 20120722
def labelstrain(data):
    labels = []
    lst_userA = "="
    userBList = []
    for line in data:
        (userA,userB,times,action) = line.split(' ')
        if userA == "USER_ID_A":
            continue
        if lst_userA == "=" or userA == lst_userA:
            userBList.append(getActionScore(action))
        else:
            labels.append(str(' '.join(userBList))+'\r\n')
            userBList = []
            userBList.append(getActionScore(action))
        lst_userA = userA
    labels.append(str(' '.join(userBList)) + '\r\n')
    return labels

def getActionScore(action):
    if action == "rec\r\n":
        return '0'
    elif action == "click\r\n"    :
        return '1'
    else:
        return '2'

import sys
if len(sys.argv) < 2:
    print 'Usage: labelstrain <train.test>'
    sys.exit()

f_data = open(sys.argv[1])
data = []
#f_data.readline()#aband first line
while True:
    newline = str(f_data.readline())
    if len(newline) == 0:
        break
    else:
        data.append(newline)
f_data.close()
```

```
print len(data)
labels = labelstrain(data)
print len(labels)


f_labels = open(sys.argv[1] + '.labels', 'w')
f_labels.writelines(labels)
f_labels.close()
```

Splitdata.py

```
# encoding=gb2312
# for split train data into train.test and train.train randomly
# @auther wrchow
# @date 20120722
import random
def splitdata(data, M, k, seed):
    test = []
    train = []
    random.seed(seed)
    for line in data:
        if random.randint(0,M) == k:
            test.append(line)
        else:
            train.append(line)
    return train, test

'''read data from train.txt'''
import sys
if len(sys.argv) < 3:
    print 'Usage: splitdata <train.txt> <K(0~5)>'
    sys.exit()

f_data = open(sys.argv[1])
data = []
newline = str(f_data.readline()) # abandon first lines
while True:
    newline = str(f_data.readline())
    if len(newline) == 0:
        break
    else:
        data.append(newline)
f_data.close()
```

```
train, test = splitdata(data,5,int(sys.argv[2]),10)
print (len(train),len(test))
f_train = open(sys.argv[1]+'.train', 'w')
f_train.writelines(train)
f_train.close()
f_test = open(sys.argv[1]+'.test','w')
f_test.writelines(test)
f_test.close()
print 'look file ' + sys.argv[1] + '.train and ' + sys.argv[1] + '.test'
```

Dodm.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;
import java.util.Collections;
import java.util.Map.Entry;

/**
 * recommender system for sjjy members.
 *
 * @author zhouweiran
 * @date 20120720~20120722
 */
public class DODM {
	/**
	 * get ranks from train popularity and sort userB list of userA \n
	 * the result is a serial number.
	 * @param[in] trainFile file format like (userA,userB,times,rec|click|msg)
	 * @param[in] testFile file format like (userA,userB,times)
	 * @throws IOException
	 */
	public static void getRanks(String trainFile,String testFile) throws IOException{
		HashMap<String,Integer> popMap = popularity(trainFile);
		BufferedReader reader = null;
		try {
			reader = new BufferedReader(new FileReader(testFile));
```

```
            Vector<Vector<Integer>> userBVSet = new Vector<Vector<Integer>>();
            String line = "";
            String lst_userA = "=";
            Vector<Integer> userBV = new Vector<Integer>();
            int count_test_userB = 0;
            while ((line = reader.readLine()) != null) {
                String[] fields = line.split(" ");
                String userA = fields[0];
                String userB = fields[1];
                int pop = 0;
                if (popMap.get(userB) != null) {
                    pop = popMap.get(userB);
                }
                if (lst_userA.equals("=") || lst_userA.equals(userA)) {
                    userBV.add(pop);
                } else {
                    count_test_userB += userBV.size();
                    userBVSet.add(getRankNo(userBV));
                    userBV.clear();
                    userBV.add(pop);
                }
                lst_userA = userA;
            } // while ((line = reader.readLine()) != null)
            userBVSet.add(getRankNo(userBV));
            writeToFile(userBVSet,testFile+".rank");
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch(IOException e) {
                    e.printStackTrace();
                }
            }
        }
}
/**
  * rank the numbers and write the serial number to "filename.rank". \n
  * userA<123>-userB<331,221,441,551>:pop<3,3,1,5> -> userB.serial.no<2,3,4,1>.
  *
  * @param userBV a vector of disorder numbers
  * @return Vector<Integer> ranked userB lists
  */
@SuppressWarnings("unchecked")
public static Vector<Integer> getRankNo(Vector<Integer> userBV) {
```

```
        Map<Integer,Integer> userBVMap = new HashMap<Integer,Integer>();
        for (int i=0; i<userBV.size(); i++) {
            userBVMap.put(i,userBV.elementAt(i));
        }
        List<Map.Entry<Integer, Integer>> mapList =
            new ArrayList<Map.Entry<Integer, Integer>>(userBVMap.entrySet());
        Collections.sort(mapList,
                new Comparator<Map.Entry<Integer, Integer>>(){
                    @Override
                    public int compare(Entry<Integer, Integer> o1,
                            Entry<Integer, Integer> o2) {
                        return o1.getValue()>o2.getValue()?1:-1;
                    }
                }
        );
        Vector<Integer> userBV_ranked = new Vector<Integer>();
        userBV_ranked = (Vector<Integer>) userBV.clone();
        int no = 1;
        for (Entry<Integer, Integer> e : mapList) {
            userBV_ranked.set(e.getKey(),no++);
        }
        return userBV_ranked;
    }
    /**
     * write the matrix userB list serial number to file.
     *
     * @param userBVSet Vector<Vector<Integer>>
     * @param filename
     */
    public static void writeToFile(Vector<Vector<Integer>> userBVSet, String filename) {
        try {
            FileWriter fw = new FileWriter(filename);
            for (Vector<Integer> userBV : userBVSet) {
                int cnt=0;
                for (Integer userB : userBV) {
                    if(cnt++ > 0) {
                        fw.append(' ');
                    }
                    fw.append(Integer.toString(userB));
                }
                fw.append('\n');
            }
            fw.close();
        } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
        /**
         * caculate the popularity of the userB in trainFile(userA,userB,times,rec|click|msg),\n
         * p(userB)= rec*(-1)+click*10+msg*100;
         *
         * @param trainFile input train.txt data format like (userA,userB,times,rec|click|msg)
         * @throws IOException
         * @return HashMap<userB,p>
         */
        private static HashMap<String,Integer> popularity(String trainFile) throws IOException{
            HashMap<String,Integer> popMap = new HashMap<String,Integer>();
            BufferedReader reader = null;
            try {
                reader = new BufferedReader(new FileReader(trainFile));
                String line = reader.readLine(); // abandon first line
                while ((line = reader.readLine()) != null) {
                    String[] fields = line.split(" ");
                    if (fields.length == 4) {
                        String userB = fields[1];
                        String action = fields[3];
                        if (popMap.containsKey(userB)) {
                            Integer cur_pop = popMap.get(userB);
                            popMap.put(userB, cur_pop + getPopularity(action));
                        } else {
                            popMap.put(userB, getPopularity(action));
                        }
                    }
                }
            } finally {
                if (reader != null) {
                    try {
                        reader.close();
                    } catch(IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            System.out.println("HashMap of " + trainFile + ".size() =" + popMap.size()); // log the
num of userB in train.txt
            return popMap;
        }
        /**
```

```
     * calculate the points of action.
     * @param action like (rec|click|msg)
     * @return rec=-1,click=10,msg=100
     */
    private static Integer getPopularity(String action) {
        Integer pop = 0;
        if (action.equals("rec")) {
            pop = 1; // try -1 and 1
        } else if (action.equals("click")) {
            pop = 10;
        } else if (action.equals("msg")) {
            pop = 100;
        }
        return pop;
    }
    public static void main(String args[]) throws IOException {
        if (args.length != 2) {
            System.err.println("Usage: DODM <input path> <output path>");
            System.exit(-1);
        }
        DODM.getRanks(args[0],args[1]);
    }
}
```