个性化推荐

张岚

December 11, 2016

1. 使用全量数据，取 index 为用户 id，column 为电影 id，从训练集中取对应的用户 id 对电影 id 的评分，建立一个新的矩阵，对于未知的项全定为 0，见算法 1。
   由于数据中，测试集的电影维度（电影数目：9983）和训练集（电影数目：10000）不一致，如果直接在训练集中去掉在测试集中不存在的 17 部电影，会对用户相似度计算（矩阵分解）产生影响。因此本文在数据处理阶段记录差异的电影，计算完相似度（矩阵分解）后，删除测试集没有的电影，保证矩阵运算维度的准确性。

算法 **1** 数据预处理

输入**:** $train$ 训练数据,$test$ 测试数据

function procData($train, test$)
    $uids \leftarrow train.uid.unique.sort\_values()$
    $fids \leftarrow train.fid.unique.sort\_values()$
    $tfids \leftarrow test.fid.unique.sort\_values()$
    $diff \leftarrow fids - tfids$
    $df1 \leftarrow dataframe(uids, fids)$
    $df2 \leftarrow dataframe(uids, tfids)$
    for $line \in train$ **do**
        $df1.loc[line[1], line[2]] \leftarrow line[3]$
    end for
    for $line \in test$ **do**
        $df2.loc[line[1], line[2]] \leftarrow line[3]$
    end for
    return $df1, df2, diff$
end function

2. 使用训练集的数据计算用户的相似度，预测测试集中用户对电影的打分，最后评估准确率。最终结果 RMSE 值为 0.5993。本文使用矩阵形式运算程序运行 CPU 耗时 47.0688M。如果直接去掉训练集中的值，最终结果 RMSE 值为 0.5987，差异的确不大。使用矩阵形式的协同过滤方法见算法 2。

算法 2 协同过滤

输入: X_train 训练数据,X_test 测试数据,$diff$ 差异电影

function TASK1($X\_train, X\_test, diff$)
    $sim \leftarrow cosine\_similarity(X\_train)$
    $pred \leftarrow sim.dot(X\_train)/np.array([np.abs(sim).sum(axis = 1)]).T$
    $pred \leftarrow delete(pred, diff, axis = 1)$
    $print \leftarrow RMSE(X\_test, pred)$
end function

3. 矩阵分解算法见算法 3。产生从 10 到 90 步长为 5 的 k，从 0.001 到 0.1 步长为 0.004 的 λ，从 0.0001 到 0.1 步长为 0.0004 的 α，分别计算不同 k、λ、α 对应的 RMSE，并记录下其中最小的 RMSE。

    a. 给定 $k = 50$, $\lambda = 0.01$，不同 α 得到的目标函数值见图 1和测试集上 RMSE 变化见图 2。

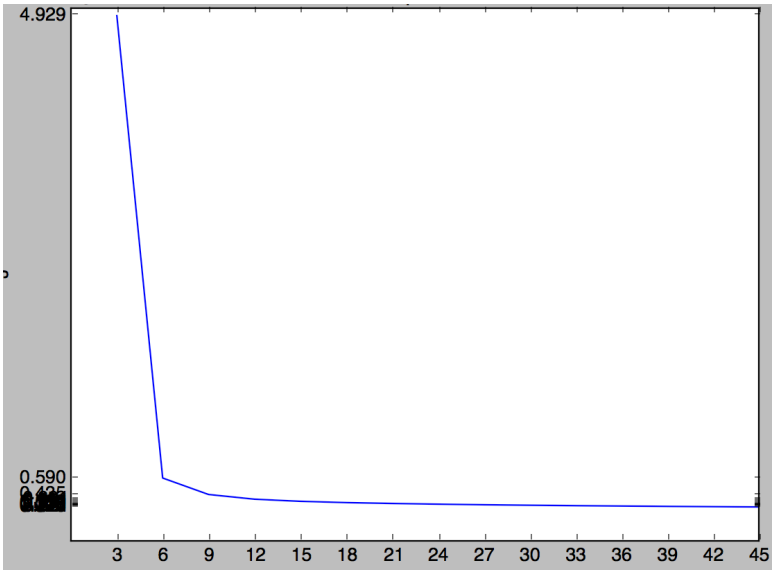

Figure 1: J

算法 **3** 矩阵分解

输入**:** X_train 训练数据,X_test 测试数据,$diff$

function TASK2($X\_train, X\_test, diff$)
    $ks \leftarrow np.arange(10, 90, 5)$
    $min \leftarrow 'min' : NaN$
    for $k \in ks$ **do**
        $lambdas \leftarrow np.arange(0.001, 0.1, 0.004)$
        for $lambda \in lambdas$ **do**
            $alphas = np.arange(0.0001, 0.1, 0.0004)$
            $resRMSE \leftarrow []$
            $resJ \leftarrow []$
            for $alpha \in alphas$ **do**
                $(a, u, v) = random(k, X_train)$
                $count \leftarrow 0$
                $j \leftarrow calJ(a, x, u, v, lambda)$
                while count<1000 and j>0.1 **do**
                    $count+ = 1$
                    $u \leftarrow u - alpha * calJU(a, x, u, v, lambda)$
                    $v \leftarrow v - alpha * calJV(a, x, u, v, lambda)$
                    $j \leftarrow calJ(a, x, u, v, lambda)$
                end while
                $pred \leftarrow u.dot(v.T)$
                $pred \leftarrow delete(pred, diff, axis = 1)$
                $rmse \leftarrow RMSE(X\_test, pred)$
                $resRMSE \leftarrow resRMSE \cup rmse$
                $resJ \leftarrow resJ \cup j$
                if $min['min'] > rmse$ **then**
                    min['min'] = rmse
                    min['k']=k
                    min['lambda'] = lambda
                    min['alpha']=alpha
                end if
            end for
            print k,lambda,alpha,resJ,resRMSE
        end for
    end for
    print min
end function

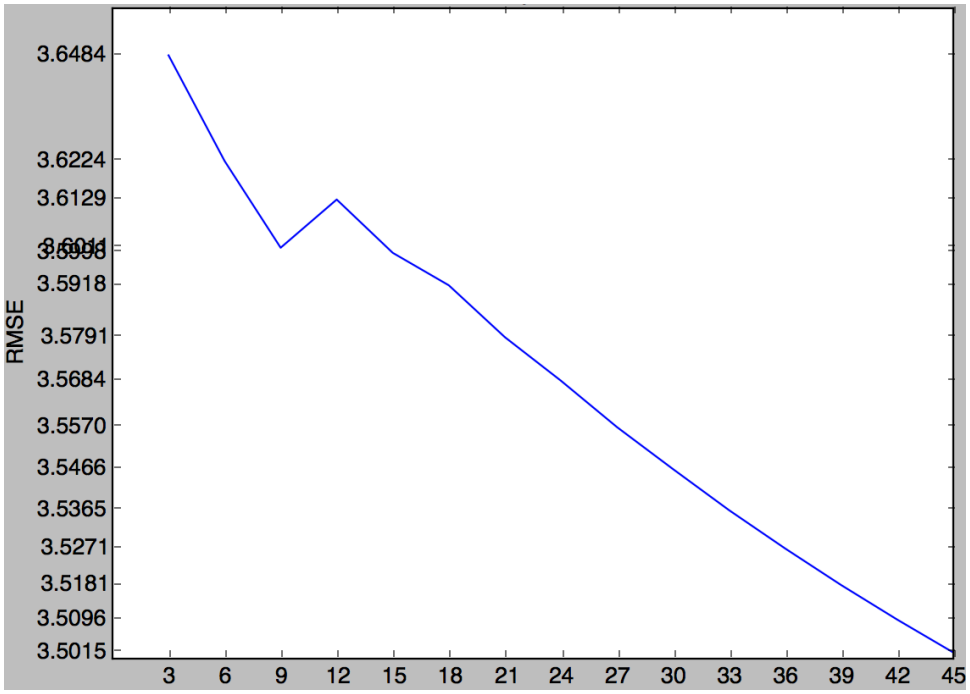

Figure 2: RMSE

b. k 值越大，计算需要设定的学习率 α 越大，迭代耗时越长；λ 控制正则项大小的参数。程序最终选择 $k = 30$，$\lambda = 0.09$，$\alpha = 9e - 05$，迭代 1000 次后，最终 $RMSE = 2.68$。

4. 协同过滤和矩阵分解对比

   a. 协同过滤很容易实现并产生合理的预测质量。其中本文使用的用户相似度矩阵适合用户比较少的场合，否则计算代价很大。如果用户做出评价过少，很容易导致算出的相关系数不准确。

   b. 矩阵分解很容易扩展加入新的特征（如用户的个人特征：年龄组、地理位置、性别，和电影特征：年份、导演、演员等），但如果只有少数可用的数据，难以学习潜在特征，预测效果将会很差

   c. 在真实世界中，两种方法都很难解决一个新用户或者新电影进入系统，在并不了解他的喜好前，很难进行准确的预测

5. 如果一个用户给两个电影评分的时间差小于给定阈值 σ（两天以内），那么这两个电影之间很有可能具备相关性。

计算每个用户点击时间序列对，按照升序排列，在每个用户的商品点击序列中，计算两两商品时间序列对的点击时间差，并计算商品点击相关性 $score = 0.2^{timedelta}$。使用和用户相似度相似的计算公式 $sim = \frac{\sum_{i_b} sim_i(i_m, i_b)(s_{k,b})}{\sum_{i_b} |sim_i(i_m, i_b)|}$，计算商品相似度。
将点击相似度和商品相似度相加，得到最终的相似度。再使用协同过滤相似方法预测测试集打分情况和 RMSE，最终 RMSE 为 0.50158。