郭兰哲

专注机器学习

主页

所有文章

标签云

关于我

python实现推荐系统

2016-06-25

machine learning 机器学习实战

两种最普遍的推荐系统的类型是基于内容和协同过滤（CF）。协同过滤基于用户对产品的态度产生推荐，基于内容的推荐系统基于物品属性的相似性进行推荐。CF可以分为基于内存的协同过滤和基于模型的协同过滤。
我们将使用MovieLens数据集，它是在实现和测试推荐引擎时所使用的最常见的数据集之一，包含来自943个用户以及精选的1682部电影的评分。 下载地址

导入numpy和pandas库

```
1 import numpy as np
2 import pandas as pd
```

读入u.data数据文件

```
1 header = ['user_id', 'item_id', 'rating', 'timestamp']
2 df = pd.read_csv('u.data', sep = '\t', names = header)
```

查看用户和电影的数量

```
1 n_users = df.user_id.unique().shape[0]
2 n_items = df.item_id.unique().shape[0]
3 print 'Number of users = ' + str(n_users) + ' | Number of movies = ' + str(n_ite
```

```
1 Number of users = 943 | Number of movies = 1682
```

郭兰哲

专注机器学习

主页

所有文章

标签云

关于我

使用scikit-learn库将数据集分割成测试集和训练集，调用Cross_validation.train_test_split根据测试样本的比例(test_size)将数据混洗并分割成两个数据集。

```
1 from sklearn import cross_validation as cv
2 train_data,test_data = cv.train_test_split(df, test_size = 0.25)
```

##基于内存的协同过滤

基于内存的协同过滤方法可以分为两个部分：用户 - 产品协同过滤和产品 - 产品协同过滤。用户 - 产品协同过滤将选取一个特定的用户，基于打分的相似性发现类似于该用户的用户，并推荐那些相似用户喜欢的产品。产品 - 产品协同过滤会选取一个产品，发现喜欢该产品的用户，并找到这些相似用户还喜欢的其它产品。

用户 - 产品协同过滤：“喜欢这东西的人也喜欢……”
产品 - 产品协同过滤：“像你一样的人也喜欢……”

在这两种情况下，从整个数据集构建一个用户产品矩阵。

用户产品矩阵的例子：

计算相似性，并创建一个相似性矩阵。

在产品 - 产品协同过滤中的产品之间的相似性是通过观察所有对两个产品打分的用户来度量的。

在用户 - 产品协同过滤中的用户之间的相似性是通过观察所有同时被两个用户打分的产品来度量的。

通常用于推荐系统中的距离矩阵是余弦相似性，其中，打分被看成n维空间中的向量，而相似性是基于这些向量之间的角度进行计算的。用户a和m的余弦相似性可以用下面的公

式进行计算：

$$s_u^{cos}(u_k, u_a) = \frac{u_k \cdot u_a}{||u_k|| \, ||u_a||} = \frac{\sum x_{k,m} x_{a,m}}{\sqrt{\sum x_{k,m}^2 \sum x_{a,m}^2}}$$

要计算产品m和b之间的相似性，使用公式：

$$s_u^{cos}(i_m, i_b) = \frac{i_m \cdot i_b}{||i_m|| \, ||i_b||} = \frac{\sum x_{a,m} x_{a,b}}{\sqrt{\sum x_{a,m}^2 \sum x_{a,b}^2}}$$

创建用户产品矩阵，针对测试数据和训练数据，创建两个矩阵：

```
train_data_matrix = np.zeros((n_users,n_items))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1] = line[3]
test_data_matrix = np.zeros((n_users, n_items))
for line in test_data.itertuples():
    test_data_matrix[line[1]-1, line[2]-1] = line[3]
```

使用sklearn的pairwise_distances函数来计算余弦相似性。

```
from sklearn.metrics.pairwise import pairwise_distances
user_similarity = pairwise_distances(train_data_matrix, metric = "cosine")
item_similarity = pairwise_distances(train_data_matrix.T, metric = "cosine")
```

已经创建了相似性矩阵：user_similarity和item_similarity，因此，可以通过基于用户的CF应用下面的公式做出预测：

郭兰哲

专注机器学习

主页
所有文章
标签云
关于我

郭兰哲

专注机器学习

主页

所有文章

标签云

关于我

可以将用户k和用户a之间的相似性看成权重，乘以相似用户a(校正的平均评分用户)的评分，这里需要规范化该值，使得打分位于1到5之间，最后对尝试预测的用户的平均评分求和。

基于产品的CF应用下面的公司进行预测，此时无需纠正用户的平均打分

```
def predict(rating, similarity, type = 'user'):
    if type == 'user':
        mean_user_rating = rating.mean(axis = 1)
        rating_diff = (rating - mean_user_rating[:,np.newaxis])
        pred = mean_user_rating[:,np.newaxis] + similarity.dot(rating_diff) / np.a
    elif type == 'item':
        pred = rating.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

```
item_prediction = predict(train_data_matrix, item_similarity, type = 'item')
user_prediction = predict(train_data_matrix, user_similarity, type = 'user')
```

####评估

这里采用均方根误差(RMSE)来度量预测评分的准确性

可以使用sklearn的mean_square_error(MSE)函数，其中RMSE仅仅是MSE的平方根。

```
from sklearn.metrics import mean_squared_error
from math import sqrt
def rmse(prediction, ground_truth):
    prediction = prediction[ground_truth.nonzero()].flatten()
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    return sqrt(mean_squared_error(prediction, ground_truth))
```

```
print 'User based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix))
```

```
print 'Item based CF RMSe: ' + str(rmse(item_prediction, test_data_matrix))
```

```
User based CF RMSE: 3.12466203536
Item based CF RMSe: 3.45056350625
```

可以看出，基于内存的算法很容易实现并产生合理的预测质量。

##基于模型的协同过滤

基于模型的协同过滤是基于矩阵分解(MF)的，矩阵分解广泛应用于推荐系统中，它比基于内存的CF有更好的扩展性和稀疏性。MF的目标是从已知的评分中学习用户的潜在喜好和产品的潜在属性，随后通过用户和产品的潜在特征的点积来预测未知的评分。

计算MovieLens数据集的稀疏度：

```
sparsity = round(1.0 - len(df) / float(n_users*n_items),3)
print 'The sparsity level of MovieLen100K is ' + str(sparsity * 100) + '%'
```

```
The sparsity level of MovieLen100K is 93.7%
```

###SVD

一般的方程可以表示为：

$$X = USV^T$$

给定m * n矩阵X:

U 是一个(m * r)正交矩阵

S 是一个对角线上为非负实数的(r * r)对角矩阵

郭兰哲

专注机器学习

主页
所有文章
标签云
关于我

郭兰哲

专注机器学习

主页

所有文章

标签云

关于我

V^T是一个(r * n)正交矩阵

S的对角线上的元素被称为X的奇异值。

阵X可以被分解成U，S和V。U矩阵表示对应于隐藏特性空间中的用户的特性矩阵，而V矩阵表示对应于隐藏特性空间中的产品的特性矩阵。

现在，可以通过U, S和V^T的点积进行预测了：

```
1
2 import scipy.sparse as sp
3 from scipy.sparse.linalg import svds
4 u, s, vt = svds(train_data_matrix, k = 20)
5 s_diag_matrix = np.diag(s)
6 x_pred = np.dot(np.dot(u,s_diag_matrix),vt)
7 print 'User-based CF MSE: ' + str(rmse(x_pred, test_data_matrix))
8
```

```
1 User-based CF MSE: 2.72035726617
```

总结：

实现了简单的协同过滤方法，包括基于内存的CF和基于模型的CF

基于内存的模型是基于产品或用户之间的相似性，这里采用余弦相似性。

基于模型的CD是基于矩阵分解，采用SVD来分解矩阵

标准的协同过滤方法在面对冷启动的情况时表现不佳。

本文标题: python实现推荐系统

文章作者: 郭兰哲
发布时间: 2016-06-25, 16:33:39
最后更新: 2016-08-29, 21:57:04
原始链接: http://yoursite.com/2016/06/25/36.python实现推荐系统/
许可协议: "署名-非商用-相同方式共享 4.0" 转载请保留原文链接及作者。

郭兰哲

专注机器学习

主页
所有文章
标签云
关于我

← kaggle实战-titanic

HMM三个基本问题的数学推导及python实现 →