CS315 Machine Learning Assignment2

Xi'an Jiaotong-Liverpool University

Author: Zuopeng Liu
ID:1406090

November 10, 2022

1 Introduction

This is CSE315 machine learning assignment2 report of xjtlu. It indculdes classification of digit and iris, anomaly detection and recommender systems. The data sets are mnist digital set, iris and data provided by Standford University. The algorithms are CNN, SVM, PCA, K-means and Gaussion distribution ect. And the tools includes tensorflow, sk-learn and matlab.

2 Implementation and Result

2.1 Task1

2.1.1 Problem Specification

This task is asked to implement two different classification algorithms for MNIST data set and achieve more than 90 percent accuracy. The algorithms should be explained and some other methods can improve should be discussed in the part.

2.1.2 Task1.1

For the first algorithm. The first algorithm is softmax classification, the description of layers are as shown. Digital classification is a multi-classification problem and softmax is the generalization case of logisticate regression to handle multiple class.
The result of softmax is the predication of result of cross entropy can be applied as loss function. In this case, there are 10 kinds of outputs and we want the output contains a one and nine 9 zero so we want entropy as less as possible.
The function of softmax is as shown below.

$$S_i = \frac{e_i^V}{\sum_i^C e_i^V} \tag{1}$$

i represent class and e is result of sampling.

$$H(p,q) = -\sum xp(x)log(x) \tag{2}$$

p and q are probability distribution of real output and module output
This module achieves 0.9044 accuracy and the calculation graphic is visualized by tensorboard and shown below. The inputs is the value of 28 multiply 28

pixels and the output is an array with length 10 for possibilities of 10 classes.

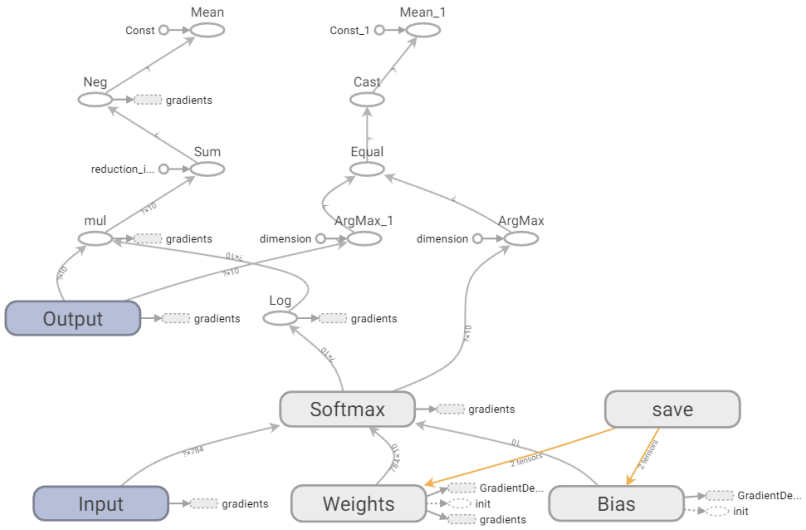

Figure 1: Softmax Calculation Graphics

For the second algorithm ,CNN is applied to do classification.The module consist of 7 layers to do sampling and a softmax classifier same with the first algorithm.

The structure of Variant LeNet	
Input Layer	28×28
Convolution layer1	5/5/1/32. The kernel size is 5×5. The depth is 1. There are 32 kernels. The sampling step is 1 for each dimensionality (batch, height, width, output channels).
Pooling layer1	Max-pooling for size 2×2 and it is padding to the input size of 28×28
Convolution layer2	5/5/32/64. The kernel size is $5 \times 5 \times 32$ and there are 64 kernels in this layer. The sampling step is also 1 for each dimensionality (batch, height, width, output channels).
Pooling layer	Max-pooling for size 2×2 with padding.
Flatten layer$\times 2$	The features are flaten into an array and the size is $7 \times 7 \times 64$ calculated by second convolution layer and pooling layer.
ReLu Perceptron Layer layer	There are 1024 perceptrons in this layer and the activation function is ReLu
Droup out layer	Some features are drouped to avoid over fitting.
Classifier	Softmax Classifier which introduced in algorithm one.

2.1.3 Task1.2

This task is asked to describe the techniques used in classification algorithm.
For the first algorithm, it applies softmax classification function and cross entropy loss function to classify the digits. The equations are mentioned before. The softmax algorithm calculates the possibility for different classes and cross entropy can be applied to be the loss function and the reason is mentioned before. The advantage of cross entropy loss function is to avoid learning rate decrease problem of squared error loss function. For square error loss function, differential value is reduce when the error close too 0.
For the second algorithm, it to do classification by CNN. The techniques will discuss next.
ReLu activation is applied in this module. The equation is as shown.

$$g(x) = max(0, x) = \begin{cases} x & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \tag{3}$$

As the equation shown, the ReLu has sparse representation. When the negative input comes in an neur, the neur keep output 0 which is good to avoid over fitting and can help to find the best global minimum.
Drop out features, when the over fitting occur, deorp out some feature is helpful to find global minimum. However, this method increase the train time very much which should be careful to use.
Initialization of weights. By visualize the weights as shown below, Gaussion distribution with a very small standard deviation is the best choice to initialize the weights.

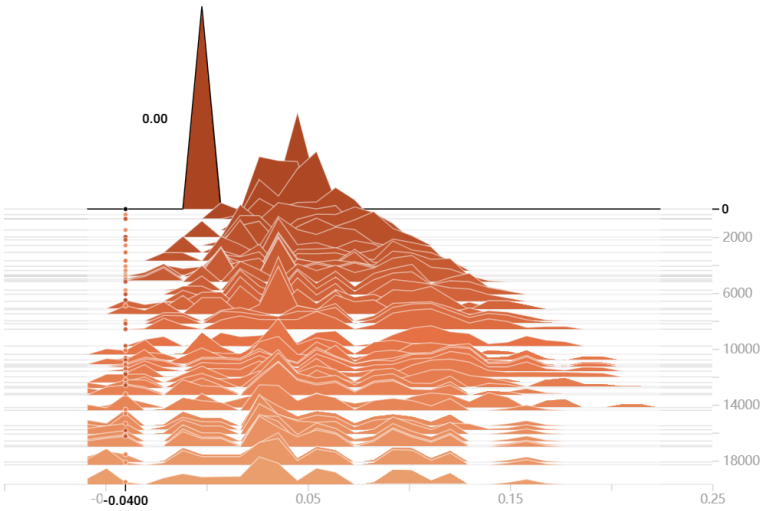

Figure 2: Convolution1 Weights Updates

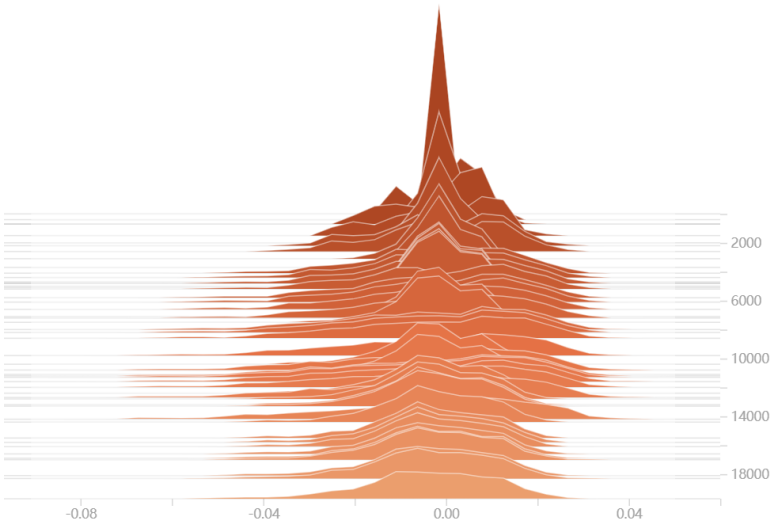

Figure 3: Convolution2 Weights Updates

Convolution Layer. There are many kernels in convolution layer. Each kernel extracts a feature. The kernel has a size and maps the size of the whole receptive fields to a number and then general a new feature map. In this case, the kernel is 5×5 and each kernel percepts the whole inputs. As mentioned, the size of input picture is 28×28 so, each kernel generate a 25×25 feature picture.
Puddling. For feature maps are puddled to the size of input picture to make the nerual network easy to bulid and understand.
Pooling. Pooling layers also contains kernel with a size. But it do not use duplicate pixels. So, for this case, the pooling size is 4×4 and the output size is 7×7 and the feature becaume 28×28 after puddling. The pooling is applied to remove duplicate features.

2.1.4 Task1.3

Other methods to improve the performance includes data normalization, ELU activation function. Date normalization maps inputs to a small value. The advantage is to avoid the values in different denotations are too significant different. Too much difference, makes it hard to train and find out a good result. ELU is a function to avoid dying proble of ReLu. The equation is as shown.

$$g(x) = max(0, x) = \begin{cases} x & if\ x \geq 0 \\ \alpha(exp(h_k) - 1) & if\ x < 0 \end{cases} \tag{4}$$

As equation, even weighted inputs is negative, the node also can output a small value which may helpful to find global minimum. $Task2$

2.1.5 Problem Specification

This problem is asked to classify iris data set by SVM and PCA and achieve more than 90 percent accuracy.

2.1.6 Task 2.1

This section is asked to classify iris by SVM. There are three classes and 50 examples for each class. It is separated into train set and validation set by 7 /3 so, there are 15 test samples for each class.
SVM is an algorithm to do linear classification. It can find decision boundary with same distance to different classes. And it can be used to solve linear indivisible problem by mapping the data set to high dimension.
To make it better, the gaussian kernel is applied. And it reaches 0.9788888888 accuracy. There is only one error. The code is as shown.

```
#print(__doc__)
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
```

```
import tensorflow as tf
from sklearn import svm,datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
print(iris.data)

X=iris.data
y=iris.target
x_train, x_test, y_train, y_test = train_test_split(
       X, y, test_size=0.3, random_state=5)

clf = svm.SVR(kernel='rbf', gamma=0.7, C=1.0)
res_test=clf.fit(x_train, y_train).predict(x_test)
res=y_test-res_test
ress=[]
for i in range(len(res)):
    if res[i] <0.5 and res[i]>-0.5:
        ress.append(0)
    else:
        ress.append(1)
error_number=0
for i in range(len(ress)):
    if ress[i]==1:
        error_number=error_number+1

error_rate=error_number/len(ress)
```

3 Task2.2

This section asks to find the first three main components of iris data set. It is implement by sk-learn library and the core code is as shown.

```
x_train_pca=pca.fit(x_train).transform(x_train)
x_test_pca=pca.fit(x_train).transform(x_test)
```

The values is as shown below.

```
In [43]: first_component
Out[43]:
[1.2489231500891484,
 0.502398771341l026,
 2.26818106452979 4,
 3.483957550169047,
 -2.3168694681953586,
 1.516742179325 9035,
 -2.8743946162401657,
 -2.82985102190342 8,
 -2.3880865910181246,
 -2.646535712850177,
```

Figure 4: PCA Result

They plot of first three dimension is as shown.

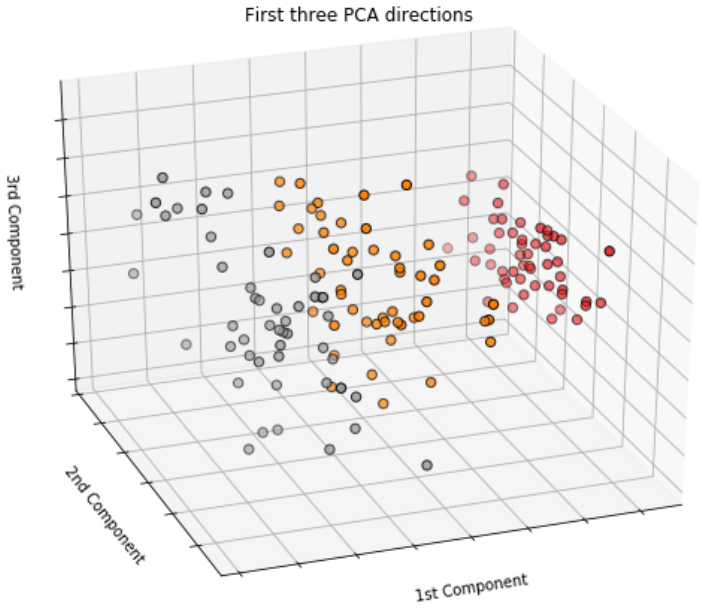

Figure 5: PCA Plot

3.0.1 Task2.3

This section is asked to do SVM separately on first three principl components of iris data set. A class is implemented to help. The code is as shown.

```
class SVM_Classifier(object):
    def __init__(self,kernal_,x_train_,y_train_,x_test_,y_test_):
        clf_ = svm.SVR(kernel=kernal_, gamma=0.7, C=1.0)
        self.res_row_= clf_.fit(x_train_, y_train_).predict(x_test_)
        res_=y_test_-self.res_row_
        ress_i_=[]
        for i in range(len(res_)):
            if res_[i] <0.5 and res_[i]>-0.5:
                ress_i_.append(0)
            else:
                ress_i_.append(1)
        self.ress_=ress_i_
        error_number_i_=0
        for i in range(len(ress_i_)):
            if ress_i_[i]==1:
                error_number_i_=error_number_i_+1
        self.error_number_=error_number_i_
        self.error_rate_=error_number_i_/len(y_test_)
        self.accuracy_=1-self.error_rate_
```

There are three attributes in this class. Attribute ress_ is the result of classification. Attribute error_number_ and error_rate_ are number and rate of error.

```
In [47]: res_first_rbf.error_rate_
Out[47]: 0.022222222222222223

In [48]: res_second_rbf.error_rate_
    ...:
Out[48]: 0.6444444444444445

In [49]: res_third_rbf.error_rate_
Out[49]: 0.622222222222222
```

Figure 6: Error rate

The variance ratio of first three component are 0.920, 0.058 and 0.016. The first is much bigger than another two which means it holds more information and it achieves much better.

3.0.2 Task4

This section is ask to combination of components; train the model and compare accuracy. In this section, a combination function is implemented to do selection. And the result is as shown.

Combination	Error rate
123	0.022
12	0.022
23	0.644
13	0.022

3.1 Task3

3.1.1 Problem Specification

This section is asked to do k-means clustering on iris data set and first three priciple comonents of iris data set.

3.1.2 Task3.1

This section asks to do k-means clustering on iris. K-means clustering is an unsupervised learning algorithm. It cluster the data by distance to the center and update the center by finding the center of clusters. It iterates until converge which means centering do not change anymore.
The core code is as shown.

```
for i in range(6):
    y_pred = KMeans(n_clusters=i+1, random_state=4).fit_predict(x)
```

First two dimension and some results is visualized as shown below.The accuracy of 3 cluster is 0.32 which is very bad.

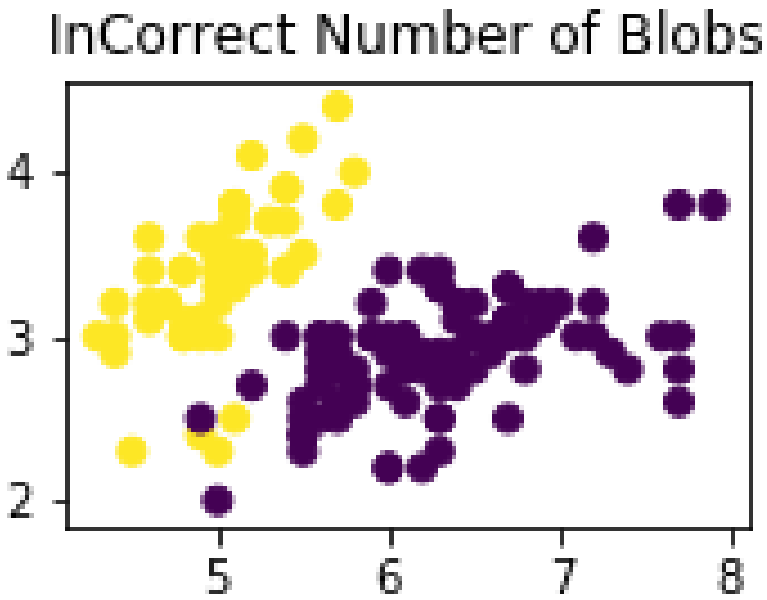

Figure 7: 2 Clusters

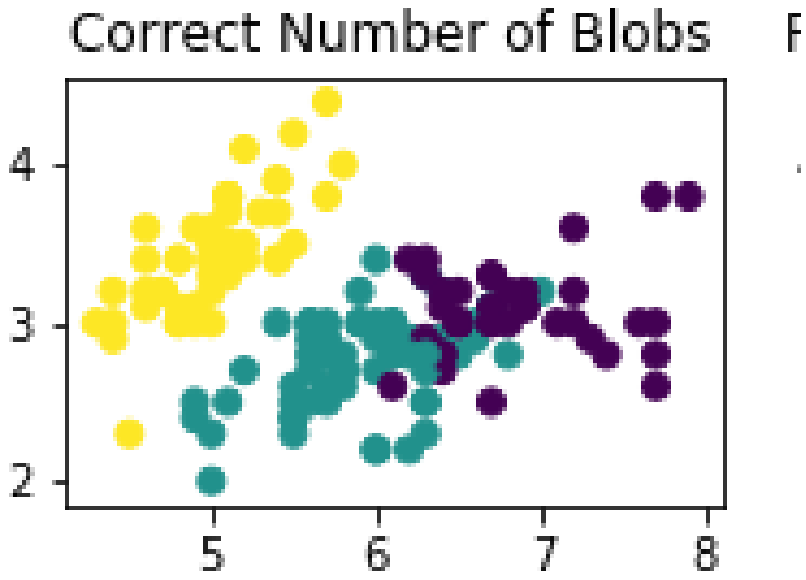

Figure 8: 3 Clusters

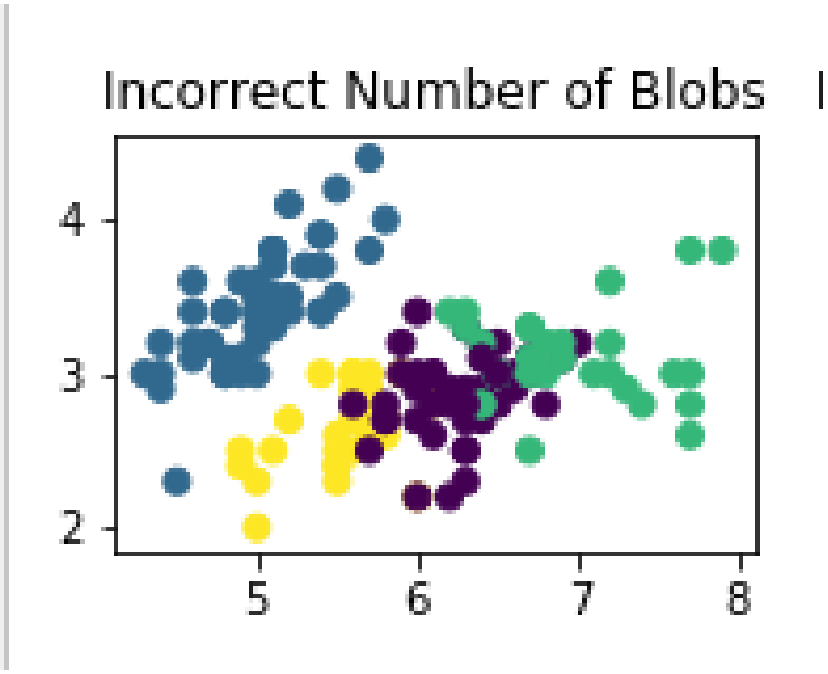

Figure 9: 4 Clusters

3.1.3 Task3.2

This section is asked to do k-means on first three principle components. The core code is as shown.

```
pca = PCA(n_components=3)
for i in range(6):
    y_PAC_pred = KMeans(n_clusters=i+1,
        random_state=4).fit_predict(x_pca)
```

The visualization is as shown.

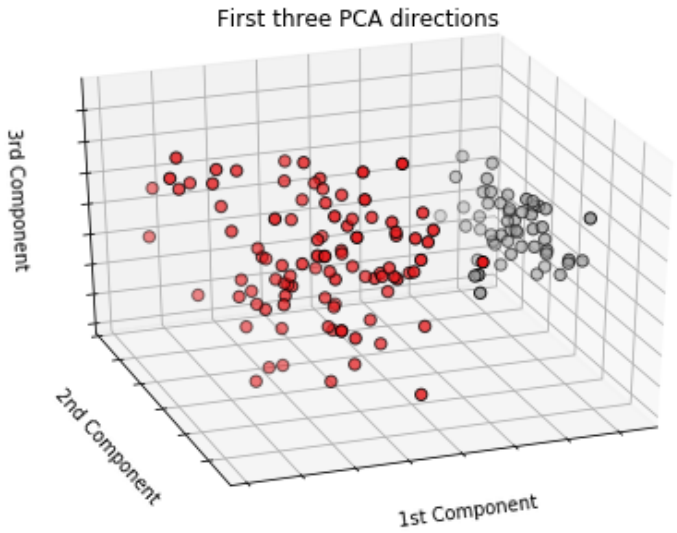

Figure 10: PCA 2 Clusters

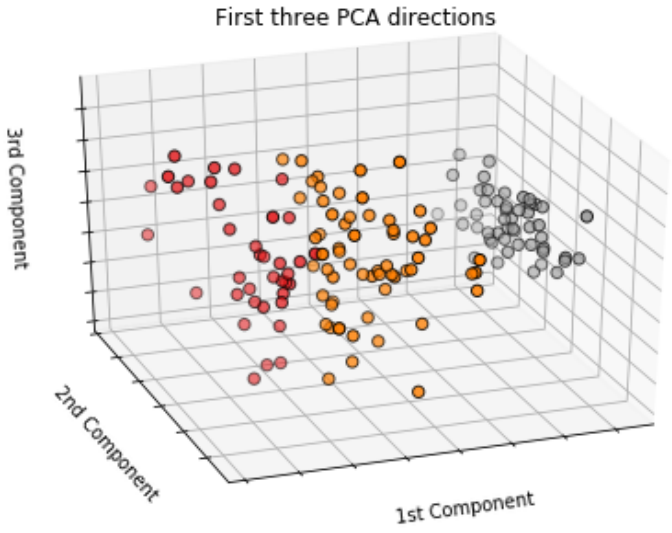

Figure 11: PCA 3 Clusters

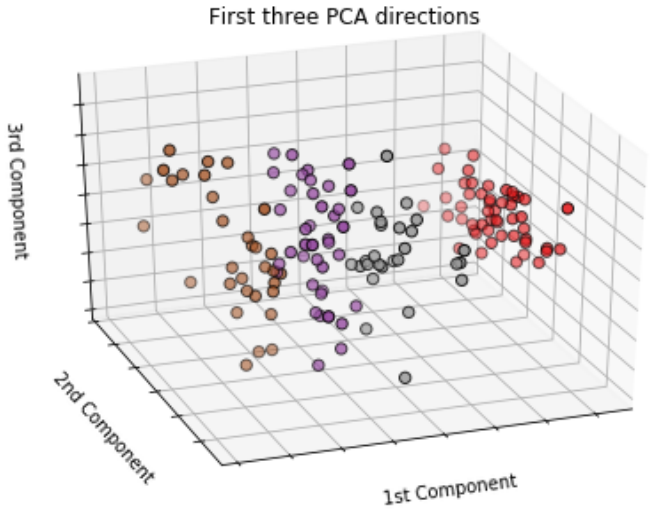

Figure 12: PCA 4 Clusters

3.2 Task4

This task is asked to implement exercise8 Stanford ML course published on coursera. There are two parts of exercise.
For the first part is about the annotation detection. There are many files should be implemented. For estimatiegaussian file, mean and variance should be calculated. The core code is as shown. mu and sigma2 are mean and standard

deviation.

```
% ====================== YOUR CODE HERE
mu=1/m*sum(X);
sigma2=1/m*sum((X-repmat(mu,m,1)).^2);
```

For thresholdselection file. The core code is as shown.

```
    predictions = (pval < epsilon);
    fp = sum((predictions == 1) & (yval == 0));
    fn = sum((predictions == 0) & (yval == 1));
    tp = sum((predictions == 1) & (yval == 1));

    prec = tp / (tp + fp);
    rec = tp / (tp + fn);

    F1 = 2 * prec * rec / (prec + rec);
```

For the second part, a recommend system should be implemented. For cofiCostFunc.m file, regulation loss function and gradient of bias and weights should be implemented. The core code is as shown.

```
J =
    (1/2).*sum(sum(((X*Theta').*R-Y.*R).^2))+(lambda./2.*sum(sum(Theta.^2)))+(lambda./2.*sum(sum(X.^
% Only predict rating X*Theta' if user has rated (i.e. R=1)

X_grad = (((X*Theta').*R*Theta-Y.*R*Theta)+lambda.*X);
Theta_grad = ((X'*((X*Theta').*R)-X'*(Y.*R)))'+lambda.*Theta;
```

The result is as shown.

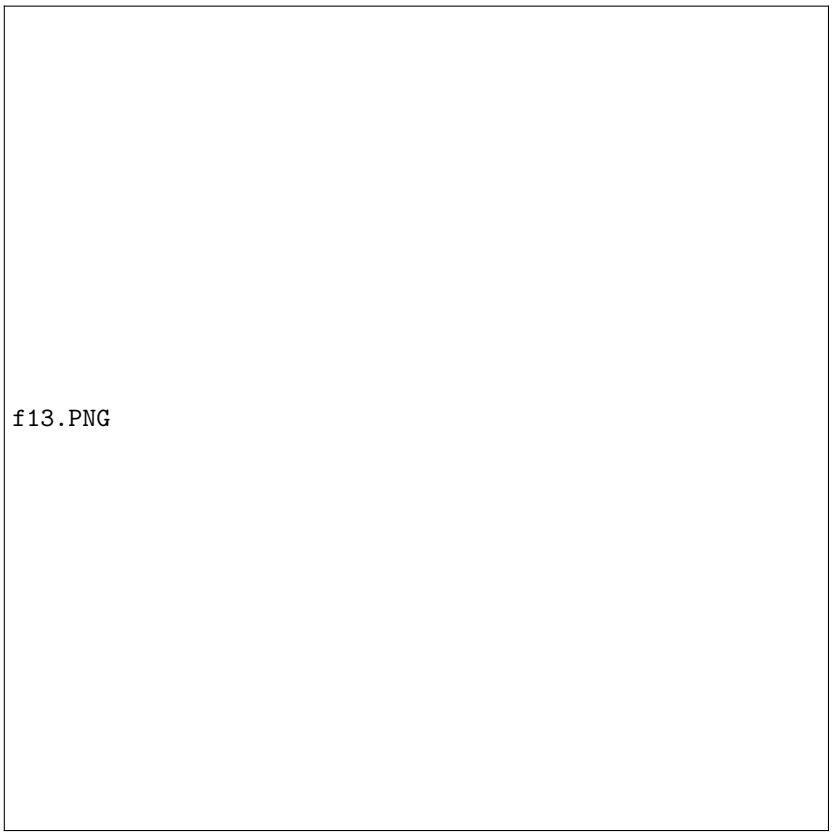

Figure 13: Recommend system result

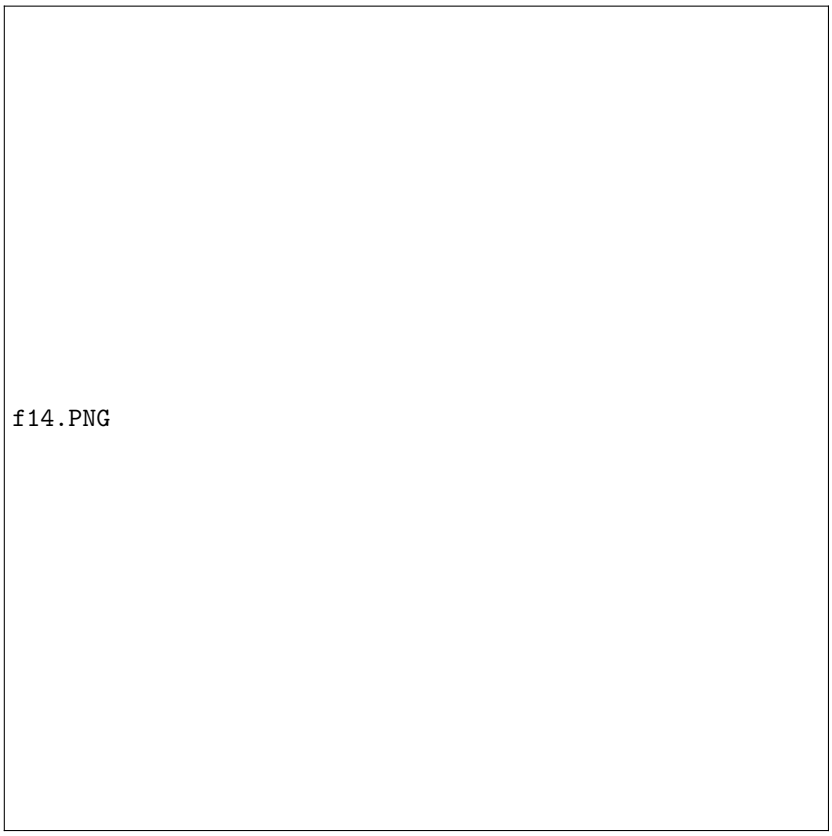

Figure 14: Recommand system result