3. 模型可视化应用

之前我们已经完成了 spark 的逻辑回归模型训练，并得到了一个效果还可以的模型，接着做大数据的可视化应用。

三、实验过程或算法（源程序）

1. 实验准备

1.1 建立文件夹得到文件树

```
[hadoop@slave02 chn]$ mkdir src
[hadoop@slave02 chn]$ ls
df_logistic.py  feature_hog.py  handwrittenWords_frontend.zip  ModelBasedSklearn.py  RawDataset  RawDataset.zip  rdd_logistic.py  src  tsne_plot.py
```

1.2 解压文件

1.2.1 安装 linux 下的解压软件

sudo yum install unzip

```
[root@slave02 ~]# su hadoop
[hadoop@slave02 root]$ sudo yum install unzip
[sudo] password for hadoop:
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
base
epel
extras
google-chrome
updates
Package unzip-6.0-24.el7_9.x86_64 already installed and latest version
Nothing to do
```

1.2.2 解压相关文件

cd ~/chn/

unzip RawDataset.zip

解压后得到 RawDataset 文件夹

```
[hadoop@slave02 Ex3_CHN]$ ls
df_logistic.py  feature_hog.py                ModelBasedSklearn.py  RawDataset.zip   tsne_plot.py
ex3.md          handwrittenWords_frontend.zip RawDataset            rdd_logistic.py
```

1.3 数据集可分性测试

1.3.1 安装必要库文件

sudo pip3 install numpy matplotlib scikit-learn

```
[hadoop@slave02 ~]$ sudo pip3 install numpy matplotlib scikit-learn
```

查看输出：

cd /home/hadoop/chn/

python3 ModelBasedSklearn.py

```
[hadoop@slave02 ~]$ cd /home/hadoop/chn/
[hadoop@slave02 chn]$ python3 ModelBasedSklearn.py
11250 3750
load data successful
model train start at:  2023-10-31 20:56:19
model train successful at: 2023-10-31 20:56:19
knn,n=5 model accuracy:  0.4053333333333333
```

这时的模型准确率连 50%也达不到，显然，这个模型的准确率太低了。

1.4 hog 特征提取

1.4.1 补充 feature_hog.py 文件

(1) 实现单机版（使用 skimage 库）

①安装 py4j

```
[hadoop@slave02 root]$ pip install py4j
```

②实现代码

```python
from skimage.feature import hog


data_dir = "RawDataset"

def _get_label(pic_name):
    set_str = pic_name.strip("Locate{}.jpg")  # cut paddings
    label = set_str[-set_str[::-1].index(","):] # get label after the last ','
    return int(label)-1

def _get_pic_data(dir_name):
    pic_names = os.listdir(dir_name)
    img_arrs, labels = [], []

    for pic_name in pic_names:
        imgarr = plt.imread(dir_name + "/" + pic_name)   # matplot读图片
        img_arr = hog(imgarr, cells_per_block=(2, 2))    # hog特征计算
        label = _get_label(pic_name)                     # 求图片的标签
        img_arrs.append(img_arr)
        labels.append(label)
    return img_arrs, labels
```

(2) 实现单机版（使用 opencv 库）

①安装 open-cv

```
[hadoop@slave02 root]$ pip3 install -i https://pypi.douban.com/simple/ pip install opencv-python==4.3.0.38
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.douban.com/simple/
Requirement already satisfied: pip in /usr/local/lib/python3.6/site-packages (21.3.1)
Collecting install
  Using cached https://mirrors.cloud.tencent.com/pypi/packages/4d/c8/8cbca135f9e167810756ea2bc34b028501936675fcbd7dadccf752fa4622/install-1
-any.whl (3.2 kB)
Collecting opencv-python==4.3.0.38
  Downloading https://mirrors.cloud.tencent.com/pypi/packages/26/c5/18f67071e56a1d164757650c39f01f20298a5d4a9fa3ea00fd18187713d9/opencv_pyth
cp36-cp36m-manylinux2014_x86_64.whl (49.3 MB)
     |[redacted]          | 33.3 MB 892 kB/s eta 0:00:18
```

②修改代码

```python
def _get_pic_data(dir_name):
    pic_names = os.listdir(dir_name)
    img_arrs, labels = [], []
    winSize = (64,64)
    blockSize = (16,16)
    blockStride = (8,8)
    cellSize = (8,8)
    nbins = 9
    derivAperture = 1
    winSigma = 4.
    histogramNormType = 0
    L2HysThreshold = 2.0000000000000001e-01
    gammaCorrection = 0
    nlevels = 64
    for pic_name in pic_names:
        imgarr = plt.imread(dir_name + "/" + pic_name)   # matplot读图片
        hog = HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins, derivAperture, winSigma, his
                            L2HysThreshold, gammaCorrection, nlevels)
        img_arr = hog.compute(imgarr)    # hog特征计算
        label = _get_label(pic_name)                    # 求图片的标签
        img_arrs.append(img_arr)
        labels.append(label)
    return img_arrs, labels
```

③实现代码

```
[hadoop@slave02 chn]$ python3 feature_hog.py
load data successful
data saved successful
```

1.5 上传 hdfs

1.5.1 创建文件夹

cd /usr/local/hadoop

bin/hadoop fs -mkdir -p /chn/

```
[hadoop@master spark]$ cd /usr/local/hadoop
[hadoop@master hadoop]$ bin/hadoop fs -mkdir -p /chn/
```

1.5.2 上传数据集

bin/hadoop fs -put /home/hadoop/chn/src/train.csv /chn/

bin/hadoop fs -put /home/hadoop/chn/src/test.csv /chn/

```
[hadoop@master hadoop]$ bin/hadoop fs -put /home/hadoop/chn/src/tra
in.csv  /chn/
```

```
[hadoop@master hadoop]$ bin/hadoop fs -put /home/hadoop/chn/src/tes
t.csv /chn/
```

1.5.3 查看上传是否成功

bin/hadoop fs -ls -R

```
[hadoop@master hadoop]$ ./bin/hadoop fs -ls -R /
```

2. 完成编码

2.1 补充 rdd_logistic.py（单机版）

大多为修改路径

Experiment > Ex3 > Ex3_CHN > rdd_logistic.py

```python
# author: Marx
# time:2020-11-15
# pyspark 2.4.7

from pyspark import SparkConf, SparkContext
# from pyspark.sql import SparkSession
from pyspark.ml.image import ImageSchema
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.classification import SVMWithSGD
from pyspark.mllib.classification import LogisticRegressionModel
from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from pyspark.ml.classification import LogisticRegression
from pyspark.sql.functions import lit
import numpy as np
import time

from pyspark.sql import SQLContext


conf = SparkConf().setAppName("ChineseHandwrittingNumber").setMaster("local")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
sc.setLogLevel("WARN")   # 设置日志级别
# spark = SparkSession(sc)

TRAINPATH = "file:/home/hadoop/Experiment/Ex3/Ex3_CHN/src/train.csv"
TESTPATH = "file:/home/hadoop/Experiment/Ex3/Ex3_CHN/src/test.csv"


## 保存模型
import os, tempfile
path = tempfile.mkdtemp()
model.save(sc, "file:" +  path)
print("Model saved at: ", "file:" + path)
```

2.2 补充 rdd_logistic.py（集群版）

spark_rdd.py > ...

```python
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.ml.image import ImageSchema
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.classification import SVMWithSGD
from pyspark.mllib.classification import LogisticRegressionModel
from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from pyspark.ml.classification import LogisticRegression
from pyspark.sql.functions import lit
import numpy as np
import time

conf = SparkConf().setAppName("ChineseHandwrittingNumber").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")   # 设置日志级别
spark = SparkSession(sc)

print("load spark successful")

TRAINPATH = "/chn/train.csv"
TESTPATH = "/chn/test.csv"
```

spark_rdd.py > ...

```python
def GetParts(line):
    """将一行不定长的数字数据转换成LabeledPoint，便于pyspark.mllib中的各种库调用
    其中最后一个数字是label，前面的都是feature
    例如 1,2,3,4,5,6,7,8 转换成[8, Vector[1,2,3,4,5,6,7]]
        9,10,11,12 转换成 [12, Vector[9,10,11]]
    """
    parts = line.split(',')
    return LabeledPoint(float(parts[-1]),Vectors.dense(parts[:-1]))

rdd_train = sc.textFile(TRAINPATH)
rdd_test = sc.textFile(TESTPATH)

rdd_train = rdd_train.map(lambda line:GetParts(line))
rdd_test = rdd_test.map(lambda line:GetParts(line))

print("load hdfs data successful")
```

```
spark_rdd.py > ...
## 训练逻辑回归多分类器
print("model train start at:", time.strftime('%Y-%m-%d %H:%M:%S'))
model = LogisticRegressionWithLBFGS().train(rdd_train, iterations=100, numClasses=15)
print("model train successful at:", time.strftime('%Y-%m-%d %H:%M:%S'))

## 保存模型
import os, tempfile
path = tempfile.mkdtemp()
model.save(sc, path)
print("Model saved at: ",path)

## 计算准确率
scoreAndLabels = rdd_test.map(lambda point:(model.predict(point.features),point.label))
accuracy = scoreAndLabels.filter(lambda l: l[0]==l[1]).count() / rdd_test.count()
print("accuracy: ",accuracy)
```

2.3 集群运行

2.3.1 集群运行任务

(1) 启动集群

启动 hadoop 集群

cd /usr/local/hadoop

sbin/start-all.sh

```
[hadoop@master ~]$ cd /usr/local/hadoop/
[hadoop@master hadoop]$ sbin/start-all.sh
```

启动 spark 集群

cd /usr/local/spark

sbin/start-master.sh

sbin/start-slaves.sh

```
[hadoop@master hadoop]$ cd /usr/local/spark/
[hadoop@master spark]$ sbin/start-master.sh
```

(2) 上传集群运行任务

①单机版

```
[hadoop@master spark]$ bin/spark-submit /home/hadoop/chn/rdd_logist
ic.py
```

②集群版

cd /usr/local/spark

bin/spark-submit --master spark://master:7077 --executor-memory 500M /home/hadoop/chn/rdd_logistic.py

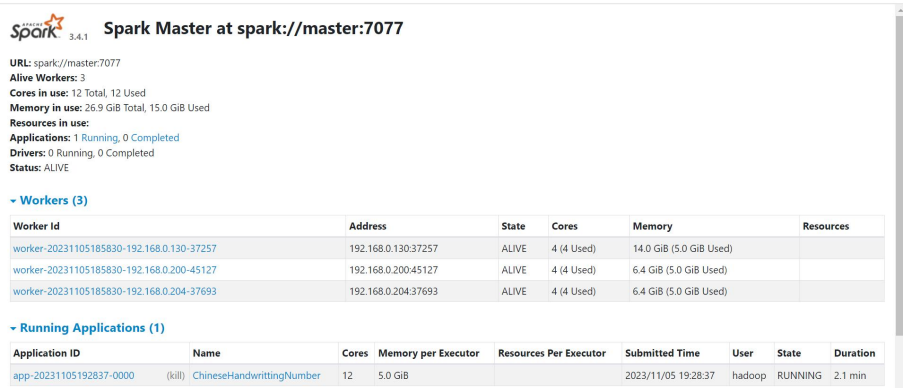
```
[hadoop@master spark]$ bin/spark-submit --master spark://master:707
7 --executor-memory 5G /home/hadoop/chn/spark_rdd.py
```

(3) 运行过程

```
load spark successful
load hdfs data successful
model train start at: 2023-11-05 19:28:38
model train successful at: 2023-11-05 19:42:30
Model saved at:  /tmp/tmpjx67v7uq
accuracy:  0.8490666666666666
```

(4) Web UI 查看

Master 服务器输入：master:8080， 可查看此前运行的应用进程信息：



(5) SparkUI 查看

在浏览器里打开 master:4040，这里的 master 需要更换成 master 主机的 ip 地址，即可查看 SparkUI，详细分析任务的执行情况。具体结果请见第四大点。

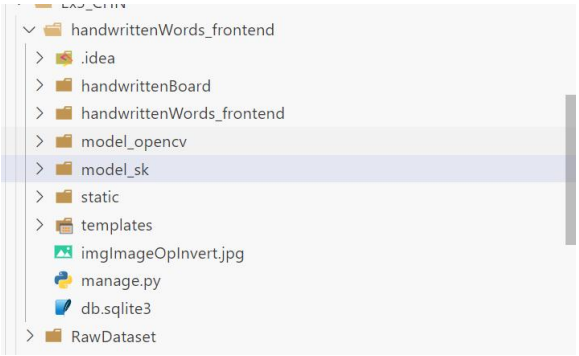
2.4 模型可视化应用

2.4.1 运行环境准备

- 更新必要的依赖库，使其满足 web 项目要求

```
^C[hadoop@slave04 handwrittenWords_frontend]$ sudo pip3 install numpy scikit-image==0.17.2 django==2.1.8 pyspark
2.4.7
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Requirement already satisfied: numpy in /usr/local/lib64/python3.6/site-packages (1.19.5)
Requirement already satisfied: scikit-image==0.17.2 in /usr/local/lib64/python3.6/site-packages (0.17.2)
Requirement already satisfied: django==2.1.8 in /usr/local/lib/python3.6/site-packages (2.1.8)
Requirement already satisfied: pyspark==2.4.7 in /usr/local/lib/python3.6/site-packages (2.4.7)
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib64/python3.6/site-packages (from scikit-image==0.17
.2) (1.5.4)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib64/python3.6/site-packages (from sciki
t-image==0.17.2) (3.3.4)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib64/python3.6/site-packages (from scikit-image=
=0.17.2) (1.1.1)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib64/python3.6/site-packages (from s
cikit-image==0.17.2) (8.4.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.6/site-packages (from scikit-image=
```

- 解压 web 项目并将模型加入项目目录下(model_opencv)



- 修改源代码

```
# start spark local and load model
conf = SparkConf().setAppName("load_and_predict").setMaster("local")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")   # 设置日志级别
spark = SparkSession(sc)

model = LogisticRegressionModel.load(sc, "./model_opencv")
```

2.4.2 代码运行

```
[hadoop@slave04 handwrittenWords_frontend]$ sudo python3 manage.py runserver 0.0.0.0:80 --noreload
[sudo] password for hadoop:
Performing system checks...

==============run============== handwrittenBoard.views
23/11/06 02:49:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin
-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
System check identified no issues (0 silenced).
```

运行结果展示：

3. 扩展实验

3.1 采用其它算法分析数据集

我们使用一个多层感知机（Multilayer Perceptron，MLP）分类模型，并对手写数字的图像进行分类。其中我们导入了必要的 PySpark 和相关库，创建了一个 SparkConf 对象，设置了应用程序的名称和运行模式（本地模式 "local"）。我们的模型层次结构是多层的，包括输入层、2 个隐藏层和输出层。

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.sql import SQLContext
import tempfile
import time
from pyspark.ml.linalg import Vectors
from pyspark.ml.evaluation import MulticlassClassificationEvaluator


conf = SparkConf().setAppName("ChineseHandwritingNumber").setMaster("local")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
sc.setLogLevel("WARN")
```

```
spark = SparkSession.builder.getOrCreate()

print("Spark session created successfully.")

TRAINPATH = "file:/home/hadoop/Experiment/Ex3/Ex3_CHN/src/train.csv"
TESTPATH = "file:/home/hadoop/Experiment/Ex3/Ex3_CHN/src/test.csv"


def GetParts(line):
    parts = line.split(',')
    return (float(parts[-1]), Vectors.dense([float(i) for i in parts[:-1]]))

# 转格式为 DF
df_train=spark.read.text(TRAINPATH).rdd.map(lambdar:r[0]).map(GetParts).toDF(['
label', 'features'])
df_testspark.read.text(TESTPATH).rdd.map(lambdar:r[0]).map(GetParts).toDF(['labe
l', 'features'])

df_train.show()
df_test.show()

# 网络结构定义
layers = [len(df_train.select("features").first()[0]), 125, 32, 15]

print(len(df_train.select("features").first()[0]))

# 神经网络（此处为简单的感知机）创建
mlp = MultilayerPerceptronClassifier(layers=layers, seed=1234)

# 训练模型
print("Model training started at:", time.strftime('%Y-%m-%d %H:%M:%S'))
model = mlp.fit(df_train)
print("Model training successful at:", time.strftime('%Y-%m-%d %H:%M:%S'))


# model_path = tempfile.mkdtemp()
# model.save(model_path)
# print("Model saved at:", model_path)

# Make predictions and evaluate the model
```

```
predictions = model.transform(df_test)

# 评估预测结果
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("准确率 = %g" % accuracy)
```

四、实验结果及分析和（或）源程序调试过程

五、遇到的问题及解决方案

1.1 安装依赖库出现超时或找不到指定版本

```
[hadoop@slave02 ~]$ sudo pip3 install numpy matplotlib scikit-learn
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Requirement already satisfied: numpy in /usr/local/lib64/python3.6/site-packages (1.19.5)
Requirement already satisfied: matplotlib in /usr/local/lib64/python3.6/site-packages (3.3.4)
Collecting scikit-learn
  Downloading scikit_learn-0.24.2-cp36-cp36m-manylinux2010_x86_64.whl (22.2 MB)
     |                                | 30 kB 2.3 kB/s eta 2:38:12ERROR: Exception:
Traceback (most recent call last):
```

解决方案：

安装时使用镜像网站(如清华镜像)，必要时可以指定版本：

sudo pip3 install numpy matplotlib scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple

1.2 解决实验教程中提到的 opencv 编译问题：

在我们的实验中，python 使用的是 3.6.8 版本，只需 pip3 时指定 opencv 的使用版本号(如 4.3.0.38)即可。

解决方案：

```
[hadoop@slave02 root]$ pip3 install -i https://pypi.douban.com/simple/ pip install opencv-python==4.3.0.38
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.douban.com/simple/
Requirement already satisfied: pip in /usr/local/lib/python3.6/site-packages (21.3.1)
Collecting install
  Using cached https://mirrors.cloud.tencent.com/pypi/packages/4d/c8/8cbca135f9e167810756ea2bc34b028501936675fcbd7dadccf752fa4622/install-1.3.5-py3-none
-any.whl (3.2 kB)
Collecting opencv-python==4.3.0.38
  Downloading https://mirrors.cloud.tencent.com/pypi/packages/26/c5/18f67071e56a1d164757650c39f01f20298a5d4a9fa3ea00fd18187713d9/opencv_python-4.3.0.38-
cp36-cp36m-manylinux2014_x86_64.whl (49.3 MB)
     |                                | 49.3 MB 1.4 MB/s
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.6/site-packages (from opencv-python==4.3.0.38) (1.19.5)
Installing collected packages: opencv-python, install
Successfully installed install-1.3.5 opencv-python-4.3.0.38
```

可以正常导入和使用：

```
# 如果你知道如何解决opencv-python的编译问题,可以使用opencv
# from skimage.feature import hog
from cv2 import HOGDescriptor
```

1.3 无法解析导入 sklearn

```
feature_hog.py 2
periment > Ex3 >
1  impor
2  impor
3  impor
4  from sklearn.model_selection import train_test_split
5
```

无法从源解析导入“sklearn.model_selection”
Pylance(reportMissingModuleSource)
(module) model_selection
查看问题 (Alt+F8)　快速修复... (Ctrl+.)

解决方案：

使用 sudo pip3 而非 sudo pip 即可

1.4 无法解析导入 skimage

```
# 如果你知道如何解决opencv-python的编译问题，
from skimage.feature import hog
```

解决方案：

安装指定版本的 skimage 即可，如：

sudo pip3 install scikit-image==0.17.2

1.5 执行该步出现 py4j 的问题(图缺)

3.3.5 运行web项目

进入到项目文件夹下，输入指令

```bash
cd /home/hadoop/chn/hand
sudo python3 manage.py runserver 0.0.0.0:80 --noreload
```

这个命令的作用是打开web项目，并把项目注册到服务器的80端口，这个端口是http服务的默认端

解决方案：

反复卸载安装 pyspark 即可解决...