模式识别与机器学习课程作业

学院	中国科学院大学 人工智能学院
姓名	芮志清
学号	2018Z8020661080
邮箱	ruizhiqing18📫mails.ucas.ac.cn
时间	2019年5月31日

作业要求

老师给一些数据，从中任选一个，自选主题，做一个实验

截止日期：**6月1日**

选题

数据集

本次选择的数据为“Bank Marketing Data”[1]（银行市场数据）。该数据与葡萄牙银行机构的电话推广活动有关。 数据包含该数据集的分类目标为预测客户是否订阅定期存款服务。

本次实验采用数据集的标准子集，包含以下特征列：

1. 年龄
2. 工作类型
3. 婚姻状况
4. 教育
5. 默认信用
6. 是否有住房贷款
7. 是否有个人贷款
8. 联系人沟通方式
9. 上次联系的在每月的第几天
10. 上次联系的月份
11. 上次联系为周几
12. 沟通时长
13. 与客户的联系次数
14. 上一次推广后经过的天数
15. 此活动前和此客户之前的联系次数
16. 上一次营销活动的结果
17. 有客户是否订阅定期存款？

实验目标

搭建神经网络模型，实现对用户订阅存款的预测。

实验分析

数据预处理

本次数据集为结构化数据，包括17列，其中16列为特征列，1列为标签列。特征列中包含连续值、类别和布尔值，但是类别和布尔值均为字符串表示，需要将其进行数值化处理。

标签为布尔值，因此本实验为一个**二分类**的预测问题。

在类别特征中，类别之间并无数值大小的关系，因此对其进行onehot编码。

在onehot编码后，出现了较多特征，因此可以计算特征和目标的权重，取一部分特征，进行特征降维，准备使用xgboost实现此功能。

在特征降维之前，为了使得xgboost使用更加准确，因此在onehot编码后先进行归一化处理。

搭建模型选择

本次实验准备采用tensorflow及keras搭建多层全连接神经网络模型。

在层间采用**relu**激活函数，在输出层采用**sigmod**激活函数，优化器采用**rmsprop**，损失函数采用**binary_crossentropy**。

实验进行情况

实验环境

本次实验采用python3作为编程语言，实验代码运行在Google Colab上(ML-Assignment)，代码工程托管在github上(ZQRui / ML-Assignment)

代码及运行情况

使用的库

```python
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import xgboost as xgb
import json
import operator
from collections import OrderedDict
import os,sys
import logging

```

设置日志输出

```python
#DEBUG to const.LOGFILE
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s', # 输出格式
                    datefmt='%a, %d %b %Y %H:%M:%S',
                    filename="debug.log.txt",
                    filemode='a')

# to console
console = logging.StreamHandler()
console.setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s: %(levelname)-5s %(message)s')
console.setFormatter(formatter)
logging.getLogger('').addHandler(console)

#INFO to const.INFOLOGFILE
infolog = logging.FileHandler("info.log.txt")
infolog.setLevel(logging.INFO)
errformatter = logging.Formatter('%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s')
infolog.setFormatter(errformatter)
logging.getLogger('').addHandler(infolog)

#ERROR to const.ERRLOGFILE
errlog = logging.FileHandler("error.log.txt")
errlog.setLevel(logging.WARNING)
errformatter = logging.Formatter('%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s')
errlog.setFormatter(errformatter)
logging.getLogger('').addHandler(errlog)

```

读数据

```python
dataset_file="./bank.csv"
```

```python
df=pd.read_csv(dataset_file,sep=";")
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campa
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1

数据集的描述

```
df.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

建立onehot映射表

```
onehot_cols=["job","marital","education","default","housing","loan","contact","month","poutcome"]
```

```
for col in onehot_cols:
  df[col]=df[col].astype("category")
```

```
onehot_newkey=[]
for col in onehot_cols:
  for v in df[col].unique():
    c=(f"{col}-{v}",col,v)
    onehot_newkey.append(c)
```

```
onehot_newkey.append(("pdays--1","pdays",-1))
onehot_newkey.append(("pdays-yes","pdays--1",0))
```

Onehot编码后数据列的映射表

```
onehot_newkey
```

```
# 分别对应编码后的列名，编码前的列名，编码值
[('job-unemployed', 'job', 'unemployed'),
('job-services', 'job', 'services'),
('job-management', 'job', 'management'),
('job-blue-collar', 'job', 'blue-collar'),
('job-self-employed', 'job', 'self-employed'),
('job-technician', 'job', 'technician'),
('job-entrepreneur', 'job', 'entrepreneur'),
('job-admin.', 'job', 'admin.'),
('job-student', 'job', 'student'),
('job-housemaid', 'job', 'housemaid'),
('job-retired', 'job', 'retired'),
('job-unknown', 'job', 'unknown'),
('marital-married', 'marital', 'married'),
('marital-single', 'marital', 'single'),
('marital-divorced', 'marital', 'divorced'),
('education-primary', 'education', 'primary'),
('education-secondary', 'education', 'secondary'),
('education-tertiary', 'education', 'tertiary'),
('education-unknown', 'education', 'unknown'),
('default-no', 'default', 'no'),
('default-yes', 'default', 'yes'),
('housing-no', 'housing', 'no'),
('housing-yes', 'housing', 'yes'),
('loan-no', 'loan', 'no'),
('loan-yes', 'loan', 'yes'),
('contact-cellular', 'contact', 'cellular'),
('contact-unknown', 'contact', 'unknown'),
('contact-telephone', 'contact', 'telephone'),
('month-oct', 'month', 'oct'),
('month-may', 'month', 'may'),
('month-apr', 'month', 'apr'),
('month-jun', 'month', 'jun'),
('month-feb', 'month', 'feb'),
('month-aug', 'month', 'aug'),
('month-jan', 'month', 'jan'),
('month-jul', 'month', 'jul'),
('month-nov', 'month', 'nov'),
('month-sep', 'month', 'sep'),
('month-mar', 'month', 'mar'),
('month-dec', 'month', 'dec'),
('poutcome-unknown', 'poutcome', 'unknown'),
('poutcome-failure', 'poutcome', 'failure'),
('poutcome-other', 'poutcome', 'other'),
('poutcome-success', 'poutcome', 'success'),
('pdays--1', 'pdays', -1),
('pdays-yes', 'pdays--1', 0)]
```

```
conf={"onehot_cols":onehot_cols,"onehot_newkey":onehot_newkey}
```

```
with open("conf.json","w") as fp:
  json.dump(conf,fp,indent=2)
```

对数据进行Onehot编码及归一化处理

```
df["y"]=(df["y"]=="yes").astype(float)
```

```
def onehot(df,newkeys,oldkeys):
  df=df.copy()
  for key,oldkey,value in newkeys:
    logging.debug(f"{key},{oldkey},{value}")
    df[key]=(df[oldkey]==value).astype(float)
  for oldkey in oldkeys:
    df.pop(oldkey)
  return df
```

```
df2=onehot(df,onehot_newkey,onehot_cols)
```

```
def split_train_test(source, frac, ):
    train_dataset = source.sample(frac=0.8, random_state=0)
    test_dataset = source.drop(train_dataset.index)

    return train_dataset, test_dataset
```

```
train_dataset,test_dataset=split_train_test(df2,0.8)
```

```
# y=train_dataset.pop("y")
train_labels = train_dataset.pop('y')
test_labels = test_dataset.pop('y')


```

```
stats=train_dataset.describe().transpose()
print(stats)
```

```
                     count         mean          std  ...    50%     75%      max
age                 3617.0    41.115842    10.573495  ...   39.0    49.0     87.0
balance             3617.0  1405.922311  2972.465627  ...  444.0  1465.0  71188.0
day                 3617.0    15.848217     8.220174  ...   16.0    21.0     31.0
duration            3617.0   268.094001   265.199283  ...  187.0   333.0   3025.0
```

```
campaign             3617.0     2.809511     3.137596  ...    2.0     3.0     50.0
pdays                3617.0    39.948023   100.672342  ...   -1.0    -1.0    871.0
previous             3617.0     0.553221     1.729015  ...    0.0     0.0     25.0
job-unemployed       3617.0     0.029030     0.167913  ...    0.0     0.0      1.0
job-services         3617.0     0.093171     0.290712  ...    0.0     0.0      1.0
job-management       3617.0     0.215648     0.411328  ...    0.0     0.0      1.0
job-blue-collar      3617.0     0.210395     0.407646  ...    0.0     0.0      1.0
job-self-employed    3617.0     0.040918     0.198127  ...    0.0     0.0      1.0
job-technician       3617.0     0.167819     0.373757  ...    0.0     0.0      1.0
job-entrepreneur     3617.0     0.037600     0.190254  ...    0.0     0.0      1.0
job-admin.           3617.0     0.107271     0.309501  ...    0.0     0.0      1.0
job-student          3617.0     0.017971     0.132863  ...    0.0     0.0      1.0
job-housemaid        3617.0     0.024606     0.154943  ...    0.0     0.0      1.0
job-retired          3617.0     0.048659     0.215184  ...    0.0     0.0      1.0
job-unknown          3617.0     0.006912     0.082861  ...    0.0     0.0      1.0
marital-married      3617.0     0.613215     0.487081  ...    1.0     1.0      1.0
marital-single       3617.0     0.264860     0.441320  ...    0.0     1.0      1.0
marital-divorced     3617.0     0.121924     0.327244  ...    0.0     0.0      1.0
education-primary    3617.0     0.150401     0.357513  ...    0.0     0.0      1.0
education-secondary  3617.0     0.508432     0.499998  ...    1.0     1.0      1.0
education-tertiary   3617.0     0.299419     0.458067  ...    0.0     1.0      1.0
education-unknown    3617.0     0.041747     0.200039  ...    0.0     0.0      1.0
default-no           3617.0     0.984241     0.124559  ...    1.0     1.0      1.0
default-yes          3617.0     0.015759     0.124559  ...    0.0     0.0      1.0
housing-no           3617.0     0.424938     0.494402  ...    0.0     1.0      1.0
housing-yes          3617.0     0.575062     0.494402  ...    1.0     1.0      1.0
loan-no              3617.0     0.851811     0.355336  ...    1.0     1.0      1.0
loan-yes             3617.0     0.148189     0.355336  ...    0.0     0.0      1.0
contact-cellular     3617.0     0.644457     0.478744  ...    1.0     1.0      1.0
contact-unknown      3617.0     0.291125     0.454344  ...    0.0     1.0      1.0
contact-telephone    3617.0     0.064418     0.245530  ...    0.0     0.0      1.0
month-oct            3617.0     0.016865     0.128783  ...    0.0     0.0      1.0
month-may            3617.0     0.311861     0.463317  ...    0.0     1.0      1.0
month-apr            3617.0     0.060271     0.238021  ...    0.0     0.0      1.0
month-jun            3617.0     0.118883     0.323696  ...    0.0     0.0      1.0
month-feb            3617.0     0.050041     0.218061  ...    0.0     0.0      1.0
month-aug            3617.0     0.140448     0.347499  ...    0.0     0.0      1.0
month-jan            3617.0     0.031794     0.175476  ...    0.0     0.0      1.0
month-jul            3617.0     0.156483     0.363363  ...    0.0     0.0      1.0
month-nov            3617.0     0.086812     0.281599  ...    0.0     0.0      1.0
month-sep            3617.0     0.010782     0.103291  ...    0.0     0.0      1.0
month-mar            3617.0     0.011059     0.104593  ...    0.0     0.0      1.0
month-dec            3617.0     0.004700     0.068405  ...    0.0     0.0      1.0
poutcome-unknown     3617.0     0.818911     0.385145  ...    1.0     1.0      1.0
poutcome-failure     3617.0     0.110036     0.312978  ...    0.0     0.0      1.0
poutcome-other       3617.0     0.043406     0.203798  ...    0.0     0.0      1.0
poutcome-success     3617.0     0.027647     0.163983  ...    0.0     0.0      1.0
pdays--1             3617.0     0.818911     0.385145  ...    1.0     1.0      1.0
pdays-yes            3617.0     0.181089     0.385145  ...    0.0     0.0      1.0

[53 rows x 8 columns]
```

```python
def norm(df,stats):
  return (df - stats['mean'])/stats['std']
```

```python
normed_train_data=norm(train_dataset,stats)
normed_test_data = norm(test_dataset, stats)
```

特征权重计算

```python
def feature_importance(df,Xl,yl):
    df=df.copy()
    features=Xl
    with open('xgb.fmap',"w") as fpmap:
        i=0
        for fe in features:
            fpmap.write(f"{i}\t{fe}\tq\n")
            i=i+1
    params = {
            'min_child_weight': 0,
            'eta': 0.02,
            'colsample_bytree': 0.7,
            'max_depth': 12,
            'subsample': 0.7,
            'alpha': 1,
            'gamma': 1,
            'silent': 1,
            'verbose_eval': True,
            'seed': 12
        }

    rounds=100
    y=df[yl]

    X=df[Xl]

    xgtrain=xgb.DMatrix(X,label=y)
    bst=xgb.train(params,xgtrain,num_boost_round=rounds)
    importance=bst.get_fscore(fmap="xgb.fmap")

    importance=sorted(importance.items(),key=operator.itemgetter(1),reverse=True)

    return importance
```

```
all_cols=list(df.columns)
all_cols.remove("y")
feature_sel_data=normed_train_dataset.copy()
feature_sel_data["y"]=train_labels
importance=feature_importance(feature_sel_data,normed_train_dataset.columns,"y")
importance
```

特征权重值:

```
[('duration', 275),
 ('age', 131),
 ('day', 113),
 ('pdays', 87),
 ('balance', 80),
 ('poutcome-success', 71),
 ('month-oct', 51),
 ('month-mar', 49),
 ('previous', 39),
 ('contact-unknown', 30),
 ('month-apr', 28),
 ('housing-no', 25),
 ('marital-married', 23),
 ('month-feb', 22),
 ('education-tertiary', 21),
 ('campaign', 20),
 ('month-jun', 17),
 ('poutcome-other', 13),
 ('contact-cellular', 13),
 ('month-may', 13),
 ('month-nov', 10),
 ('month-sep', 9),
 ('contact-telephone', 7),
 ('month-dec', 7),
 ('loan-no', 7),
 ('job-management', 6),
 ('job-blue-collar', 6),
 ('job-student', 6),
 ('month-jul', 5),
 ('job-retired', 4),
 ('job-technician', 4),
 ('housing-yes', 4),
 ('month-aug', 4),
 ('poutcome-failure', 4),
 ('default-yes', 3),
 ('default-no', 3),
 ('month-jan', 3),
 ('job-entrepreneur', 2),
 ('job-housemaid', 2),
 ('education-primary', 2),
 ('job-unemployed', 2),
 ('marital-divorced', 2),
 ('education-unknown', 2),
 ('job-unknown', 2),
 ('pdays--1', 1),
 ('loan-yes', 1)]
```

神经网络模型

超参数

```
args={"feature_count": 6, "train_data_frac": 0.85, "layers_count": [6, 5], "epochs": 5000}

```

取部分特征

```
import_features = [feature for feature, v in importance[:feature_count]]
normed_train_data=normed_train_data[import_features]
normed_test_data = normed_test_data[import_features]
```

神经网络模型

```
def build_model():
    layers_list = [layers.Dense(layers_count[0], activation=tf.nn.relu, input_shape=[feature_count]), ] + \
                  [layers.Dense(count, activation=tf.nn.relu, ) for count in layers_count[1:]] + \
                  [layers.Dense(1, activation=tf.nn.sigmoid)]
    model = keras.Sequential(layers_list)
    model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])

    return model
```

模型训练

```
logging.info(f"evaluate model with arg {args}")
feature_count = args.get("feature_count", 5)

layers_count = args.get("layers_count", [5])
logging.debug(train_dataset.describe())

model=build_model()
EPOCHS = args.get("epochs", 1000)
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
logging.debug(normed_train_data.describe())
history = model.fit(
    normed_train_data, train_labels,
    epochs=EPOCHS, validation_split=0.2, verbose=0, callbacks=[early_stop]
)
test_loss, test_acc = model.evaluate(normed_test_data, test_labels)

logging.info(f"acc: {test_acc} loss:{test_loss} args: {args}")
```

训练结果

```
    2019-05-30 22:31:58,507: INFO  evaluate model with arg {'feature_count': 6, 'train_data_frac': 0.85, 'layers_count': [6, 5], 'epochs': 5000, }


    WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
    Instructions for updating:
    Use tf.cast instead.


    2019-05-30 22:31:58,812: WARNING From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
    Instructions for updating:
    Use tf.cast instead.


    904/904 [==============================] - 0s 30us/sample - loss: 0.2311 - acc: 0.9126

```

模型的测试准确率

模型的准确率达到了91%

```
2019-05-30 22:32:03,508: INFO  acc: 0.9126105904579163 loss:0.23113595727270683 args: {'feature_count': 6, 'train_data_frac': 0.85, 'layers_count': [6, 5], 'epochs': 5000, }

```

超参数优化

本实验还采用了超参数的优化，采用了随机方法选择参数。

```
def model_evaluate_times(d, times):
    s = 0
    for i in range(times):
        logging.debug(f"evaluate times : {times}")
        s += model_evaluate(d)
    s /= times
    logging.info(f"model acc average = {s}")
    return s


def generate_args():
    l = []
    for epochs in [100, 500, 1000, 5000, 10000]:
        for frac in [0.6, 0.7, 0.75, 0.8, 0.85, 0.9]:
            for layers_count in [[i] for i in range(1, 10)] + \
                                [[i, j] for i in range(1, 10) for j in range(1, 10)]:
                #  [[i, j, k] for i in range(1, 10) for j in range(1, 10) for k in range(1, 10)]:
                for fc in range(2, 10):
                    d = {"feature_count": fc, "train_data_frac": frac, "layers_count": layers_count,
"epochs": epochs}
                    l.append(d)
                    logging.debug(f"generate args {d}")
                    yield d
```

实验总结

收获

- 通过本实验，学习了使用TensorFlow搭建多层神经网络模型解决二分类问题
- 采用了OneHot编码方案
- 使用了随机超参数优化方案
- 达到了91%的准确率

问题及解决方案

- 由于超参数优化比较消耗时间，因此不容易寻找到准确率更高的方案。
    - 采用随机方法寻找在超参数空间寻找。
    - 对于数值型超参数，从粗度较大的参数表开始，根据已得到的性能较高的超参数为基础，再进行细粒度的超参数搜索。

1. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.↩