机器学习学习报告

周珊琳

上海电力大学
上海, 2019-07-23, 中国

第一节　算法概述

算法名称：AdaBoost

算法原理：Boosting，也称为增强学习或提升法，是一种重要的集成学习技术，能够将预测精度仅比随机猜度略高的弱学习器增强为预测精度高的强学习器，这在直接构造强学习器非常困难的情况下，为学习算法的设计提供了一种有效的新思路和新方法。作为一种元算法框架，Boosting 几乎可以应用于所有目前流行的机器学习算法以进一步加强原算法的预测精度，应用十分广泛，产生了极大的影响。而 AdaBoost 正是其中最成功的代表，被评为数据挖掘十大算法之一。在 AdaBoost 提出至今的十几年间，机器学习领域的诸多知名学者不断投入到算法相关理论的研究中去，扎实的理论为 AdaBoost 算法的成功应用打下了坚实的基础。

第二节　算法设计

2.1　算法流程

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.

2: **for** $t = 1, 2, \ldots, T$ **do**

3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;

4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;

5: **if** $\epsilon_t > 0.5$ **then break**

6: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;

7: $$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t} \end{aligned}$$

8: **end for**

输出: $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \right)$

图 1: AdaBoost 算法流程 (引自西瓜书 174 页图 8.3)

2.2 核心代码

源代码 1:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 23 15:55:05 2019

@author: zsl
"""

import numpy as np
from sklearn.datasets import load_iris,load_wine
from sklearn import preprocessing
import matplotlib.pyplot as plt

def process_data(data):
    min_max_scaler =preprocessing.MinMaxScaler()
    return min_max_scaler.fit_transform(data)

#iris,wine
def loadData(data,y1):
    #load
    dataSet =data.data
    target =data.target
    #shuffle
    num_example=dataSet.shape[0]
    array =np.arange(num_example)
    np.random.shuffle(array)
    dataSet =dataSet[array]
    target =target[array]
    #transform
    classLabels =list(target)
    dataArr =process_data(dataSet)
    for i in range(len(classLabels)):
        if classLabels[i]==y1:
            classLabels[i]=1.0
        else:
            classLabels[i]=-1.0
    return dataArr,classLabels

#car
def loadDataSet(fileName):
    dataMat =[]; classLabels =[]
    fr =open(fileName)
    for line in fr.readlines():
        lineArr =line.strip().split(',')
        # print(lineArr)
        if lineArr[0] =='vhigh':
            lineArr[0] =1
        if lineArr[0] =='high':
            lineArr[0] =2
        if lineArr[0] =='med':
            lineArr[0] =3
```

```
        if lineArr[0] =='low':
            lineArr[0] =4
        if lineArr[1] =='vhigh':
            lineArr[1] =1
        if lineArr[1] =='high':
            lineArr[1] =2
        if lineArr[1] =='med':
            lineArr[1] =3
        if lineArr[1] =='low':
            lineArr[1] =4
        if lineArr[2]=='2':
            lineArr[2]=1
        if lineArr[2]=='3':
            lineArr[2]=2
        if lineArr[2]=='4':
            lineArr[2]=3
        if lineArr[2]=='5more':
            lineArr[2]=4
        if lineArr[3]=='2':
            lineArr[3]=1
        if lineArr[3]=='4':
            lineArr[3]=2
        if lineArr[3]=='more':
            lineArr[3]=3
        if lineArr[4]=='small':
            lineArr[4]=1
        if lineArr[4]=='med':
            lineArr[4]=2
        if lineArr[4]=='big':
            lineArr[4]=3
        if lineArr[5]=='low':
            lineArr[5]=1
        if lineArr[5]=='med':
            lineArr[5]=2
        if lineArr[5]=='high':
            lineArr[5]=3
        dataMat.append([float(lineArr[0]),float(lineArr[1]), float(lineArr[2]),
                    float(lineArr[3]),float(lineArr[4]),
                    float(lineArr[5])])


        if lineArr[6] =='unacc':
            lineArr[6] =1.0
#       elif lineArr[6] == 'acc':
#           lineArr[6] =1
#       elif lineArr[6] == 'good':
#           lineArr[6] =2
        else:
            lineArr[6] =-1.0

        classLabels.append(float(lineArr[6]))

    return np.array(dataMat,dtype="float64"),classLabels

```

```python
def stumpClassify(dataMartix,col,threshVal,threshIneq):
    retArray =np.ones((np.shape(dataMartix)[0],1))
    if threshIneq =='lt':
        retArray[dataMartix[:,col] <=threshVal] =-1.0
    else:
        retArray[dataMartix[:,col] >threshVal] =-1.0
    return retArray

def buildStump(dataArr,classLabels,W):
    dataMatrix =np.mat(dataArr)
    classMatrix =np.mat(classLabels).T
    m,n =np.shape(dataMatrix)
    numSteps =10.0
    bestStump={}
    bestClasEst =np.mat(np.zeros((m,1)))
    minError =np.inf
    for i in range(n):
        colMax =dataMatrix[:,i].max()
        colMin =dataMatrix[:,i].min()
        stepSize =(colMax-colMin)/numSteps
        for j in range(-1,int(numSteps)+1):
            for inequal in ['lt','gt']:
                threshVal =(colMin +float(j)*stepSize)
                predictedVals =stumpClassify(dataMatrix,i,threshVal,inequal)
#               print('predictedVals:================',predictedVals)
                errArr =np.mat(np.ones((m,1)))
#               print('errArr',errArr)
                errArr[predictedVals ==classMatrix] =0
                weightedError =W.T*errArr
#               print("split: col %d, thresh %.2f, thresh ineqal: %s, the weighted error is %.3f" % (i, threshVal, inequal, weightedError))
                if weightedError <minError:
                    minError =weightedError
                    bestClasEst =predictedVals.copy()
                    bestStump['col']=i
                    bestStump['thresh']=threshVal
                    bestStump['ineq'] =inequal
    return bestStump,minError,bestClasEst

def adaBoostTrainDS(dataArr,classLables,T=30):
    weakClassArr =[]
    errorList =[]
    m =np.shape(dataArr)[0]
    W =np.mat(np.ones((m,1))/m)
    aggClassEst =np.mat(np.zeros((m,1)))
    for i in range(T):
        bestStump,error,classEst=buildStump(dataArr,classLables,W)
        alpha =float(0.5*np.log((1.0-error)/max(error,1e-16)))
        bestStump['alpha'] =alpha
        weakClassArr.append(bestStump)
        expon =np.multiply(-1*alpha*np.mat(classLables).T,classEst)
        W =np.multiply(W,np.exp(expon))
        W =W/W.sum()
        aggClassEst +=alpha*classEst
```

```
#        print(aggClassEst)
        aggErrors =np.multiply(np.sign(aggClassEst)!=np.mat(classLables).T,np.ones((m,1)))
#        print('==============',aggErrors)
        errorRate =aggErrors.sum()/m
#        print('total error',errorRate)
        errorList.append(errorRate)
        if errorRate ==0.0:
            break
    return weakClassArr,aggClassEst,errorList

def adaClassify(datToClass,classifierArr):
    dataMatrix =np.mat(datToClass)
    m =np.shape(dataMatrix)[0]
    aggClassEst =np.mat(np.zeros((m,1)))
    for i in range(len(classifierArr)):
        classEst =stumpClassify(dataMatrix,classifierArr[i]['col'],
                                 classifierArr[i]['thresh'],
                                 classifierArr[i]['ineq'])
        aggClassEst +=classifierArr[i]['alpha']*classEst
#        print(np.sign(aggClassEst))
    return np.sign(aggClassEst)

def classifytest(testDataSet, classifierArr,target_test):
    """
        计算准确率
    """
    i =0
    cnt =0
    for testVec in testDataSet:
        pre =adaClassify(testDataSet,classifierArr)
        if (pre ==target_test[i]).all():
            cnt +=1
        i +=1
#    print('cnt',cnt)
    return cnt/len(target_test)

##iris
data =load_iris()
dataArr1,classLabels1=loadData(data,0)

#wine
data =load_wine()
dataArr2,classLabels2=loadData(data,3)

##car
dataArr3,classLabels3=loadDataSet('car.data')
#W = np.mat(np.ones((np.shape(dataArr)[0],1))/np.shape(dataArr)[0])
#bestStump,minError,bestClasEst=buildStump(dataArr,classLabels,W)
#classifierArr,aggClassEst,errorList = adaBoostTrainDS(dataArr1,classLabels1,50)

num_example =dataArr2.shape[0]
sample =np.int(num_example *0.9)
x_train =dataArr2[: sample]
y_train =classLabels2[: sample]
```

```python
x_test =dataArr2[sample:]
y_test =classLabels2[sample:]
res=[]
for j in range(10):
    pri =[]
    tmp =0
    prob =0.1
    for i in range(10):
        classifierArr,aggClassEst,errorList =adaBoostTrainDS(x_train[tmp:np.int(sample*prob)],y_train[
                                                    tmp:np.int(sample*prob)],50)
        pre=classifytest(x_test, classifierArr,y_test)
        pri.append(pre)
        tmp=np.int(sample*prob)+1
        prob +=0.1
    res.append(sum(pri)/len(pri))
    print(sum(pri)/len(pri))
print('wine AdaBoost, 10次十折交叉验证结果:',(sum(res)/len(res)))



#plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
#plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
#plt.figure()
#ln1, = plt.plot(errorList1,linestyle='dashed',linewidth=0.5,color='red',marker='.',)
#ln2, = plt.plot(errorList2,linestyle='dashed',linewidth=0.5,color='b',marker='.',)
#ln3, = plt.plot(errorList3,linestyle='dashed',linewidth=0.5,color='g',marker='.',)
#ln4, = plt.plot(errorList4,linestyle='dashed',linewidth=0.5,color='yellow',marker='.',)
#plt.ylim(0,0.3)
#plt.legend(handles=[ln1,ln2,ln3,ln4], labels=['unacc', 'acc','good','vgood'],
#    loc='uper right')
#plt.ylabel('errorRate')
#plt.title('训练50次过程中数据集car错误率的变化情况')
#plt.show()
```

第三节　选用数据

iris 行数：150 列数：5

列属性及取值:

1) 萼片长度 cm，数值型

2) 萼片宽度 cm，数值型

3) 花瓣长度 cm，数值型

4) 花瓣宽度 cm 数值型

类别：

Iris Setosa

Iris Versicolour

Iris Virginica

wine，行数：178，列数：13

属性：

1) Alcohol

2) Malic acid

3) Ash

4) Alcalinity of ash

5) Magnesium

6) Total phenols

7) Flavanoids

8) Nonflavanoid phenols

9) Proanthocyanins

10) Color intensity

11) Hue

12) OD280/OD315 of diluted wines

13) Proline

类别：

Alcohol 1，2，3

car，行数：1728，列数：6

列属性及取值：

1)buying: vhigh, high, med, low.

2)maint: vhigh, high, med, low.

3)doors: 2, 3, 4, 5more.

4)persons: 2, 4, more.

5)lugboot: small, med, big.

6)safety: low, med, high.

类别：

unacc, acc, good, vgood

第四节　实验结果展示

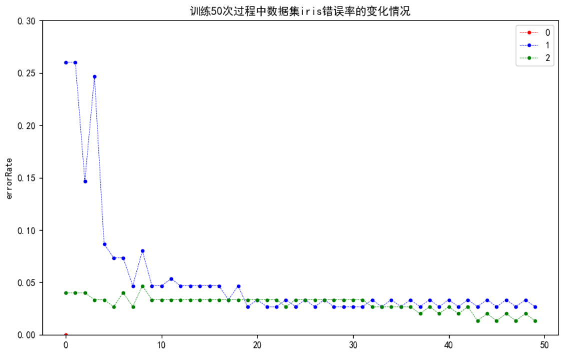

图 2: iris 类标号选择对比图

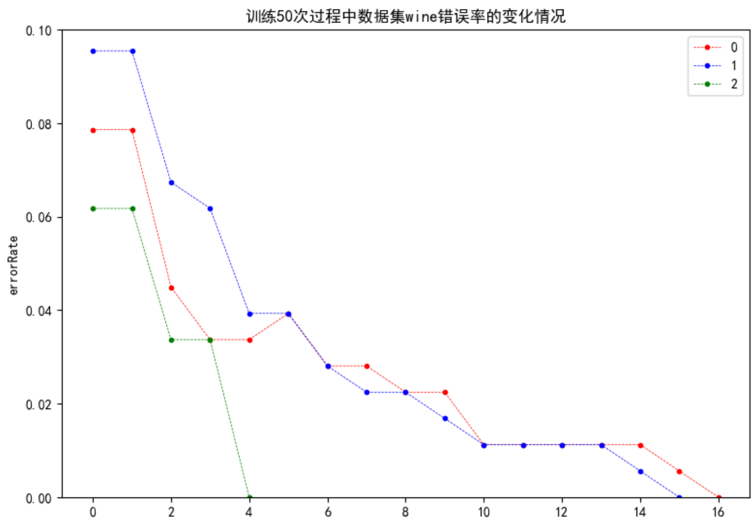

图 3: wine 类标号选择对比图

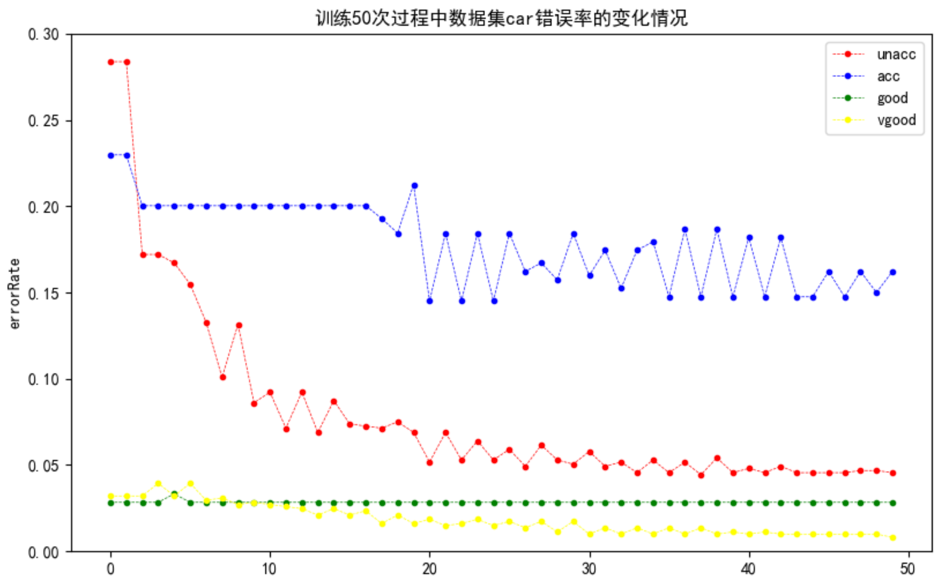

图 4: car 类标号选择对比图

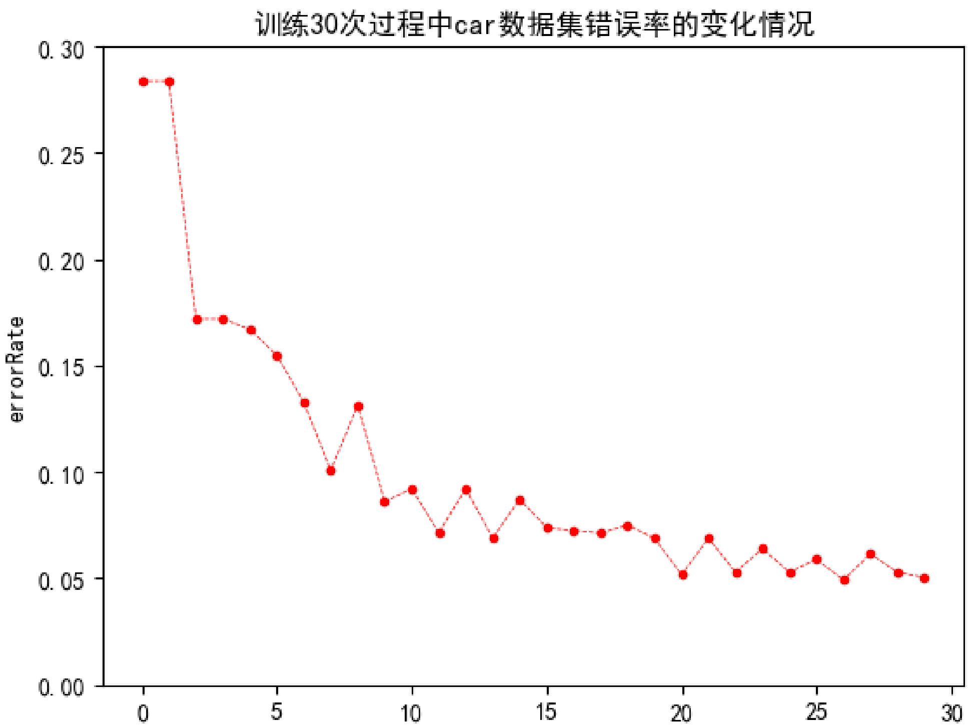

图 5: car30 次训练情况图

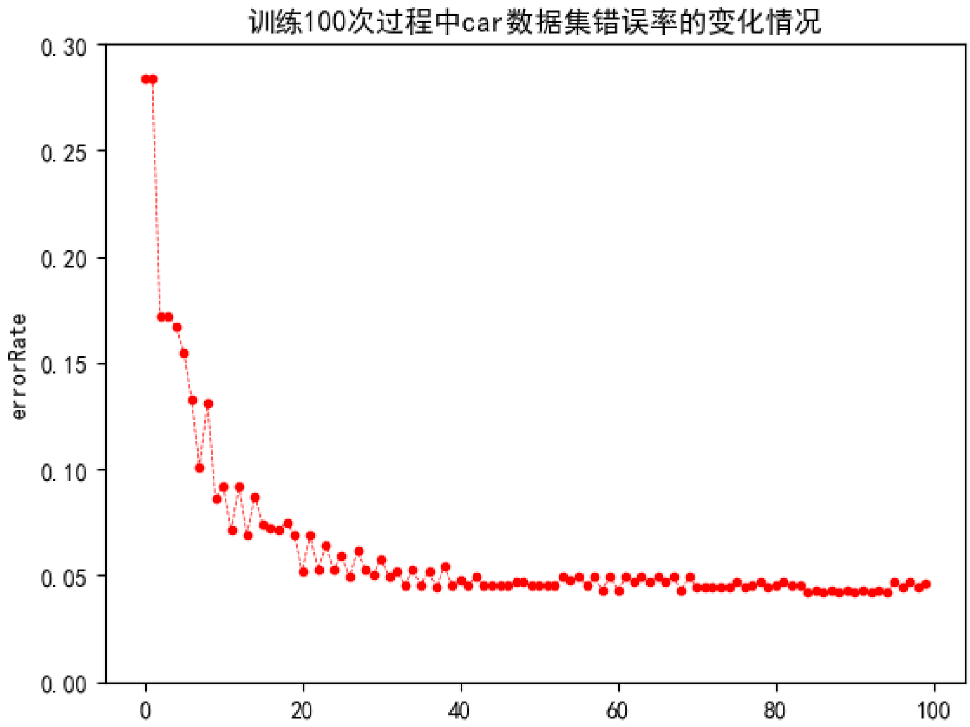

图 6: car100 次训练情况图

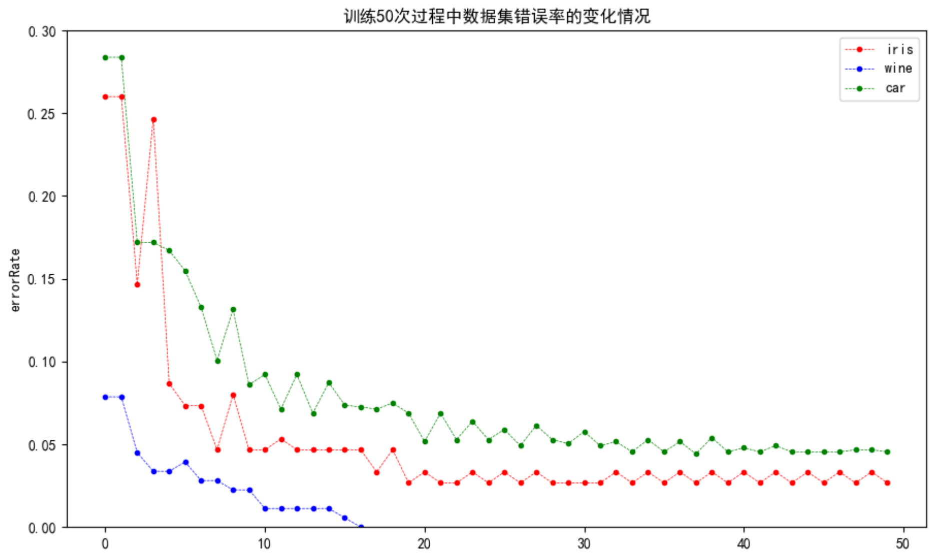

图 7: 三类数据训练情况对比图

第五节 评价方法

十折交叉验证。将数据集分成十份，轮流将其中 9 份作为训练数据，1 份作为测试数据，进行试验。每次试验都会得出相应的正确率（或差错率）。10 次的结果的正确率（或差错率）的平均值作为对算法精度的估计，一般还需要进行多次 10 折交叉验证（例如 10 次 10 折交叉验证），再求其均值，作为对算法准确性的估计。

第六节 实验分析和比较

对于鸢尾花数据进行 10 次十则交叉验证之后的结果为：0.07333；对于红酒数据结果为：0.9；对于汽车数据结果为：0.08588。可以看出此算法相比较而言，可能对于汽车数据更为合适。另外，这次 adaboost 是实现了其二分类的功能，所以在使用的时候需要对数据进行合并，本次采取的合并方式是选择类标号中的一个类作为 1，其他作为-1。进行多次对比实验，寻找训练的 errror 结果最佳的。结果图片中对此有所展示。在学习算法的过程中，对过程的理解一开始花了很久的时间，细节上有很多容易搞混，比如其实 adaboost 算法在调权的时候其实调整了两个权重，一个权重是对于样本而言的，一般以权重向量的方式出现，另外一个是对于弱分类器而言的。对于分类器的形成方式有很多，此次实现的方式是对于训练集的不同属性的子集训练而成的基分类器。