CSE 258 Assignment 1

PID: A53274222 Name: Ke Liu Kaggle name: DaisyL

Task 1 Purchase Prediction

1.1 Model

1.1.1 Most popular items

Like question 1 in Homework3, I found the most popular items by sorting the times of the purchase of every item. I set the threshold to 0.4*totalPurchase (that is, 200000). For a (user, item) pair in "pairs_Purchase.txt", if the item belongs to the set of most popular items, predict as Purchase (1).

1.1.2 Same Category

Like question 3 in Homework 3, I find the categories of all the users and items in the train set respectively. For a (user, item) pair in "pairs_Purchase.txt", return `True' if a user has purchased an item of the same category before (the user and the item have at least one category in common).

1.1.3 Jaccard Similarity

For a (user, item) pair in "pairs_Purchase.txt", find all the users that have bought this item. And then find the jaccard similarity of this user and all those users. If any the jaccard similarity is non-zero, return 1.

1.1.4 My model

My model is to combine the three methods mentioned above, if any of the three conditions is established, then we predict user will purchase the item. Otherwise, return 0. By choosing the proper threshold of most popular items, I obtained a score: 0.69307.

1.2 Models I tried and computing difficulties

I have tried to use pearson correlation and jaccard similarity to find the most similar users for the user we want to predict for. And if the item is within the items bought by the most similar users, we return true. But this method took too long(like 2 hours for our txt.) to get a result. And these two methods just decrease the accuracy.

Task 2 Rating Prediction

2.1 Model

The model I used : Latent-factor models

$$\arg\min_{\alpha,\beta,\gamma} \sum_{u,i}(\alpha+\beta_u+\beta_i+\gamma_u\cdot\gamma_i-R_{u,i})^2+\lambda\left[\sum_u \beta_u^2+\sum_i \beta_i^2+\sum_i \|\gamma_i\|_2^2+\sum_u \|\gamma_u\|_2^2\right]$$

I used the method shown in lecture 8: Alternating Least Squares. First fix γ_i, and solve $\arg min_{\alpha,\beta,\gamma_u}\ objective(\alpha,\beta,\gamma)$.

Then fix γ_u, and solve $\arg min_{\alpha,\beta,\gamma_i}\ objective(\alpha,\beta,\gamma)$, repeat until convergence.

2.2 Parameters

I need to determine the dimension k of γ_u and γ_i and the regularization constant λ to prevent overfitting. By observing the MSE of the validation set, the result turns better with k=1 and $\lambda = 6.4$.

2.3 Difficulties

I have trained the latent-factor models with γ_u and γ_i all set to zero, but it turns out they just would not change and cannot get a reasonable result. Then I set them with some random small values which actually work.

I have also tried gradient descent method to solve this problem, but the MSE will decrease at first and then have a little increment which I cannot explain.

3 Appendix

Partial code of task1(next page)

```python
def is_category(user,item):
    user_set = user_purchased_category[user]
    item_set = item_belong_to_category[item]
    for cat in item_set :
        if cat in user_set :
            return True
    return False
```

```python
## part 3
def jaccard(u,v) :
    set1 = u_bought[u]
    set2 = u_bought[v]
    if len(set1 | set2)==0 :
        return 0
    return len(set1 & set2)/len(set1 | set2)

def jaccard_u(i,u) :
    u_buy_i = i_be_bought[i]
    sim = 0
    for t in u_buy_i :
        sim += jaccard(t,u)
    if sim == 0:
        return 0
    else : return 1
```

Partial code of task2

```python
def als(lamda, fix_u,alpha,beta_u,beta_i,gamma_u,gamma_i):
    if not fix_u:
        for u in u_bought_rating:
            sum_gm = 0
            sum_bias = 0
            for i in u_bought_rating[u]:
                sum_gm += gamma_i[i]*gamma_i[i]
                sum_bias += gamma_i[i]*(u_bought_rating[u][i]-alpha-beta_u[u]-beta_i[i])
            gamma_u[u] = sum_bias/(lamda+sum_gm)

    else :
        for i in i_be_bought_rating:
            sum_gm = 0
            sum_bias = 0
            for u in i_be_bought_rating[i]:
                sum_gm += gamma_u[u]*gamma_u[u]
                sum_bias += gamma_u[u]*(i_be_bought_rating[i][u]-alpha-beta_u[u]-beta_i[i])
            gamma_i[i] = sum_bias/(lamda + sum_gm)
    # update alpha
    bias = 0
    for u,i,r in train:
        bias += (r-beta_u[u]-beta_i[i]-gamma_u[u]*gamma_i[i])
    alpha = bias/len(train)

    # update beta u
    for u in u_bought_rating :
        bias = 0
        for i in u_bought_rating[u]:
            bias += (u_bought_rating[u][i]-alpha-beta_i[i]-gamma_u[u]*gamma_i[i])
        beta_u[u] = bias/(lamda+len(u_bought_rating[u]))

    # update beta i
    for i in i_be_bought_rating:
        bias =0
        for u in i_be_bought_rating[i]:
            bias += (i_be_bought_rating[i][u]-alpha -beta_u[u]-gamma_u[u]*gamma_i[i])
        beta_i[i] = bias/(lamda +len(i_be_bought_rating[i]))
    return alpha,beta_u,beta_i,gamma_u,gamma_i
```