Central University of Finance and Economics

中央财经大学

探索性数据分析课程

图像数据降维分析报告

吴宇翀

2021210793

email@wuyuchong.com

指导老师：刘苗

2021 年 11 月 26 日

报告概述

使用服饰图像分类数据集，我们探究降维对于图像识别的有效性。

未降维时变量数量过多，我们发现只有当模型参数数量充足时，才能解决拟合不足问题。且深度学习和随机森林尽管总体准确率达到 80% 以上，但品类间预测效果差距较大，部分品类准确率较低。

在使用 NMF 和 PCA 进行降维之后，降维对特征的提取效果较好，可较大程度地加快模型的训练速度，增强模型的收敛效果，各类准确率达到 90% 以上。

对错判结果分析后我们发现，图像识别主要依靠图像中的形状特征，因此有些在人类看来完全不同类的物品，模型的错判率反而比同类物品高。

1 数据集介绍

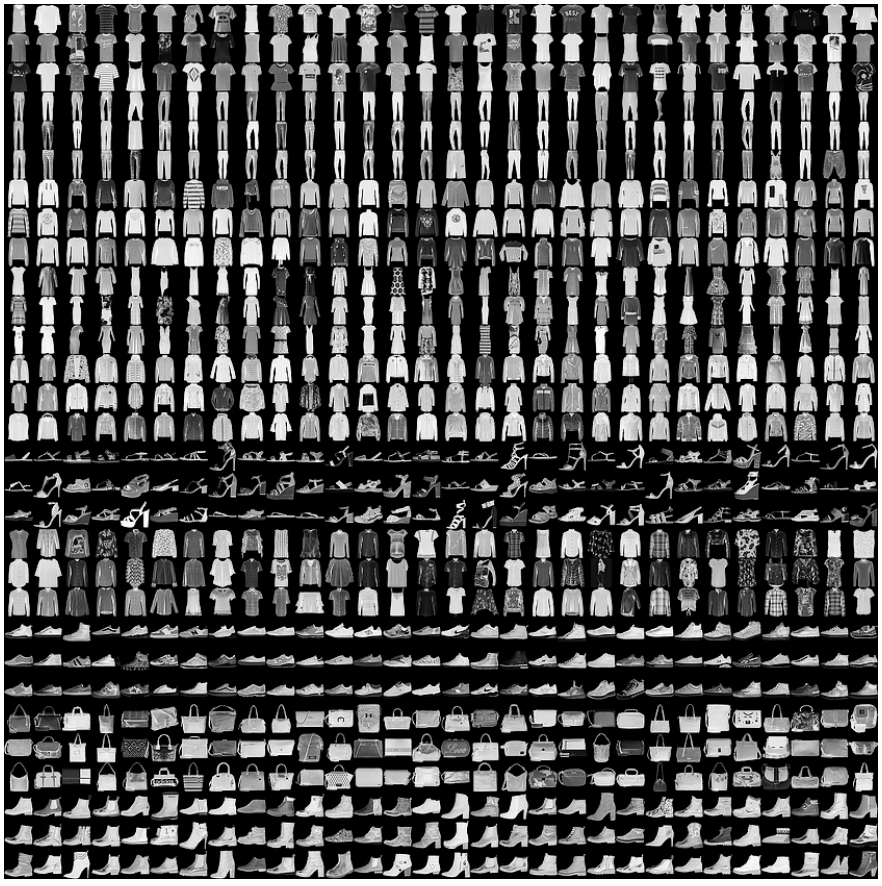

图 1: 数据集概览

我们使用 `Fashion-MNIST` 数据集[1]，它共有 6 万条训练样本和 1 万条测试样本，图片像素为 28 X 28，颜色为灰度。这些图片为服饰的照片，共分为 10 类。[1]

[1] https://github.com/zalandoresearch/fashion-mnist

表 1: 标签与分类名称

标签	0	1	2	3	4	5	6	7	8	9
名称	T 恤/上衣	裤子	套头衫	连衣裙	外套	凉鞋	衬衫	运动鞋	包	短靴

由于我们的训练样本和测试样本中，每个类别的样本数量相同，是一个平衡的数据集，因此我们选用准确率作为我们的模型评价指标。

使用 R 画出一个示例图像如下：

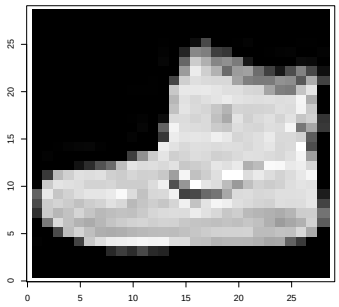

图 2: 图像示例

在数据预处理中，我们使用标准化对灰度值进行处理，以提高模型收敛的速度和效果。

$$x_i' = \frac{x_i - (\max(x) - \min(x))/2}{(\max(x) - \min(x))/2}$$

2 未降维

在未降维的情况下，我们建立模型作为基准。

2.1 NNET

我们使用 NNET [2] 建立一个较为简单的神经网络模型，以观察模型参数个数与预测准确率的关系。[2]

表 2: 神经网络模型的准确率

网络大小	1	2	3	4	5	6	7	8	9	10
准确率	20%	28%	38%	70%	64%	78%	72%	82%	80%	82%

[2]NNET 用时约 1 小时（阿里云服务器 Xeon 8 核 CPU 32G 内存）

随着网络大小的增加，预测准确率逐渐上升。这是因为由于未降维时变量数量过多，神经网络的大小不得不设置得非常大，模型才能被充分训练。

2.2 深度学习

我们使用 python 在 `TensorFlow` 框架下搭建一个多层神经网络。[3] 我们在中间层使用 Relu 作为激活函数，使用 Adam 作为优化器。[3]

由于我们进行分类任务，需要约束输出的十类的预测概率值之和为 1，因此我们在输出层使用 Softmax 函数：

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

表 3: 搭建的神经网络结构

神经网络层	神经元个数
输入层	784
drop out (20%)	0
中间层 1	128
drop out (50%)	0
中间层 2	128
drop out (50%)	0
输出层	10

[3] 未降维的 python 深度学习训练用时约 4 分钟（阿里云服务器 Xeon 8 核 CPU 32G 内存）

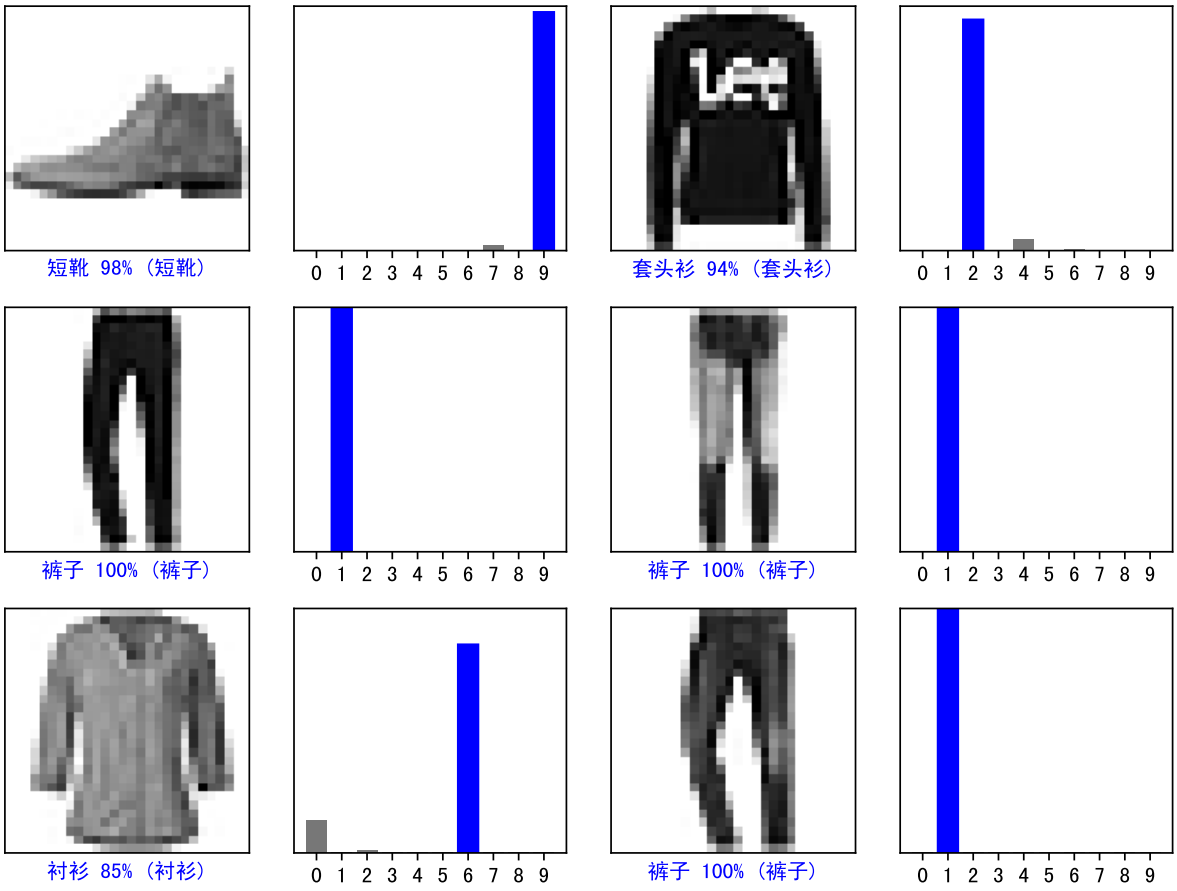

图 3: 深度学习预测结果

在未降维时，使用深度学习进行识别的正确率约为 85%。对于一些较为相似的品类，模型可能会发生错判，如上图中的衬衫，虽然最终预测是正确的，模型认为有 14% 的概率为 T 恤。

2.3 随机森林模型

搭建一个树的数目取 10 的随机森林模型 [4][4]

[4]未降维的随机森林模型训练用时约 40 分钟（阿里云服务器 Xeon 8 核 CPU 32G 内存）

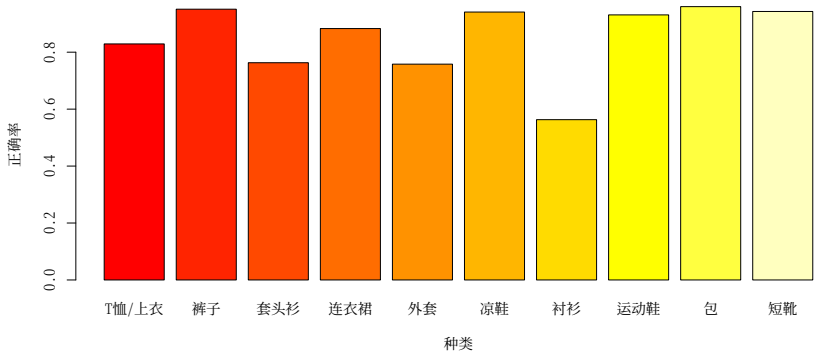

图 4: 随机森林不同类别预测正确率

对于此未降维的模型，部分品类的预测准确率较高，然而衬衫的准确率很低，仅有约 50%。

3 降维

由于未降维时模型的变量过多，收敛速度慢且收敛效果不佳，需要更多参数、更长时间的训练才能达到充分拟合。所以我们尝试使用降维方法对图像进行特征提取，之后再使用随机森林模型进行训练及预测。

3.1 降维后特征

我们使用 NMF [5] 将数据集降维至 12 维，降维后的新变量反映了原数据集中各个类别的特征。[5]

[5] 降维用时约 20 分钟（阿里云服务器 Xeon 8 核 CPU 32G 内存）

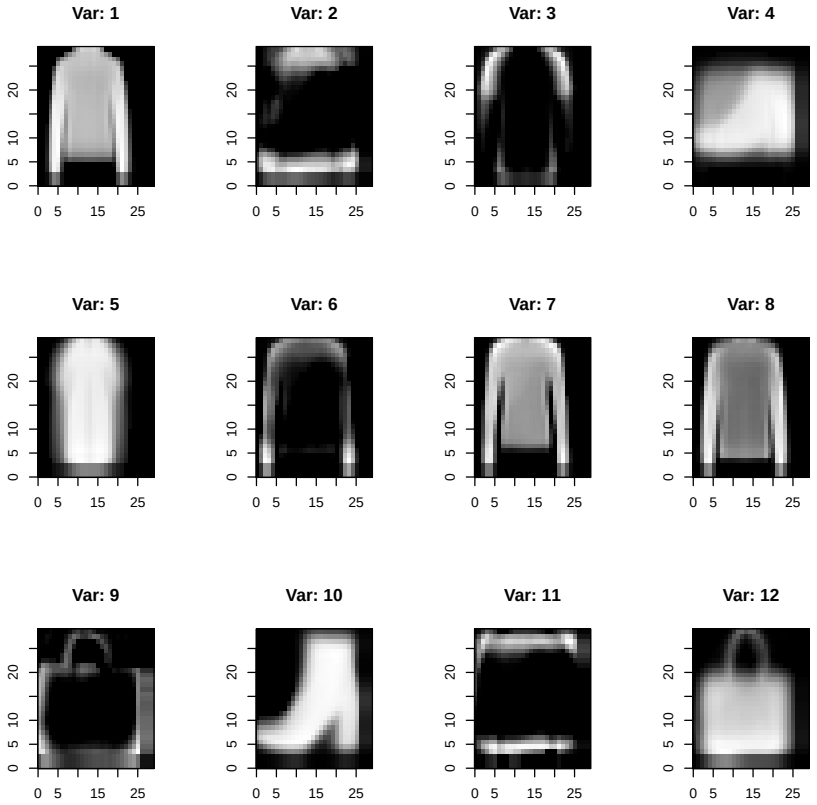

图 5: 降维后的新变量

表 4: 不同类别在前六个新变量中的权重

权重排名	变量.1	变量.2	变量.3	变量.4	变量.5	变量.6
1	外套	套头衫	T 恤/上衣	包	T 恤/上衣	套头衫
2	套头衫	包	衬衫	短靴	外套	外套
3	衬衫	外套	套头衫	套头衫	衬衫	衬衫
4	裤子	衬衫	包	衬衫	连衣裙	裤子
5	包	T 恤/上衣	裤子	外套	套头衫	T 恤/上衣
6	T 恤/上衣	连衣裙	外套	运动鞋	裤子	包
7	连衣裙	凉鞋	连衣裙	T 恤/上衣	包	凉鞋
8	凉鞋	裤子	凉鞋	凉鞋	短靴	连衣裙
9	短靴	短靴	短靴	裤子	凉鞋	短靴
10	运动鞋	运动鞋	运动鞋	连衣裙	运动鞋	运动鞋

3.2 PCA 和 NMF 降维效果对比

我们分别使用 PCA 和 NMF 降低不同的维度，应用于随机森林模型，观察预测效果。[6]

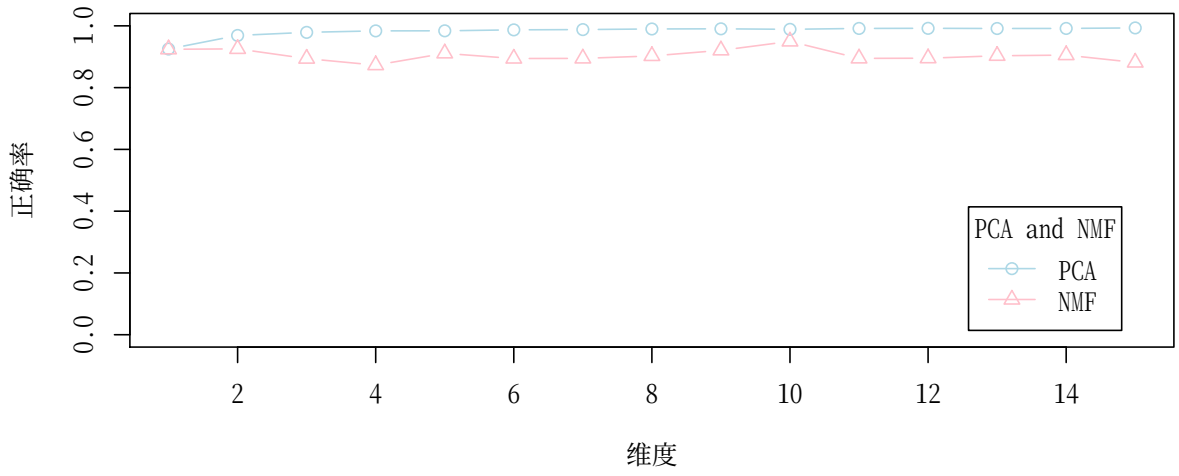

图 6: PCA 及 NMF 在不同维数预测的正确率线图

可以看出，由于我们的训练样本达到 6 万条，PCA 和 NMF 在降至 1 维时，就已经可以提供超过 90% 的准确率。

随着维度的增加，PCA 降维方法的准确率有所升高，不过提升幅度不大。而对于 NMF 方法，维度增加准确率波动较为明显，甚至会有所下降，这可能与维度增加后模型拟合不足、收敛速度变慢有关。

[6]降维后模型训练速度快于未降维，但由于训练不同降维方法、不同降维维度，总用时约 4 小时（阿里云服务器 Xeon 8 核 CPU 32G 内存）

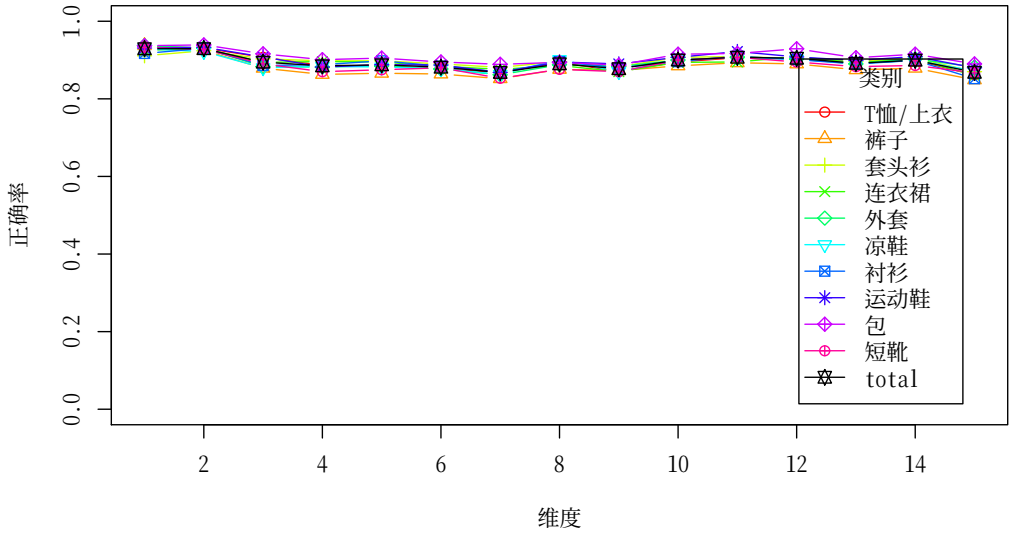

图 7: NMF 不同类别不同降维度的预测正确率

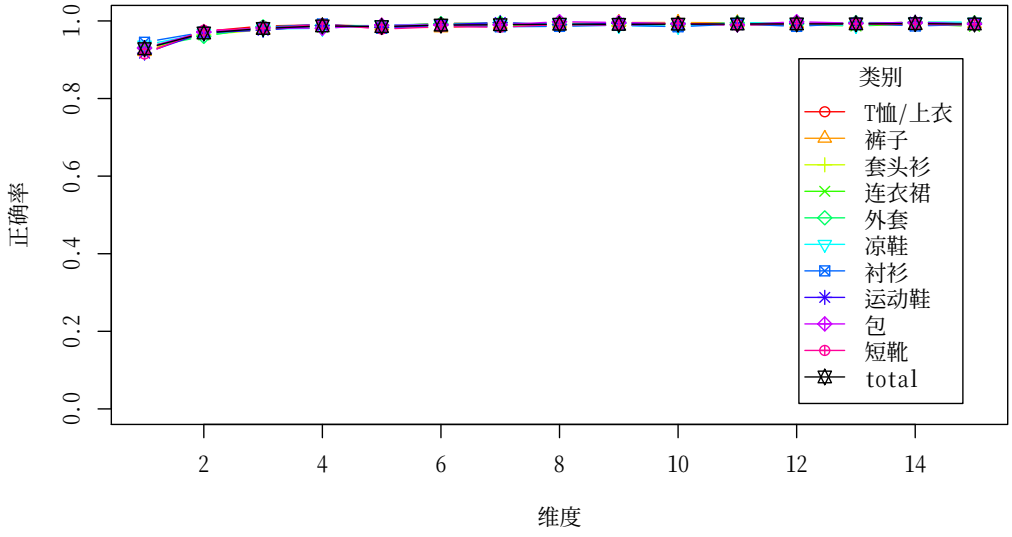

图 8: PCA 不同类别不同降维度的预测正确率

对 PCA 降维方法，不同类别的准确率差距非常小。对 NMF 降维方法，裤子的错判率最高，包的准确率最高。

4 错判结果分析

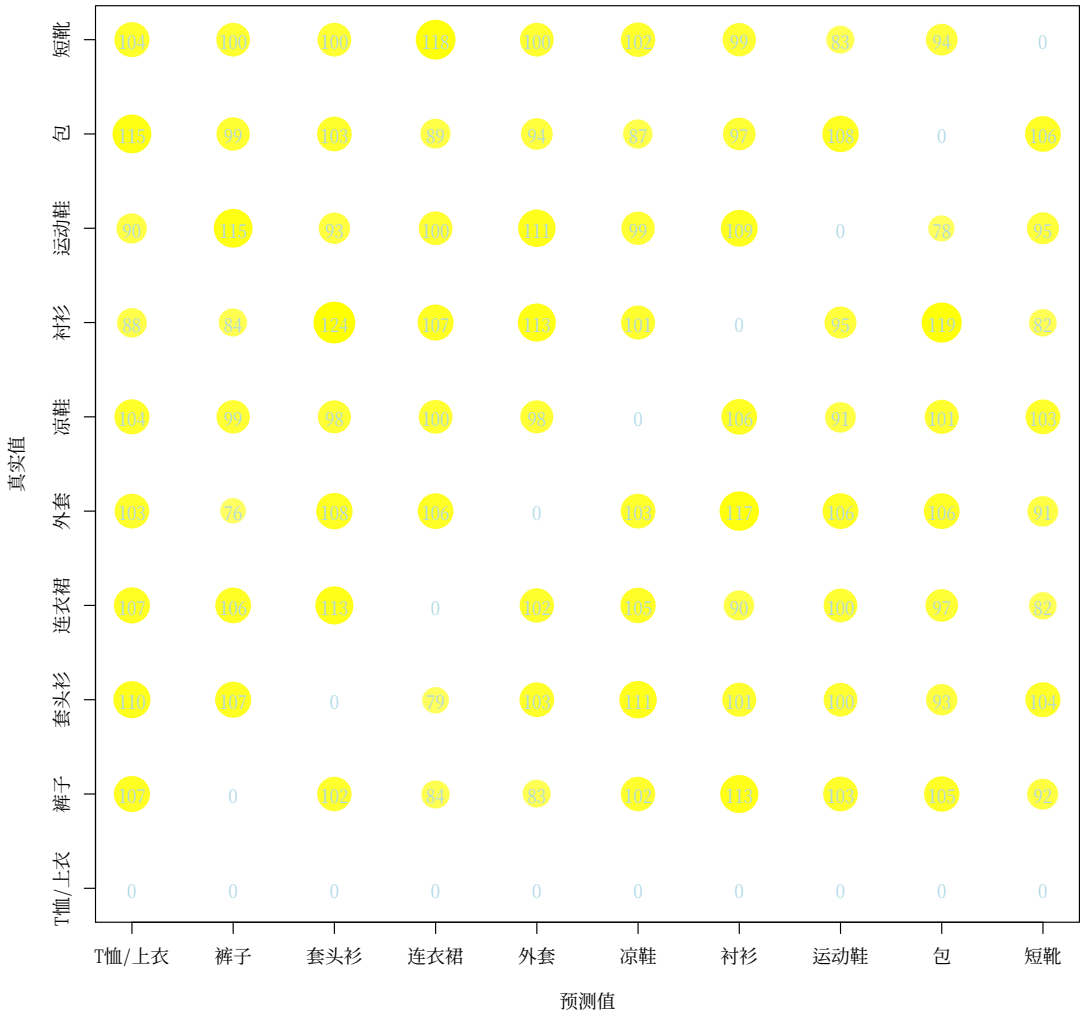

图 9: 6 维 NMF 降维随机森林错判图

对外套图像，117 次模型错判为衬衫，但只有 76 次错判为裤子，符合预期。

对衬衫图像，124 次模型错判为套头衫，但只有 88 次错判为 T 恤，这与我们的直觉相符合，套头衫由于与衬衫都为长袖，错判率较高。

对短靴图像，尽管直觉上我们猜测可能易与运动鞋混淆，然而事实上错判成为运动鞋的次数并不多，原因是尽管短靴和运动鞋都属于鞋类，然而它们在脚跟处的形状却有着很大的差别，故预测时混淆次数较少。同样的我们发现，尽管衬衫和包在人类看来是两类完全不同的东西，然而在图像识别时，两者的整体轮廓较为相似，故错判率较高。

5 总结

1. 在不降维的情况下，我们发现由于未降维时变量数量过多，神经网络的大小不得不设置得非常大，模型才能被充分训练。且深度学习和随机森林的结果表明，尽管某些品类预测准确率达到 80% 以上，但仍有品类预测效果较差。
2. 在使用 NMF 和 PCA 进行降维之后，降维对特征的提取效果较好，可较大程度地加快模型的训练速度，增强模型的收敛效果。
3. 随着维度的增加，PCA 降维方法的准确率有所升高，不过提升幅度不大。而对于 NMF 方法，维度增加准确率波动较为明显，甚至会有所下降，这可能与维度增加后模型拟合不足、收敛速度变慢有关。
4. 对错判结果分析后我们发现，大部分情况下同类物品容易产生混淆，与我们的预期相一致。但有些在人类看来完全不同类的物品，模型的错判率反而比同类物品高，这是由于图像识别主要依靠图像中的形状特征，如衬衫和包尽管类别不同，但其图像外形轮廓较为相似。

6 参考文献

[1] XIAO H, RASUL K, VOLLGRAF R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms[EB/OL]. (2017-08-28). http://arxiv.org/abs/cs.LG/1708.07747.

[2] VENABLES W N, RIPLEY B D. Modern Applied Statistics with S[M]. 第 Fourth 版. New York: Springer, 2002.

[3] ABADI M, AGARWAL A, BARHAM P, 等. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems[J]. 2015.

[4] LIAW A, WIENER M. Classification and Regression by randomForest[J]. R News, 2002, 2(3): 18–22.

[5] 刘苗. Algorithms for Non-negative Matrix Factorization[EB/OL]. (2021-11-25).

7 附录

7.1 代码目录结构

```
../code
  deep_learning.py
  main.R
  src
      load.R
      model.R
      plot.R
      table.R
  temp.R

1 directory, 7 files
```

7.2 主程序

7.2.1 R

```r
# ------------> 加载程序包
library(NMFN)
library(forcats)
library(plyr)
library(nnet)
library(randomForest)


# ------------> 加载依赖函数
source("src/load.R")
source("src/plot.R")
source("src/table.R")
source("src/model.R")


# ------------> 加载数据集
load_mnist()
# classNames = c('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneake
```

```
classNames = c("T 恤/上衣" , " 裤子" , " 套头衫" , " 连衣裙" , " 外套" , " 凉鞋" , " 衬衫" , " 运动鞋"
labels = data.frame(label = 0:9, classNames = classNames)


# ------------> 标准化: -1 -- 1
train[['Xscaled']] = (train[['x']] - 127.5) / 127.5
test[['Xscaled']] = (test[['x']] - 127.5) / 127.5


# ------------> 绘图: 原数据集
plotRaw(train[['x']][1, ])


# ------------> 非负矩阵分解: 求解
datanmf = nnmf(t(train[['x']]),12)
save(datanmf, file = '../model/datanmf.RData')
load("../model/datanmf.RData")
nmfw = datanmf[[1]]# 基矩阵
nmfh = datanmf[[2]]# 系数矩阵


# ------------> 绘图: 降维后的变量图
plotVar(nmfw)


# ------------> 表格: 降维后类别的权重
tableWeight(nmfh, train, labels)


# ------------> 神经网络
neural(train, test)


# ------------> 随机森林
forest(train, test, classNames)


# ------------> NMF / PCA - 维度 - 随机森林
# NMF
```

```
forestDimNMF(train, test)
# PCA
forestDimPCA(train, test)
# plot
plotNMFforestPCA()


# ------------> NMF / PCA - 维度 - 种类 - 随机森林
# NMF
forestDimClassNMF(train, test, classNames)
# PCA
forestDimClassPCA(train, test, classNames)


# ------------> NMF - 随机森林 - 错判图
misJudge(train, test, classNames, 1)
```

7.2.2 python: 深度学习

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

# -------------> configuration
mpl.rcParams['font.sans-serif'] = 'SimHei'
mpl.rcParams['figure.figsize'] = (10, 10)
mpl.rcParams['axes.grid'] = False

print(tf.__version__)

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
class_names = ["T 恤/上衣" , " 裤子" , " 套头衫" , " 连衣裙" , " 外套" , " 凉鞋" , " 衬衫" , " 运动鞋"

plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

train_images = train_images / 255.0
test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])


model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)

test_labels[0]

def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()

i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()

plt.figure()
num_rows = 3
num_cols = 2
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.savefig('../figure/深度学习预测结果.pdf')

img = test_images[1]
img = (np.expand_dims(img,0))

predictions_single = probability_model.predict(img)

plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)

print('finish')
```

7.3 函数封装

7.3.1 数据集加载

```
load_mnist <- function() {
  load_image_file <- function(filename) {
    ret = list()
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    ret$n = readBin(f,'integer',n=1,size=4,endian='big')
    nrow = readBin(f,'integer',n=1,size=4,endian='big')
    ncol = readBin(f,'integer',n=1,size=4,endian='big')
    x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
    ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
    close(f)
    ret
  }
  load_label_file <- function(filename) {
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    n = readBin(f,'integer',n=1,size=4,endian='big')
    y = readBin(f,'integer',n=n,size=1,signed=F)
    close(f)
    y
  }
  train <<- load_image_file('../data/train-images-idx3-ubyte')
  test <<- load_image_file('../data/t10k-images-idx3-ubyte')

  train$y <<- load_label_file('../data/train-labels-idx1-ubyte')
  test$y <<- load_label_file('../data/t10k-labels-idx1-ubyte')
}



show_digit <- function(arr784, col=gray(12:1/12), ...) {
  png(filename = "./temp.png")
  image(matrix(arr784, nrow=28)[,28:1], col=col, ...)
  dev.off()
}
```

7.3.2 模型运行

```
scale01=function(x)
{
  xscale=(x-min(x))/(max(x)-min(x))
  return(xscale)
}


neural = function(train, test)
{
  trainScaledX = apply(train[['x']], 2, scale01)
  testScaledX = apply(test[['x']], 2, scale01)

  trainScaled = data.frame(trainScaledX, train[['y']])
  testScaled = data.frame(testScaledX, test[['y']])

  names(trainScaled)[ncol(trainScaled)] = "y"
  names(testScaled)[ncol(testScaled)] = "y"

  trainScaled$y = as.factor(trainScaled$y)
  testScaled$y = as.factor(testScaled$y)

  rang1 = 1/max(abs(trainScaledX))
  corrate=rep(0,10)
  for(i in 1:10)
  {
    set.seed(1)
    nn1=nnet(y ~.,trainScaled,size=i,rang=rang1, MaxNWts=30000)
    # 预测
    prenn1=predict(nn1,testScaled,type="class")
    # 正确率
    corrate[i]=sum(prenn1==testScaled$y)/nrow(testScaled)
  }
  corrateNNET = data.frame(size = 1: 10, corrate_rate = corrate)
  write.csv(corrateNNET, "../output/corrateNNET.csv", row.names = FALSE)
}
```

```
forest = function(train, test, classNames)
{
  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)

  difnumtot=matrix(0,1,10)
  # rf1=randomForest(num~.,feattrda,mtry=16,ntree=100,importance=T)
  rf1=randomForest(num~.,feattrda,mtry=16,ntree=10,importance=T)
  prerf1=predict(rf1,featteda,type="class")
  # 总体预测正确率
  print(sum(prerf1==featteda[,ncol(featteda)])/nrow(featteda))
  # 不同数字正确率
  for(i in 1:10)
  {
    tr0=(featteda$num==(i-1))
    difnumtot[,i]=sum(prerf1[tr0]==featteda[tr0,ncol(featteda)])/sum(tr0)
  }
  write.csv(difnumtot,"../output/随机森林不同类别预测正确率.csv")

  difnumtot=as.vector(difnumtot)
  names(difnumtot) = classNames
  pdf('../figure/随机森林不同类别预测正确率.pdf', family = "GB1", width = 10, height = 5)
  barplot(difnumtot,col=heat.colors(10)[1:10], xlab=" 种类",ylab=" 正确率")
  dev.off()
}


forestDimNMF = function(train, test)
{
  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)
  datatot=rbind(feattrda,featteda)
  trnum = c(1:nrow(feattrda))
  tenum = c(1:nrow(featteda))
```

```
  rfrate=rep(0,15)
  for(i in 1:15)
  {
    nmftot=nnmf(t(datatot[,1:ncol(datatot) - 1]),i)
    nmfwtot=nmftot[[1]]# 基矩阵
    nmfhtot=nmftot[[2]]# 系数矩阵
    nmfdata=as.data.frame(t(nmfhtot))
    nmfdatay=cbind(nmfdata,datatot[, ncol(datatot)])
    names(nmfdatay)[i+1]="num"
    nmfdatay$num=as.factor(nmfdatay$num)
    # 随机森林
    # rfnmf=randomForest(num~.,nmfdatay[trnum,],mtry=ceiling(sqrt(i)),ntree=100)
    rfnmf=randomForest(num~.,nmfdatay[trnum,],mtry=ceiling(sqrt(i)),ntree=10)
    prenmf=predict(rfnmf,nmfdatay[tenum,],type="class")
    rfrate[i]=sum(prenmf==nmfdatay[tenum,i+1])/length(tenum)
  }
  correctRFdimNMF = data.frame(dim = 1: 15, correct_rate = rfrate)
  write.csv(correctRFdimNMF, "../output/correctRFdimNMF.csv", row.names = FALSE)
}


forestDimPCA = function(train, test)
{
  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)
  datatot=rbind(feattrda,featteda)
  trnum = c(1:nrow(feattrda))
  tenum = c(1:nrow(featteda))

  numpca=princomp(datatot[,1: (ncol(datatot) - 1)],cor=TRUE)
  pcasum=summary(numpca,loadings=TRUE)
  pcasum$loadings[,1]
  datatot1=as.matrix(datatot[,1: (ncol(datatot) - 1)])
  corpca=rep(0,15)
  for(i in 1:15)
  {
    datapca=datatot1%*%as.matrix(pcasum$loadings[,1:i])
```

```
    datapca1=as.data.frame(cbind(datapca,datatot[,ncol(datatot)]))
    names(datapca1)[i+1]="num"
    datapca1$num=as.factor(datapca1$num)
    # 随机森林
    # rfpca=randomForest(num~.,datapca1[trnum,],mtry=ceiling(sqrt(i)),ntree=100)
    rfpca=randomForest(num~.,datapca1[trnum,],mtry=ceiling(sqrt(i)),ntree=10)
    prepca=predict(rfpca,datapca1[tenum,],type="class")
    corpca[i]=sum(prepca==datapca1$num[tenum])/length(tenum)
  }
  correctRFdimPCA = data.frame(dim = 1: 15, correct_rate = corpca)
  write.csv(correctRFdimPCA, "../output/correctRFdimPCA.csv", row.names = FALSE)
}


forestDimClassNMF = function(train, test, classNames)
{
  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)
  datatot=rbind(feattrda,featteda)
  trnum = c(1:nrow(feattrda))
  tenum = c(1:nrow(featteda))

  rfrate=rep(0,15)
  difnum=matrix(0,15,10)
  for(i in 1:15)
  {
    datatot=rbind(feattrda,featteda)
    nmftot=nnmf(t(datatot[,1:(ncol(datatot) - 1)]),i)
    nmfwtot=nmftot[[1]]# 基矩阵
    nmfhtot=nmftot[[2]]# 系数矩阵
    nmfdata=as.data.frame(t(nmfhtot))
    nmfdatay=cbind(nmfdata,datatot[,ncol(datatot)])
    names(nmfdatay)[i+1]="num"
    nmfdatay$num=as.factor(nmfdatay$num)
    # 随机森林
    nmfdatatr1=nmfdatay[trnum,]
    nmfdatate1=nmfdatay[tenum,]
```

```
    # rfnmf=randomForest(num~.,nmfdatatr1,mtry=ceiling(sqrt(i)),ntree=100)
    rfnmf=randomForest(num~.,nmfdatatr1,mtry=ceiling(sqrt(i)),ntree=10)
    # 预测
    prenmf=predict(rfnmf,nmfdatate1,type="class")
    rfrate[i]=sum(prenmf==nmfdatate1[,i+1])/length(tenum)
    for(j in 1:10)
    {
      tr0=(featteda$num==(j - 1))
      difnum[i,j]=sum(prenmf[tr0]==nmfdatate1[tr0,i+1])/sum(tr0)
    }
  }
  pdf('../figure/NMF 不同类别不同降维度的预测正确率.pdf', family = "GB1", width = 8, height = 5)
  plot(difnum[,1],type="b",pch=1,xlab=" 维度",ylab=" 正确率",
       ylim=c(0,1),col=rainbow(10)[1])
  for(i in 2:10)
  {
    lines(difnum[,i],type="b",pch=i,col=rainbow(10)[i])
  }
  lines(rfrate,type="b",pch=11,col="black")
  legend("bottomright",inset=0.05,title=" 类别",
  c(classNames, "total"),
  lty=rep(1,11),pch=c(1:11),
  col=c(rainbow(10),"black"))
  dev.off()
}


forestDimClassPCA = function(train, test, classNames)
{
  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)
  datatot=rbind(feattrda,featteda)
  trnum = c(1:nrow(feattrda))
  tenum = c(1:nrow(featteda))

  numpca=princomp(datatot[,1: (ncol(datatot) - 1)],cor=TRUE)
  pcasum=summary(numpca,loadings=TRUE)
```

```
  pcasum$loadings[,1]
  datatot1=as.matrix(datatot[,1:(ncol(datatot) - 1)])
  corpca=rep(0,15)
  difnumpca=matrix(0,15,10)
  for(i in 1:15)
  {
    datapca=datatot1%*%as.matrix(pcasum$loadings[,1:i])
    datapca1=as.data.frame(cbind(datapca,datatot[,ncol(datatot)]))
    names(datapca1)[i+1]="num"
    datapca1$num=as.factor(datapca1$num)
    datapcatr=datapca1[trnum,]
    datapcate=datapca1[tenum,]
    # 随机森林
    # rfpca=randomForest(num~.,datapcatr,mtry=ceiling(sqrt(i)),ntree=100)
    rfpca=randomForest(num~.,datapcatr,mtry=ceiling(sqrt(i)),ntree=10)
    prepca=predict(rfpca,datapcate,type="class")
    corpca[i]=sum(prepca==datapcate$num)/length(tenum)

    for(j in 1:10)
    {
      tr0=(featteda$num==(j - 1))
      difnumpca[i,j]=sum(prepca[tr0]==datapcate[tr0,i+1])/sum(tr0)
    }
  }
  pdf('../figure/PCA 不同类别不同降维度的预测正确率.pdf', family = "GB1", width = 8, height = 5)
  plot(difnumpca[,1],type="b",pch=1,xlab=" 维度",ylab=" 正确率",
       ylim=c(0,1),col=rainbow(10)[1])
  for(i in 2:10)
  {
    lines(difnumpca[,i],type="b",pch=i,col=rainbow(10)[i])
  }
  lines(corpca,type="b",pch=11,col="black")
  legend("bottomright",inset=0.05,title=" 类别",
  c(classNames, "total"),
  lty=rep(1,11),pch=c(1:11),
  col=c(rainbow(10),"black"))
  dev.off()
}
```

```
plotpre=function(actma,prema,mainname,adjust, classNames)
{
  actma[actma==0]=10
  prema1=as.numeric(as.character(prema))
  prema1[prema1==0]=10
  tablei=table(actma,prema1)
  rgbx=rep(c(1:10),10)
  rgby=NULL
  for(i in 1:10)
  {
    rgby=c(rgby,rep(i,10))
  }

  for(j in 1:10){tablei[j,j]=0}# 主对角线线变成 0, 主对角线是预测正确
  maxrbg=max(tablei)
  shapematrix=tablei# 形状矩阵
  colormatrix=maxrbg-tablei# 颜色矩阵

  # 生成 rbg 颜色
  rgbcolor=NULL
  for(i in 1:10)
  {
    for(j in 1:10)
    {
      rgbcolor=c(rgbcolor,rgb(maxrbg,maxrbg,colormatrix[i,j],max=maxrbg))
    }

  }
  shapevector=as.vector(t(shapematrix))
  plot(rgbx,rgby,col=rgbcolor,type="p",pch=16,cex=shapevector/adjust,
       xlab=" 预测值",ylab=" 真实值",xaxt="n",yaxt="n", las = 2)
  text(rgbx,rgby,shapevector,col="lightblue")
  # axis(1,at=c(1:10),label=c(1:9,0))
  axis(1,at=c(1:10),label = classNames)
  # axis(2,at=c(1:10),label=c(1:9,0))
  axis(2,at=c(1:10),label = classNames)
}
```

```r
misJudge= function(train, test, classNames, dimension)
{

  feattrda = data.frame(train[['x']], num = train[['y']])
  featteda = data.frame(test[['x']], num = test[['y']])
  feattrda$num=as.factor(feattrda$num)
  featteda$num=as.factor(featteda$num)
  datatot=rbind(feattrda,featteda)
  trnum = c(1:nrow(feattrda))
  tenum = c(1:nrow(featteda))
  datatrnum = feattrda$num
  datatenum = featteda$num

  nmftot6=nnmf(t(datatot[,1: (ncol(datatot) - 1)]), dimension)
  nmfwtot6=nmftot6[[1]]# 基矩阵
  nmfhtot6=nmftot6[[2]]# 系数矩阵

  nmfdata6=as.data.frame(t(nmfhtot6))
  nmfdatay6=cbind(nmfdata6,datatot[, ncol(datatot)])

  names(nmfdatay6)[dimension + 1]="num"
  nmfdatay6$num=as.factor(nmfdatay6$num)
  # 随机森林
  nmfdatatr6=nmfdatay6[trnum,]
  nmfdatate6=nmfdatay6[tenum,]
  # rfnmf6=randomForest(num~.,nmfdatatr6,mtry=ceiling(sqrt(6)),ntree=100)
  rfnmf6=randomForest(num~.,nmfdatatr6,mtry=ceiling(sqrt(6)),ntree=10)
  prenmf6=predict(rfnmf6,nmfdatate6,type="class")
  sum(prenmf6==nmfdatate6$num)/length(tenum)
  table6=table(nmfdatate6$num,prenmf6)

  plotname = paste0('../figure/', dimension, " 维 NMF 降维随机森林错判图.pdf")
  pdf(plotname, family = "GB1", width = 10, height = 10)
  plotpre(datatenum,prenmf6, plotname, 25, classNames)
  dev.off()
}
```

7.3.3 图像输出

```
plotRaw = function(feature)
{
  numrgb=feature
  rgbx=rep(c(1:28),28)        # 每个图片 256, 转成 28*28 的图片
  rgby=NULL
  for(i in 28:1)
  {
    rgby=c(rgby,rep(i,28))
  }
  # 生成 rbg 颜色
  rgbcolor=NULL
  for(i in 1:784)
  {
    r1=numrgb[i]
    g1=numrgb[i]
    b1=numrgb[i]
    rgbcolor=c(rgbcolor,rgb(r1,g1,b1,max=255))
  }
  pdf('../figure/图像示例.pdf')
  plot(rgbx,rgby,col=rgbcolor,type="p",pch=15,cex=3,xlab="",ylab="")
  dev.off()
}

plotVar=function(nmfw)
{
  pdf('../figure/降维后的新变量图.pdf')
  par(mfrow = c(3, 4))
  for(i in 1: ncol(nmfw))
  {
    nmfwma = nmfw[, i]
    plotname = paste("Var:", i)
    nmfwmasc=(nmfwma-min(nmfwma))/(max(nmfwma)-min(nmfwma))# 标准化到 0~1
    nmfw255=nmfwmasc*255# 变成 rbg 颜色
    rgbx=rep(c(1:28),28)
    rgby=NULL
    for(i in 28:1)
    {
```

```
      rgby=c(rgby,rep(i,28))
    }
    rgbcolor=NULL # 生成 rbg 颜色
    for(i in 1:784)
    {
      r1=nmfw255[i]
      g1=nmfw255[i]
      b1=nmfw255[i]
      rgbcolor=c(rgbcolor,rgb(r1,g1,b1,max=255))
    }
    plot(rgbx,rgby,col=rgbcolor,type="p",pch=15,cex=3, main=plotname,xlab="",ylab="")
  }
  dev.off()
}

plotNMFforestPCA = function()
{
  rfrate = read.csv("../output/correctRFdimNMF.csv")$correct_rate
  corpca = read.csv("../output/correctRFdimPCA.csv")$correct_rate
  pdf('../figure/PCA 及 NMF 在不同维数预测的正确率线图.pdf', family = 'GB1', width = 8, height = 4)
  plot(corpca,type="b",col="lightblue",pch=1,xlab=" 维度",ylab=" 正确率", ylim=c(0,1))
  lines(rfrate,type="b",col="pink",pch=2)
  legend("bottomright",inset=0.05,title="PCA and NMF",c("PCA","NMF"),
  lty=c(1,1),pch=c(1,2),col=c("lightblue","pink"))
  dev.off()
}
```

7.3.4 表格输出

```
tableWeight = function(nmfh, train, labels)
{
  nmfht=t(nmfh)
  nmfhy=cbind(nmfht, train[['y']])# 合并后矩阵
  nmfhy=as.data.frame(nmfhy)

  var1=arrange(aggregate(V1~V13,nmfhy,mean),desc(V1))# 排序第 1 个变量

  var1=arrange(aggregate(V1~V13,nmfhy,mean),desc(V1))# 排序第 1 个变量
```

```r
  var2=arrange(aggregate(V2~V13,nmfhy,mean),desc(V2))# 排序第 2 个变量

  var3=arrange(aggregate(V3~V13,nmfhy,mean),desc(V3))# 排序第 3 个变量

  var4=arrange(aggregate(V4~V13,nmfhy,mean),desc(V4))# 排序第 4 个变量

  var5=arrange(aggregate(V5~V13,nmfhy,mean),desc(V5))# 排序第 5 个变量

  var6=arrange(aggregate(V6~V13,nmfhy,mean),desc(V6))# 排序第 6 个变量

  var7=arrange(aggregate(V7~V13,nmfhy,mean),desc(V7))# 排序第 7 个变量

  var8=arrange(aggregate(V8~V13,nmfhy,mean),desc(V8))# 排序第 8 个变量

  var9=arrange(aggregate(V9~V13,nmfhy,mean),desc(V9))# 排序第 9 个变量

  var10=arrange(aggregate(V10~V13,nmfhy,mean),desc(V10))# 排序第 10 个变量

  var11=arrange(aggregate(V11~V13,nmfhy,mean),desc(V11))# 排序第 11 个变量

  var12=arrange(aggregate(V12~V13,nmfhy,mean),desc(V12))# 排序第 12 个变量

  raw = data.frame(var1[, 1], var2[, 1], var3[, 1], var4[, 1], var5[, 1], var6[, 1])
  table = data.frame(权重排名 = 1:10)
  for(i in 1:ncol(raw))
  {
    co = data.frame(raw[, i])
    names(co) = 'label'
    co = dplyr::left_join(co, labels, by = "label")
    co = data.frame(co[, 2])
    names(co) = paste(" 变量", i)
    table = cbind(table, co)
  }
  write.csv(table, "../output/10 种不同类别在前六个新变量中的权重.csv", row.names = FALSE)
}
```