華南理工大學

South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Shoukai Xu and Yaofu Chen

Supervisor:
Qingyao Wu

Student ID：
201530611180 and 20153061380
and 201530613535

Grade:
Undergraduate

December 9, 2017

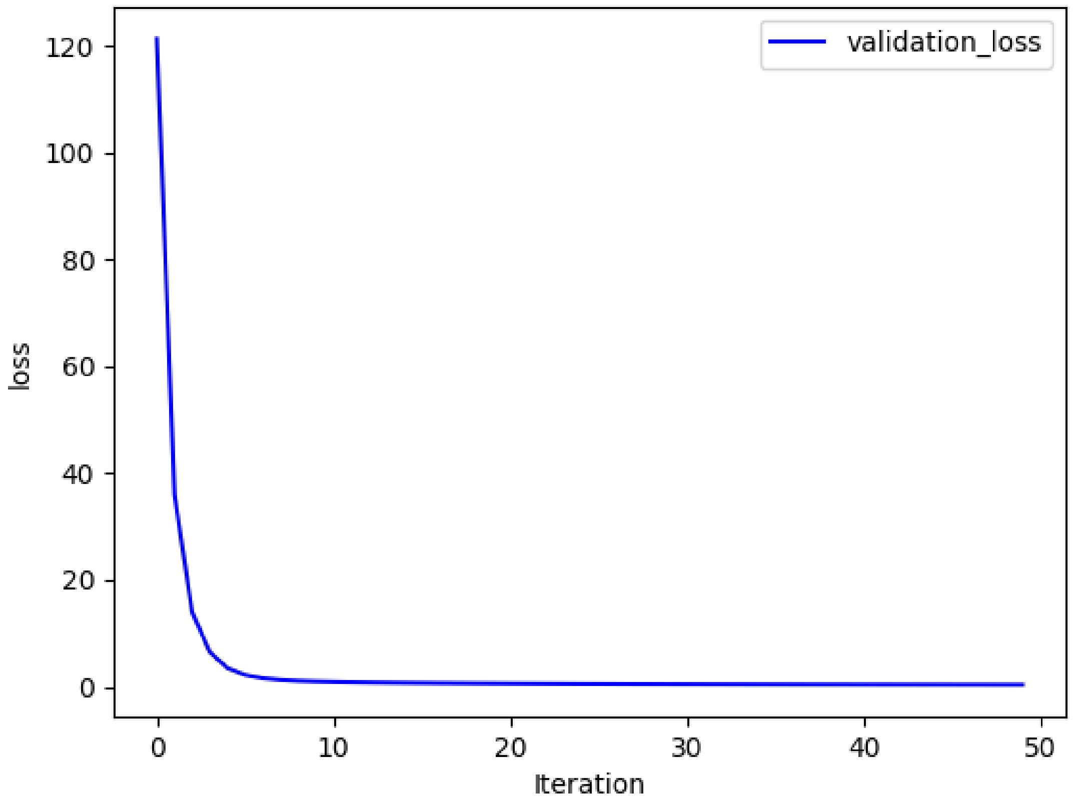
Recommender System Based on Matrix Decomposition

ABSTRACT—

Our project is based on matrix decomposition and we take several steps to complete the task:

- Read Data
- Spilt Data
- Initialization
- Training
- Draw The Loss Figure

Here is the brief look at the result:



I. INTRODUCTION

i. Motivation

a) Explore the construction of recommended system.
b) Understand the principle of matrix decomposition.
c) Be familiar to the use of gradient descent.
d) Construct a recommendation system under small-scale dataset, cultivate engineering ability.

ii. Dataset

a) Utilizing MovieLens-100k dataset.
b) u.data -- Consisting 10,000 comments from 943 users out of 1682 movies. At least, each user comment 20 videos. Users and movies are numbered consecutively from number 1 respectively. The data is sorted randomly
c) u1.base / u1.test are train set and validation set respectively, seperated from dataset u.data with proportion of 80% and 20%. It also make sense to train set and validation set from u1.base / u1.test to u5.base / u5.test.
d) You can also construct train set and validation set according to your own evaluation method.

iii. Environment

Python3， at least containing following python package: sklearn， numpy， jupyter， matplotlib. An advice is installing anaconda3 directly, which already contains the python package mentioned above.

II. METHOD AND THEORY

i. ALS

a) Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix Rnusers,ni.tems against the raw data, and fill 0 for null values.
b) Initialize the user factor matrix Pnusers,K and the item (movie) factor matrix Qnitem,K, where K is the number of potential features.
c) Determine the loss function and the hyperparameter learning rate ita and the penalty factor lamda.
d) Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
    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.
    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.
    3) Calculate the Lvalidation on the validation set, comparing with the Lvalidation of the previous iteration to determine if it has converged.

ii. SGD

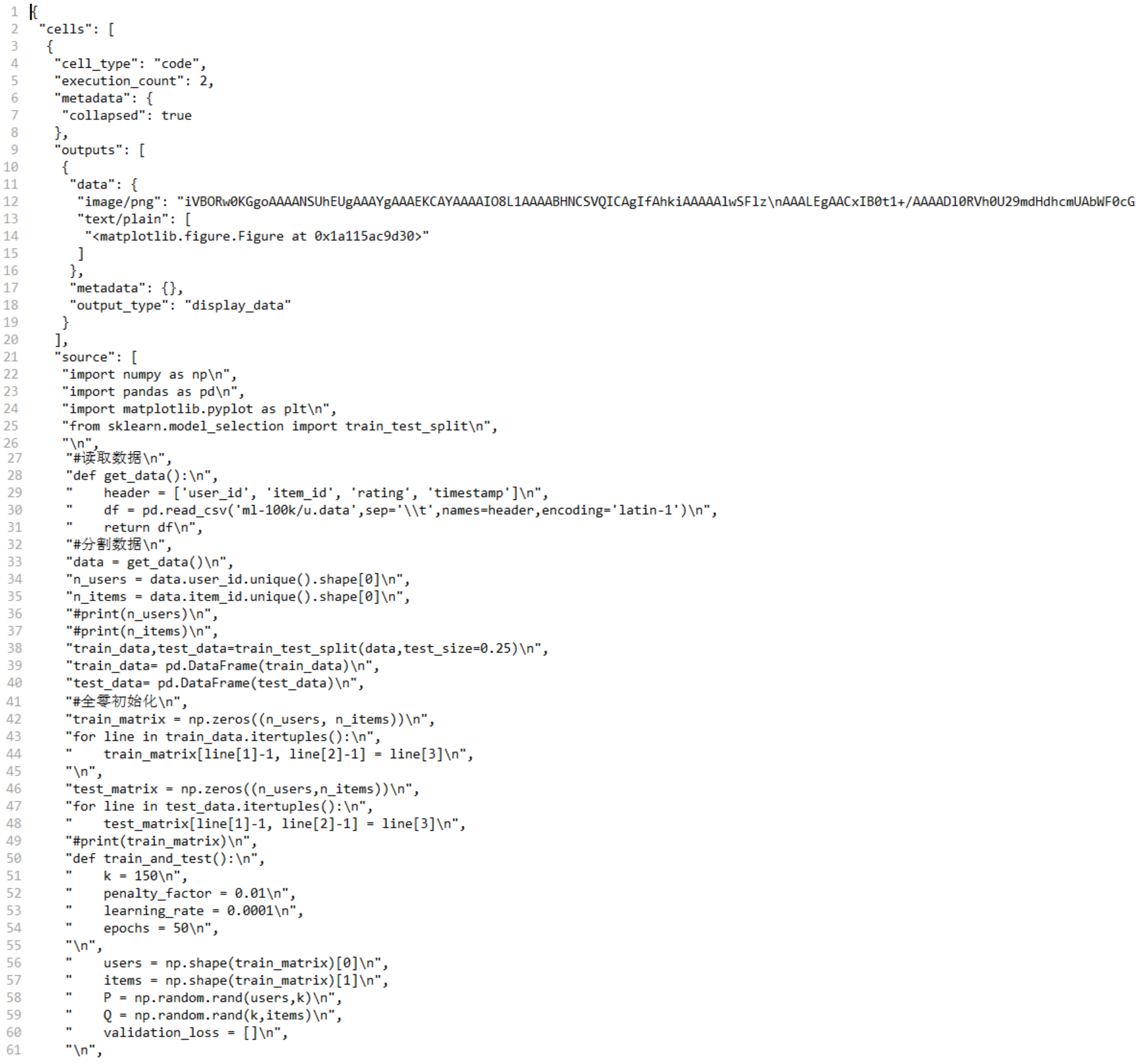
a) Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix Rnusers,ni.tems against the raw data, and fill 0 for null values.
b) Initialize the user factor matrix Pnusers,K and the item (movie) factor matrix Qnitem,K, where K is the number of potential features.
c) Determine the loss function and the hyperparameter learning rate ita and the penalty factor lamda.
d) Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
    1) S elect a sample from scoring matrix randomly;
    2) C alculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix;
    3) Use SGD to update the specific row(column) of Pn_users,K and Qn-items,K;
    4) Calculate the Lvalidation on the validation set, comparing with the Lvalidation of the previous iteration to determine if it has converged.

e) Repeat step 4. several times, get a satisfactory user factor matrix Q and an item factor matrix , Draw a Lvalidation curve with varying iterations.

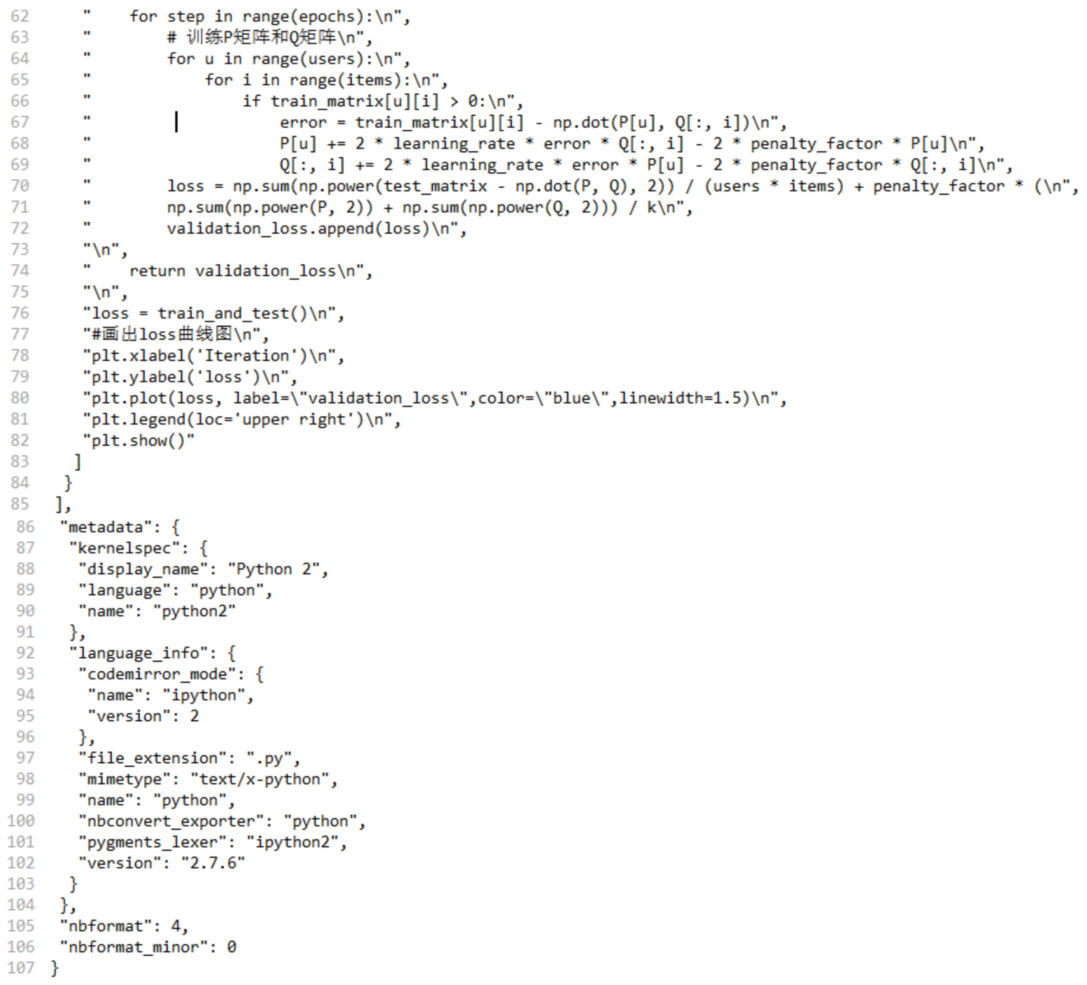
f) T he final score prediction matrix Rn_users,n_items is obtained by multiplying the user factor matrix Pn_users,K and the transpose of the item factor matrix Qn_items,K.

III. EXPERIMENT

i. Code



```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {
    "collapsed": true
   },
   "outputs": [
    {
     "data": {
      "image/png": "iVBORw0KGgoAAAANSUhEUgAAAYgAAAEKCAYAAAAIO8L1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlz\nAAALEgAACxIB0t1+/AAAADl0RVh0U29mdHdhcmUAbWF0cG
      "text/plain": [
       "<matplotlib.figure.Figure at 0x1a115ac9d30>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt\n",
    "from sklearn.model_selection import train_test_split\n",
    "\n",
    "#读取数据\n",
    "def get_data():\n",
    "    header = ['user_id', 'item_id', 'rating', 'timestamp']\n",
    "    df = pd.read_csv('ml-100k/u.data',sep='\\t',names=header,encoding='latin-1')\n",
    "    return df\n",
    "#分割数据\n",
    "data = get_data()\n",
    "n_users = data.user_id.unique().shape[0]\n",
    "n_items = data.item_id.unique().shape[0]\n",
    "#print(n_users)\n",
    "#print(n_items)\n",
    "train_data,test_data=train_test_split(data,test_size=0.25)\n",
    "train_data= pd.DataFrame(train_data)\n",
    "test_data= pd.DataFrame(test_data)\n",
    "#全零初始化\n",
    "train_matrix = np.zeros((n_users, n_items))\n",
    "for line in train_data.itertuples():\n",
    "    train_matrix[line[1]-1, line[2]-1] = line[3]\n",
    "\n",
    "test_matrix = np.zeros((n_users,n_items))\n",
    "for line in test_data.itertuples():\n",
    "    test_matrix[line[1]-1, line[2]-1] = line[3]\n",
    "#print(train_matrix)\n",
    "def train_and_test():\n",
    "    k = 150\n",
    "    penalty_factor = 0.01\n",
    "    learning_rate = 0.0001\n",
    "    epochs = 50\n",
    "\n",
    "    users = np.shape(train_matrix)[0]\n",
    "    items = np.shape(train_matrix)[1]\n",
    "    P = np.random.rand(users,k)\n",
    "    Q = np.random.rand(k,items)\n",
    "    validation_loss = []\n",
    "\n",
```



```
    "    for step in range(epochs):\n",
    "        # 训练P矩阵和Q矩阵\n",
    "        for u in range(users):\n",
    "            for i in range(items):\n",
    "                if train_matrix[u][i] > 0:\n",
    "                    error = train_matrix[u][i] - np.dot(P[u], Q[:, i])\n",
    "                    P[u] += 2 * learning_rate * error * Q[:, i] - 2 * penalty_factor * P[u]\n",
    "                    Q[:, i] += 2 * learning_rate * error * P[u] - 2 * penalty_factor * Q[:, i]\n",
    "        loss = np.sum(np.power(test_matrix - np.dot(P, Q), 2)) / (users * items) + penalty_factor * (\n",
    "        np.sum(np.power(P, 2)) + np.sum(np.power(Q, 2))) / k\n",
    "        validation_loss.append(loss)\n",
    "\n",
    "    return validation_loss\n",
    "\n",
    "loss = train_and_test()\n",
    "#画出loss曲线图\n",
    "plt.xlabel('Iteration')\n",
    "plt.ylabel('loss')\n",
    "plt.plot(loss, label=\"validation_loss\",color=\"blue\",linewidth=1.5)\n",
    "plt.legend(loc='upper right')\n",
    "plt.show()"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 2",
   "language": "python",
   "name": "python2"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 2
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython2",
   "version": "2.7.6"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 0
}
```

ii. Train of Thought

a) Content Based

For each item, you need to create a profile for the item. For each user, create a user profile. For example, if you search for programming and search for food, then your user profile will show that you are right Programming, food is interested. So if we treat the item profile simply as the question mark for each question then you are probably interested in programming or food labeling issues.

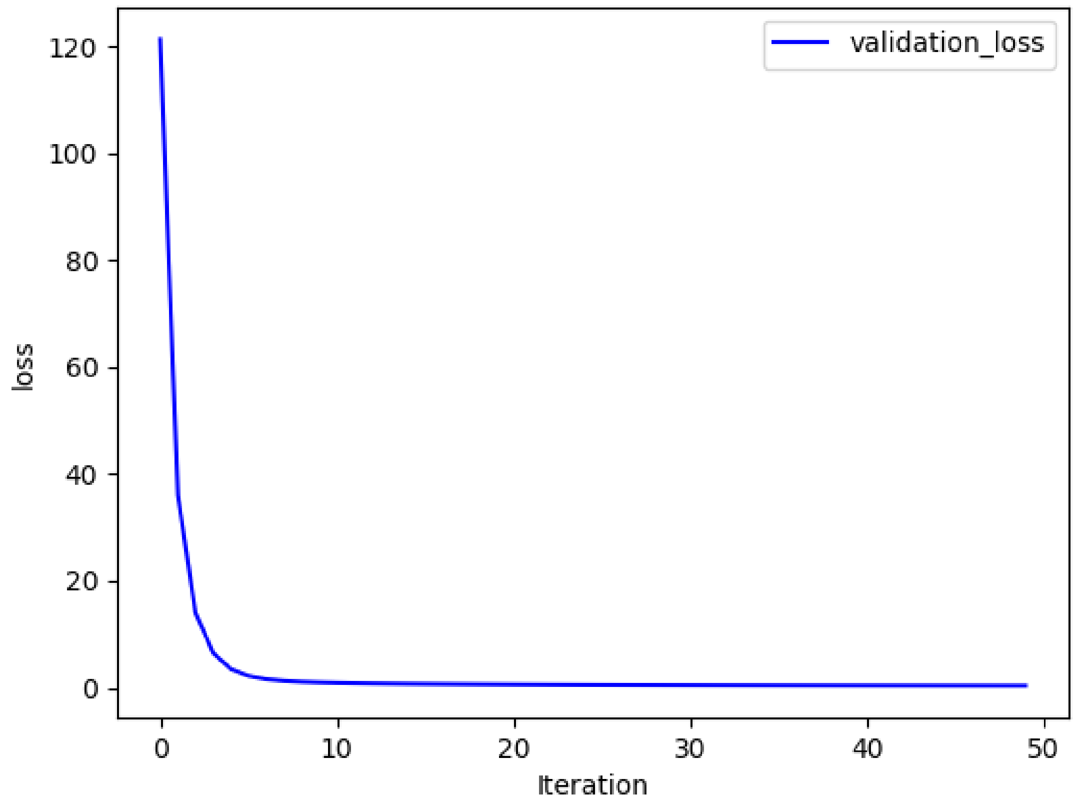
b) Collaborative Filtering

The essence of cf is that he speculates what x wants to buy using a user's buying habits similar to user x. The breakdown of the cf algorithm is user-based and item-based.

c) Matrix Factorization

We imagine a matrix M, where each row represents a user, and each column represents a movie. Then M [i] [j] is the rating of user i for movie j. It may or may not exist (if i has not seen j). Assuming there are m users, n movies, then this matrix is the m * n matrix. Now, let's break it down into the product of two matrices of (m * f) * (f * n). (F is the number of movie features) The first is called user matrix and the second is item matrix. So what is the representation of each row in the user matrix: the degree of user's preference for each feature. For example, if we define features as "comedy, horror, martial arts," then the user u may be <1.0, 0.2, -0.2 in the user matrix line, which illustrates u's favorite comedy, hate terror and hate martial arts. Then in the item matrix, each movie corresponds to the extent of their "contained" features. For example, a pancake man might be <1.0, 0, 0.2>, and a horror film might be <0, 1, 0.5>. Then we count the dot product of these two vectors and get M [u] 1-0.04 = 0.96 M [u] [Horror] = 0.2-0.1 = 0.1.

IV. CONCLUTION


validation_loss
loss
Iteration