South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
梁肖剑，梁子豪，林艾琳

Supervisor:
Qingyao Wu

Student ID：
201530612149,201530612163,20
153606122000

Grade:
Undergraduate

December 9, 2017

Recommender System Based on Matrix Decomposition

Abstract—

I. INTRODUCTION

1. Explore the construction of recommended system.
2. Understand the principle of matrix decomposition.
3. Be familiar to the use of gradient descent.
4. Construct a recommendation system under small-scale dataset, cultivate engineering ability.

II. METHODS AND THEORY

SGD and ALS

III. EXPERIMENT

A.Experiment Step

The experiment code and drawing are both completed on jupyter.

Using alternate least squares optimization(ALS):

1. Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix and the item (movie) factor matrix , where is the number of potential features.
3. Determine the loss function and the hyperparameter learning rate and the penalty factor .
4. Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:

4.1 With fixd item factor matrix, find the loss partial derivative of each row (column) of the user factor matrices, ask the partial derivative to be zero and update the user factor matrices.

4.2 With fixd user factor matrix, find the loss partial derivative of each row (column) of the item factor matrices, ask the partial derivative to be zero and update the item

4.3 Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.

5. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , Draw a curve with varying iterations.
6. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

Using stochastic gradient descent method(SGD):

1. Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix and the item (movie) factor matrix , where is the number of potential features.
3. Determine the loss function and hyperparameter learning rate and the penalty factor .
4. Use the stochastic gradient descent method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:

4.1 Select a sample from scoring matrix randomly;

4.2 Calculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix;

4.3 Use SGD to update the specific row(column) of and ;

4.4 Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.

5. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , Draw a curve with varying iterations.
6. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

B. Experiment Step

```
import numpy as np
import matplotlib.pyplot as plt
# 从指定路径文件中加载数据
def load_data(path):
    rating_mat = np.zeros((943, 1682))     # 初始化评分矩阵
    f = open(path)
    for line in f.readlines():
        nums = line.split("\t")                    # 按制表符划分
        nums = [int(x) for x in nums]              # 转化为数字
        rating_mat[nums[0]-1][nums[1]-1] = nums[2]
    f.close()
    return rating_mat
    # 固定 Q 更新 P
def gradient(P, Q, R, lamb):
    row = P.shape[0]
    for u in range(row):
        b = R[u] != 0
        n = R[u].sum()
        P[u]        =        np.linalg.pinv(Q[b].T.dot(Q[b])        +
lamb).dot(Q.T.dot(R[u].reshape(1,-1).T)).ravel()
    return P

   # 计算均方根误差
def Loss(P, Q, R):
    b = R != 0
    loss = np.sqrt(((R[b] - (P.dot(Q.T))[b])** 2).mean())
    return loss

# 交替最小二乘
def ALS(R_train, R_val):
    K = 20          # 潜在特征数
    lamb = 149011611938476.56       # 正则化参数
    iterations = 20   # 训练轮数
    loss_train = []
    loss_val = []
    n_user, n_item = R_train.shape
    np.random.normal
    P = np.random.normal(0,0.1,(n_user, K))
    Q = np.random.normal(0,0.1,(n_item, K))
    for i in range(iterations):
        P = gradient(P, Q, R_train, lamb)
        Q = gradient(Q, P, R_train.T, lamb)
        loss_t = Loss(P, Q, R_train)
        loss_train.append(loss_t)
        loss_v = Loss(P, Q, R_val)
        loss_val.append(loss_v)
    return loss_train,loss_val

R_train = load_data("ml-100k/u2.base")
R_val = load_data("ml-100k/u2.test")
loss_train, loss_val = ALS(R_train, R_val)
x = len(loss_train)
plt.figure()
plt.title("Recommander System")
plt.xlabel("iterations")
plt.ylabel("loss")
plt.plot(range(x), loss_train, "r",label="train")
plt.plot(range(x), loss_val, "b",label="validation")
plt.legend(loc="upper right")
plt.show()
```

IV. CONCLUSION

USing SGD and ALS to construct Recommender System Based on Matrix Decomposition.