实验 3：搭建一个简单基于矩阵分解的推荐系统

一、 实验目的

1. 掌握矩阵分解的数学推导过程；
2. 利用 python numpy 等软件包，实现基于矩阵分解技术下的推荐系统。

二、 实验步骤

1．准备工作

首先下载 ratings_train_v1.csv 和 ratings_test_v2.csv。

可以具体点击其中的 csv 文件查阅相应的内容：

ratings_train_v1.csv

```
1 userId,movieId,rating,timestamp
2 1,804,4,964980499
3 1,1210,5,964980499
4 1,2628,4,964980523
5 1,2826,4,964980523
6 1,2018,5,964980523
7 1,3578,5,964980668
8 1,3617,4,964980683
9 1,3744,4,964980694
10 1,101,5,964980868
11 1,441,4,964980868
12 1,2858,5,964980868
13 1,1473,4,964980875
14 1,2997,4,964980896
```

在这个 csv 的文件中，包括了 userId 表示用户，movieId 表示电影，rating 表示评分，timestamp 表示时间。在本次实验中，timestamp 是没有用到的。

```
3 import pandas as pd
4 import numpy as np
5
6
7 train_rating = pd.read_csv("/home/jeff/rec/ratings_train_v1.csv")
8 test_rating = pd.read_csv("/home/jeff/rec/ratings_test_v1.csv")
9
```

我们通过 pandas 第三方库，导入这两个对应的 csv 文件。

2、 将上述得到的 csv 评分表转换为对应的 rating 矩阵。

首先我们先统计一共有多少用户和商品:

```
all_user = np.unique(train_rating['userId'])
all_item = np.unique(train_rating['movieId'])

```

这里我们调用了 np.unique 函数。

随后可以创建对应数量的空评分表格:

```
num_user = len(all_user)
num_item = len(all_item)

rating_mat = np.zeros([num_user, num_item], dtype=int)
```

随后可以得到

```
rating_mat = np.zeros([num_user, num_item], dtype=int)

for i in range(len(train_rating)):
    user = int(train_rating.iloc[i]['userId'])
    item = int(train_rating.iloc[i]['movieId'])
    score = float(train_rating.iloc[i]['rating'])

    user_id = np.where(all_user == user)[0][0]
    item_id = np.where(all_item == item)[0][0]

    rating_mat[user_id][item_id] = float(score)

```

请先按照上述说明，输入代码并得到对应的训练用的评分矩阵。。

3、 完成矩阵分解的梯度下降算法：

```
def matrix_factorization(rating_mat, embed_dim, gamma, lamda, steps):
    '''
    INPUT:
    :param rating_mat: [m, n]
    :param embed_dim: dimension of the embeddings
    :param gamma: learning rate
    :param lamda: balanced hyper parameters
    :param steps: training epoch
    RETURN:
    :param P: user embedding [m, embed_dim]
    :param Q: item embedding [embed_dim, n]
    :param error_list: list
    '''

    ######################
    #Please code for task 2.1 (matrix_factorization) here. Hint: The get_loss can be used in matrix_factorization.#

    ######################

    return P, Q, error_list
```

具体算法参考如下流程图：

The optimization scheme of Matrix Factorization with regularization term is given as:

Algorithm 1: Matrix Factorization with regularization

Input: $\boldsymbol{R}$: user/item rating matrix; λ: hyper parameters; γ: learning rate.
Output: $\boldsymbol{P}$: User embeddings; $\boldsymbol{Q}$: Item embeddings.
For epoch $= 1$ to T

Update $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \gamma \left[\sum_{j \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j) \boldsymbol{q}_j - \lambda \boldsymbol{p}_i \right]$.

Update $\boldsymbol{q}_j \leftarrow \boldsymbol{q}_j + \gamma \left[\sum_{i \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j) \boldsymbol{p}_i - \lambda \boldsymbol{q}_j \right]$.

End For

4、 记录矩阵分解梯度下降过程中 loss 下降的趋势并绘图。

```
def get_loss(rating_mat, embed_dim, lamda, P, Q):
    '''
    INPUT:
    :param rating_mat: [m, n]
    :param embed_dim: dimension of the embeddings
    :param gamma: learning rate
    :param P: user embedding [m, embed_dim]
    :param Q: item embedding [n, embed_dim]
    RETURN: the sum of the error.
    '''

    ######################
    #Please code for task 2.2 (get loss during the training on the training datasets) here#

    ######################

    return error

```

具体而言，就是给定 P 和 Q 的情况下，计算下述式子：

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \ell = \sum_{i,j \in S} (r_{ij} - \hat{r}_{ij})^2 = \sum_{i,j \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j)^2 + \underbrace{\lambda \left(\sum_{i \in S} ||\boldsymbol{p}_i||^2 + \sum_{j \in S} ||\boldsymbol{q}_j||^2 \right)}_{\text{regularization}}$$

接着可以利用 matplotlib 仓库，实现对 loss 记录并绘图。

```
def draw(error_list):
    plt.plot(range(len(error_list)), error_list)
    plt.xlabel("epoch")
    plt.ylabel("loss")
    plt.show()

```

5、 将训练好的用户和商品 embedding 进行测试评估。

```
55 def test(test_rating, P, Q, all_user, all_item):
56     '''
57     INPUT:
58     :param test_rating:
59     :param all_user: the all_user lists
60     :param all_item: the all_item lists
61     :param P: user embedding [m, embed_dim]
62     :param Q: item embedding [n, embed_dim]
63     RETURN: the mse and rmse on the test samples.
64     '''
65
66     ######################
67     #Please code for task 2.3 (evaluate the well-trained model on the test datasets) here#
68
69     ######################
70
71     return mse, rmse
72
```

6、 调试不同的λ。调试的范围为$\lambda = \{0.001, 0.01, 0.1, 1, 10\}$。同时，调试不同的学习率$\gamma$。调试的范围为$\gamma = \{0.0001, 0.001, 0.01\}$。记录下不同参数下的模型 loss 下降曲线和最终的测试 mse 的数值。

7、 调试不同的用户-商品嵌入维度 *embed_dim*。调试的范围为$embed_dim = \{8, 16, 32, 64, 128\}$。 记录下不同 *embed* 下的模型 loss 下降曲线和最终的测试 mse 的数值。

8、 给用户-商品加上非负的约束。在此约束下，实现矩阵分解建模。

该问腿的优化过程主要如下：

In practice, one can even adopt the **non-negative** constraints:

$$\min_{P,Q} \ell = \sum_{i,j \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j)^2 + \lambda \left(\sum_{i \in S} ||\boldsymbol{p}_i||^2 + \sum_{j \in S} ||\boldsymbol{q}_j||^2 \right) \quad (18)$$
$$s.t.\ p_{ij} \geq 0, \quad q_{ij} \geq 0$$

- Similarly, it can be updated as:

$$\boldsymbol{p}_i \leftarrow \max\left(0, \boldsymbol{p}_i + \gamma \left[\sum_{j \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j) \boldsymbol{q}_j - \lambda \boldsymbol{p}_i \right] \right)$$
$$\boldsymbol{q}_j \leftarrow \max\left(0, \boldsymbol{q}_j + \gamma \left[\sum_{i \in S} (r_{ij} - \boldsymbol{p}_i^\top \boldsymbol{q}_j) \boldsymbol{p}_i - \lambda \boldsymbol{q}_j \right] \right) \quad (19)$$

对应的代码填空部分为：

```
def non_negative_matrix_factorization(rating_mat, embed_dim, gamma, lamda, steps):
    '''
    INPUT:
    :param rating_mat: [m, n]
    :param embed_dim: dimension of the embeddings
    :param gamma: learning rate
    :param lamda: balanced hyper parameters
    :param steps: training epoch
    RETURN:
    :param P: user embedding [m, embed_dim]
    :param Q: item embedding [embed_dim, n]
    :param error_list: list
    '''

    ######################
    #Please code for task 5 (non_negative_matrix_factorization) here#

    ######################

    return P, Q, error_list

```

实际完成的效果大致如下：

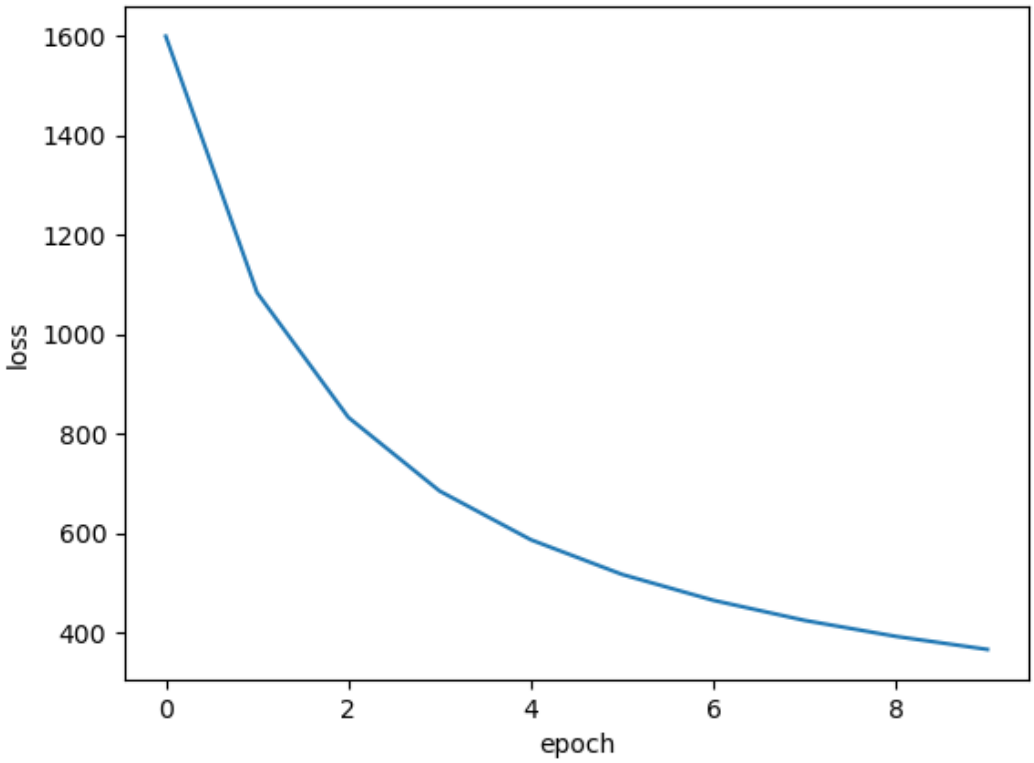
```
def train(rating_mat, all_user, all_item, embed_dim, gamma, lamda, steps):
    P, Q, error_list = matrix_factorization(rating_mat, embed_dim, gamma, lamda, steps)
    mse, rmse = test(test_rating, P, Q, all_user, all_item)

    print('Test MSE: ', mse, 'Test RMSE: ', rmse)
    draw(error_list)


'''
train(rating_mat, all_user, all_item, embed_dim = 32, gamma = 0.001, lamda = 0.01, steps = 20)
'''
```

开始训练：

```
Training step:  0 Loss on the training datasets:  1110.483774800249
Training step:  1 Loss on the training datasets:  699.7014358380587
Training step:  2 Loss on the training datasets:  527.2135875109719
Training step:  3 Loss on the training datasets:  433.06412216433273
Training step:  4 Loss on the training datasets:  374.09173172238985
Training step:  5 Loss on the training datasets:  333.85701004832123
Training step:  6 Loss on the training datasets:  304.72903085221236
Training step:  7 Loss on the training datasets:  282.68700641155897
Training step:  8 Loss on the training datasets:  265.418349433421
Training step:  9 Loss on the training datasets:  251.504694505153
Training step:  10 Loss on the training datasets:  240.03197958954442
Training step:  11 Loss on the training datasets:  230.38725895835069
Training step:  12 Loss on the training datasets:  222.14573040188253
Training step:  13 Loss on the training datasets:  215.0046093789323
Training step:  14 Loss on the training datasets:  208.74276776416104
Training step:  15 Loss on the training datasets:  203.19522003245618
Training step:  16 Loss on the training datasets:  198.236504759082
Training step:  17 Loss on the training datasets:  193.76957445626357
Training step:  18 Loss on the training datasets:  189.7181953573444
Training step:  19 Loss on the training datasets:  186.0216406734216
Test MSE:  18.58354658608209 Test RMSE:  4.310863786537692
```



提交作业要求：

1. 请在 task.py 中##和##之间添加代码，外部其他地方不要动。
2. 需要上传 task.py 文件，同时上传一份 pdf，用来记录相应的模型输出结果，以及对应的图表。
3. 请在学在浙大平台上提交。数据集无需上传。