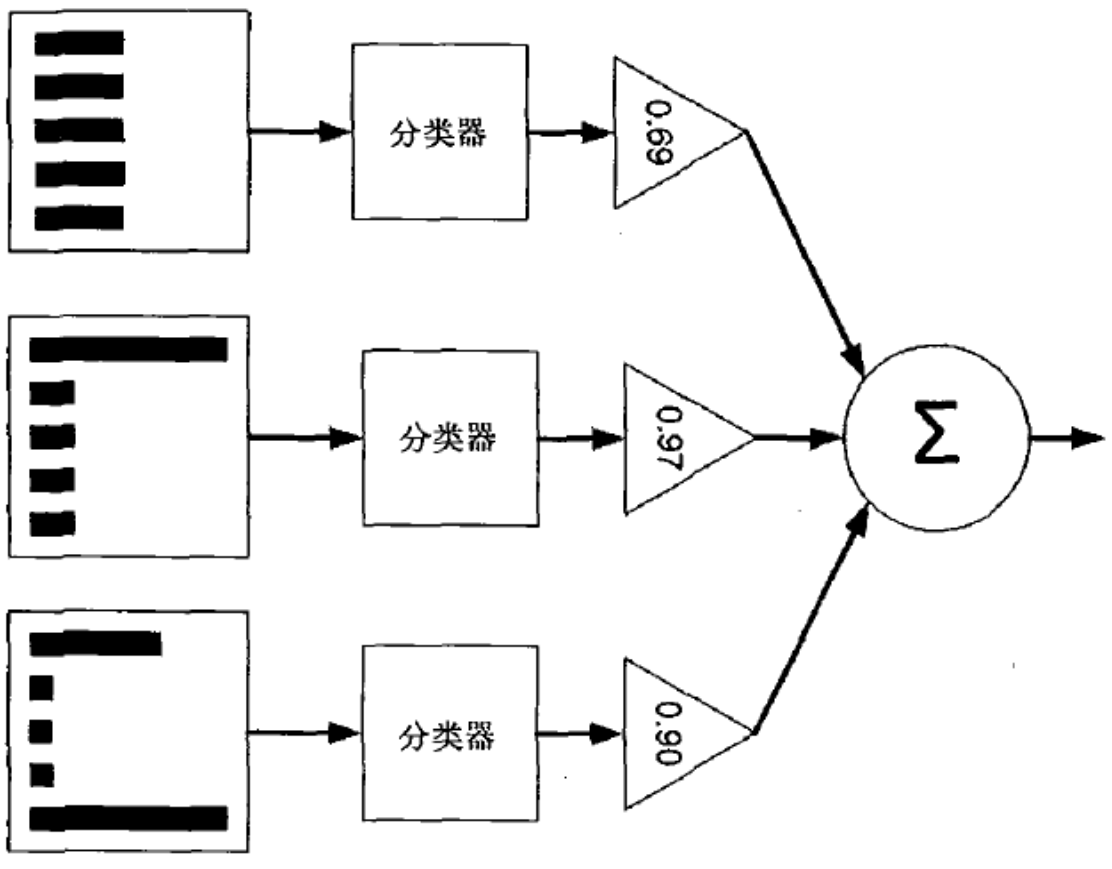
机器学习笔记（四）——Adaboost

文章来自 hidamari 's Blog // 人生苦短，我只为信仰而战。。。

2017-04-21 16:09:17
3 0 0

简介

AdaBoost，是英文"Adaptive Boosting"（自适应增强）的缩写，由Yoav Freund和Robert Schapire在1995年提出。它的自适应在于：前一个基本分类器分错的样本会得到加强，加权后的全体样本再次被用来训练下一个基本分类器。同时，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数。Adaboost的目的就是从训练数据中学习一系列弱分类器或基本分类器，然后将这些弱分类器组合成一个强分类器。



实例简单分析：假设对五个样本进行分析，只建立三个分类器

第一个分类器中的样本权重都是相同的（黑条都是相同长度），计算得到这个分类器的结果占的权重为0.69；

在第二个分类器中，由于第一个分类器分类的结果是第一个样本分

类错误，其他四个都分类正确，所以第一个样本的权重增加（第一个黑条变长），其他四个样本的的分类权重减少（其他四个黑条变短）；由此第一个分类器对第二个分类器进行了加强，计算得到这个分类器的结果占的权重为0.97；

在第三个分类器中，由于第二个分类器分类的结果是第一个样本和第五个样本分类错误，中间三个都分类正确，所以第一个样本和第五个样本的权重增加（第一个和第五个黑条变长），中间三个样本的的分类权重减少（中间三个黑条变短）；由此第二个分类器又对第三个分类器进行了加强，计算得到这个分类器的结果占的权重为0.90；

最后将这三个分类器的分类结果按权重（0.69,0.97，0.90）相加，得到最终的分类结果

算法步骤概括：

初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予相同的权值：1/N。

训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

算法具体实现步骤

步骤*1.* 首先，初始化训练数据的权值分布。每一个训练样本最开始时都被赋予相同的权值：1/N。

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}),\ w_{1i} = \frac{1}{N},\ i = 1,2,\cdots,N$$

步骤2. 进行多轮迭代，用m = 1,2, ..., M表示迭代的第多少轮

使用具有权值分布Dm的训练数据集学习，得到基本分类器：

$$G_m(x):\ \chi \to \{-1,+1\}$$

计算Gm(x)在训练数据集上的分类误差率：

Contents


```
threshval] = -1.0 return retArraydef buildStump(dataArr, classLabels,D): datamatrix = np.mat(dataArr) #转换成矩阵，方便进行运算 labelmat = np.mat(classLabels).T #转置成列向量，因为样本数据就是列向量格式 m,n=np.shape(datamatrix) #获取dataMatrix的行，列数 numsteps = 10.0 # 分割线移动多少次 beststump = { } bestclasEst = np.mat(np.zeros((m,1))) minerror=np.inf for i in range(n): # print "range(n):",range(n) rangemin = datamatrix[:,i].min() # x或者y坐标的最小值 rangemax = datamatrix[:,i].max() # x或者y坐标的最大值 stepsize = (rangemax-rangemin)/numsteps # 分割线移动的步长 for j in range(-1，int(numsteps)+1): #j =[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] for inqual in ['lt', 'gt']: # 分割线左边
```

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

由上述式子可知，$G_m(x)$在训练数据集上的**误差率**em就是被$G_m(x)$误分类样本的权值之和。

例如：

例一：有三个样本分类错误，该三个样本的权重都是0.1，则e_m = 0.1+0.1+0.1=0.3

例二：有三个样本分类错误，该三个样本的权重分别是0.1,0.2，0.3，则e_m = 0.1*1+0.2*1+0.3*1=0.6

计算$G_m(x)$的系数，am表示$G_m(x)$在最终分类器中的重要程度（目的：得到基本分类器在最终分类器中所占的权重）

$$a_m = \frac{1}{2} \ln \frac{1 - e_m}{e_m}$$

由上述式子可知，e_m <= 1/2时，a_m >= 0，且a_m随着e_m的减小而增大，意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

更新训练数据集的权值分布（目的：得到样本的新的权值分布），用于下一轮迭代

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N}),$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

使得被基本分类器$G_m(x)$分类错误的样本的权值增大，而被正确分类样本的权值减小，将重新定义的权值用于$G_{m+1}(x)$分类器上。通过这样的方式，AdaBoost方法能“重点关注”或“聚焦于”那些较难分的样本上。

其中，Zm是规范化因子，使得Dm+1成为一个概率分布：

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

步骤3.组合各个弱分类器

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

是-1 和分割线右边是-1两种情况 threshval =(rangemin+float(j)*stepsize) #分割线在[0.9,2]区间内以0.1为步长移动 predictedvals = stumpClassify(datamatrix,i,threshval,inqual) #进行分类 errarr = np.mat(ones((m,1))) #生成一个大小为5*1，元素为1的矩阵 errarr[predictedvals == labelmat] = 0 #预测的分类（predictedvals）和样本的初始分类（labelmat）一样的，也就是预测正确的，就将errarr矩阵中对应位置的元素置0 weightarr = D.T*errarr #预测正确对应errarr上的位置元素置0了，不正确的为1，错误率em等于预测错误样本的权值相加 if weightarr < minerror: minerror = weightarr bestclasest = predictedvals.copy() beststump['dim'] = i beststump['thresh'] = threshval beststump['ineq'] = inqual return beststump,minerror, bestclasest def adbBoostTrainDs(dataarr,classlabels,numit = 40): """ :param dataarr: 样

从而得到最终分类器，如下：

$$G(x)=sign(f(x))=sign\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)$$

其中：sign（x）函数：

$$\operatorname{sgn} x=\begin{cases}-1 & : \quad x<0\\ 0 & : \quad x=0\\ 1 & : \quad x>0\end{cases}$$

Adaboost的实例分析：

给定下列训练样本进行分析：

序号	1	2	3	4	5	6	7	8	9	X
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

迭代过程1

对于m=1，在权值分布为D1（10个数据，每个数据的权值皆初始化为0.1）的训练数据上，经过计算可得：

阈值v取2.5时误差率为0.3（x < 2.5时取1，x > 2.5时取–1，则6 7 8分错，误差率为0.3），

阈 值v取5.5时误差率为0.4（x < 5.5时取1，x > 5.5时取–1，则3 4 5 6 7 8皆分错，误差率0.6大于0.5，不可取。故令x > 5.5时取1，x < 5.5时取–1，则0 1 2 9分错，误差率为0.4），

阈值v取8.5时误差率为0.3（x < 8.5时取1，x > 8.5时取–1，则3 4 5分错，误差率为0.3）。

可以看到，无论阈值v取2.5，还是8.5，总得分错3个样本，故可任取其中任意一个如2.5，弄成第一个基本分类器为：

$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$

从而得到G1(x)在训练数据集上的误差率（被G1(x)误分类样本“6 7 8”的权值之和）e1=P(G1(xi)≠yi) = 3*0.1 = 0.3。

然后根据误差率e1计算G1分类器的权重系数：

$$a_1=\frac{1}{2}\ln\frac{1-e_1}{e_1}=0.4236$$

这个a1代表G1(x)在最终的分类函数中所占的权重，为0.4236。

```
本数据 :param classlabels: 样本分类值 :param numit: 迭代次数 :return: """ werkclassarr = [] m = np.shape(dataarr)[0] #样本数据多少个 D = np.mat(np.ones((m,1))/m) #初始权值 aggclassest = np.mat(np.zeros((m,1))) for i in range(numit): beststump,error,classest = buildStump(dataarr, classlabels, D) alpha = float(0.5*np.log((1.0-error)/error)) #计算当前分类器的权值 beststump['alpha'] = alpha werkclassarr.append(beststump) #更新样本权值 expon = np.multiply(-1*alpha*np.mat(classlabels).T,classest) D = np.multiply(D,np.exp(expon)) D = D/D.sum() print "D:",D.T aggclassest+=alpha*classest aggerrors = np.multiply(np.sign(aggclassest)!= np.mat(classlabels).T,np.ones((m,1))) errorrate = aggerrors.sum()/m if(errorrate==0.0): break return werkclassarr if __name__ == "__main__": D = np.mat(np.ones((5,1))/5) dat
```

接着更新训练数据的权值分布，用于下一轮迭代：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

值得一提的是，由权值更新的公式可知，每个样本的新权值是变大还是变小，取决于它是被分错还是被分正确。

即如果某个样本被分错了，则yi * Gm(xi)为负，负负得正，结果使得整个式子变大（样本权值变大），否则变小。

第一轮迭代后，最后得到各个数据新的权值分布D2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)。由此可以看出，因为样本中是数据“6 7 8”被G1(x)分错了，所以它们的权值由之前的0.1增大到0.1666，反之，其它数据皆被分正确，所以它们的权值皆由之前的0.1减小到0.0715。

分类函数f1(x)= a1*G1(x) = 0.4236G1(x)。

此时，得到的第一个基本分类器sign(f1(x))在训练数据集上有3个误分类点（即6 7 8）。

从上述第一轮的整个迭代过程可以看出：被误分类样本的权值之和影响误差率，误差率影响基本分类器在最终分类器中所占的权重。

迭代过程2

对于m=2，在权值分布为D2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)的训练数据上，经过计算可得：

阈值v取2.5时误差率为0.1666*3（x < 2.5时取1，x > 2.5时取–1，则6 7 8分错，误差率为0.1666*3），

阈值v取5.5时误差率最低为0.0715*4（x > 5.5时取1，x < 5.5时取–1，则0 1 2 9分错，误差率为0.0715*3 + 0.0715），

阈值v取8.5时误差率为0.0715*3（x < 8.5时取1，x > 8.5时取–1，则3 4 5分错，误差率为0.0715*3）。

所以，阈值v取8.5时误差率最低，故第二个基本分类器为：

$$G_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x > 8.5 \end{cases}$$

G2(x)把样本“3 4 5”分错了，根据D2可知它们的权值为0.0715, 0.0715, 0.0715，所以G2(x)在训练数据集上的误差率e2=P(G2(xi)≠yi) = 0.0715 * 3 = 0.2143。

```
mat,labelmat= loadSimData() beststump,minerror,bestclasest = buildStump(datmat,labelmat,D) print "beststump:",beststump print "minerror:",np.mat(minerror).T print "bestclasest:",np.mat(bestclasest).T adbBoostTrainDs(datmat, labelmat, 10) ')">Python 实现 Adaboost# -*- coding=UTF-8 -*-import numpy as npfrom numpy import ones def loadSimData(): datMat = np.matrix([[1.0,2.1], [2.0,1.1], [1.3,1.0], [1.0,1.0], [2.0,1.0]]) classLabels = [1.0, 1.0,-1.0,-1.0,1.0] return datMat, classLabelsdef stumpClassify(dataMatrix,dimen, threshval, threshIneq): """ :param dataMatrix: 训练样本 :param dimen: 维度（x轴或y轴） :param threshval: 阈值（分割线） :param threshIneq: 决定分割线左边是-1 和分割线右边是-1两种情况 :return: 返回一个分类好的np.array """ retArray = np.ones((np.shape(dataMatrix)[0],1)) if threshIne
```

计算G2的系数：

$$a_2 = \frac{1}{2}\ln\frac{1-e_2}{e_2} = 0.6496$$

更新训练数据的权值分布：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+i} = \frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

D3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.01667, 0.1060, 0.1060, 0.1060, 0.0455)。

迭代过程3：

对于m=3，在权值分布为D3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667，0.01667, 0.1060, 0.1060, 0.1060, 0.0455)的训练数据上，经过计算可得：

阈值v取2.5时误差率为0.1060*3（x < 2.5时取1，x > 2.5时取–1，则6 7 8分错，误差率为0.1060*3），

阈值v取5.5时误差率最低为0.0455*4（x > 5.5时取1，x < 5.5时取–1，则0 1 2 9分错，误差率为0.0455*3 + 0.0715），

阈值v取8.5时误差率为0.1667*3（x < 8.5时取1，x > 8.5时取–1，则3 4 5分错，误差率为0.1667*3）。

阈值v取5.5时误差率最低，故第三个基本分类器为：

$$G_3(x) = \begin{cases} 1, & x < 5.5 \\ -1, & x > 5.5 \end{cases}$$

此时，被误分类的样本是：0 1 2 9，这4个样本所对应的权值皆为0.0455，

所以G3(x)在训练数据集上的误差率e3 = P(G3(xi)≠yi) = 0.0455*4 = 0.1820。

计算G3的系数：

$$a_m = \frac{1}{2}\ln\frac{1-e_3}{e_3} = 0.7514$$

更新训练数据的权值分布：

D4 = (0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125)

```
q == 'lt': retArray[dataMatrix[:,dimen] <= threshval] = -1.0 else: retArray[dataMatrix[:,dimen] > threshval] = -1.0 return retArraydef buildStump(dataArr,classLabels,D): datamatrix = np.mat(dataArr) #转换成矩阵，方便进行运算 labelmat = np.mat(classLabels).T #转置成列向量，因为样本数据就是列向量格式 m,n=np.shape(datamatrix) #获取dataMatrix的行，列数 numsteps = 10.0 # 分割线移动多少次 beststump = { } bestclasEst = np.mat(np.zeros((m,1))) minerror=np.inf for i in range(n): # print "range(n):",range(n) rangemin = datamatrix[:,i].min() # x或者y坐标的最小值 rangemax = datamatrix[:,i].max() # x或者y坐标的最大值 stepsize = (rangemax-rangemin)/numsteps # 分割线移动的步长 for j in range(-1，int(numsteps)+1): #j =[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] for inqual in ['lt', 'gt']: # 分割线左边是-1 和分割线右边
```

,被分错的样本“0 1 2 9”的权值变大，其它被分对的样本的权值变小。

经过三轮迭代，得到的三个基本分类器，然后将这三个分类器使用sign函数结合在一起，得到最终的分类器，至此，整个训练过程结束，

$$G(x) = sign[f3(x)] = sign[\ a1 * G1(x) + a2 * G2(x) + a3 * G3(x)\ ]$$

$$G(x) = sign[f3(x)] = sign[\ 0.4236G1(x) + 0.6496G2(x) + 0.7514G3(x)\ ]$$

python 代码实现：

```
import math as mh

def calu_weight(y_lst,w_lst,divide):
    error=0
    zm=0.0
    em =0
    g_lst =range(10)
    w2_lst = range(10)
    for i in range(0,10):
        if i<divide:
            g_lst[i]=1
            if y_lst[i]!=1:
                em = em + w_lst[i]
        else:
            g_lst[i]=-1
            if y_lst[i]!=-1:
                em = em+w_lst[i]

    if em>0.5:
        em = 1-em
        for i in range(0, 10):
            g_lst[i]=0-g_lst[i]

    am = (1/2.0)*mh.log((1-em)/em, mh.e)

    for i in range(0,10):
        zm = zm + w_lst[i]*mh.exp(-am*(y_lst[i])*g_lst[i])
    for i in range(0, 10):
        w2_lst[i]= w_lst[i]/zm*mh.exp(-am*y_lst[i]*g_lst[i])
    return w2_lst,am,g_lst

def sign(x):
```

是-1两种情况 threshval =(rangemin+float(j)*stepsize) #分割线在[0.9,2]区间内以0.1为步长移动 predictedvals = stumpClassify(datamatrix,i,threshval,inqual) #进行分类 errarr = np.mat(ones((m,1))) #生成一个大小为5*1，元素为1的矩阵 errarr[predictedvals == labelmat] = 0 #预测的分类（predictedvals）和样本的初始分类（labelmat）一样的，也就是预测正确的，就将errarr矩阵中对应位置的元素置0 weightarr = D.T*errarr #预测正确对应errarr上的位置元素置0了，不正确的为1，错误率em等于预测错误样本的权值相加 if weightarr < minerror: minerror = weightarr bestclasest = predictedvals.copy() beststump['dim'] = i beststump['thresh'] = threshval beststump['ineq'] = inqual return beststump,minerror, bestclasest def adbBoostTrainDs(dataarr,classlabels,numit = 40): """ :param dataarr: 样本数据 :param clas

```
        if x>0:
            return 1
        elif x<0:
            return -1
if __name__ == "__main__":
    divide =2.5
    y_hat = range(10)
    y_lst =[1,1,1,-1,-1,-1,1,1,1,-1]
    w_lst = [0.1,0.1, 0.1, 0.1, 0.1, 0.1
, 0.1, 0.1, 0.1, 0.1]
    w2_lst,am1,g_lst1 = calu_weight(y_ls
t,w_lst,divide)
    print "first:"
    print "am1=%.4f" %am1
    print "g_lst1: ",g_lst1
    divide=8.5
    w3_lst,am2,g_lst2 = calu_weight(y_ls
t,w2_lst,divide)
    print "second:"
    print "am2=%.4f" %am2
    print "g_lst2: ", g_lst2
    divide = 5.5
    w4_lst,am3,g_lst3 = calu_weight(y_ls
t, w3_lst, divide)
    print "second:"
    print "am3=%.4f" %am3
    print "g_lst3: ", g_lst3

    for i in range(0,10):
        y_hat[i] = sign(am1*g_lst1[i]+am
2*g_lst2[i]+am3*g_lst3[i])
    print "y_har: ",y_hat
```

```
first:
am1=0.4236
g_lst1:  [1, 1, 1, -1, -1, -1, -1, -1, -1, -1]
second:
am2=0.6496
g_lst2:  [1, 1, 1, 1, 1, 1, 1, 1, 1, -1]
second:
am3=0.7520
g_lst3:  [-1, -1, -1, -1, -1, -1, 1, 1, 1, 1]
y_har:  [1, 1, 1, -1, -1, -1, 1, 1, 1, -1]
```

代码解析：

slabels: 样本分类值 :param numit: 迭代次数 :return: """ werkclassarr = [] m = np.shape(dataarr)[0] #样本数据多少个 D = np.mat(np.ones((m,1))/m) # 初始权值 aggclassest = np.mat(np.zeros((m,1))) for i in range(numit): beststump,error,classest = buildStump(dataarr, classlabels, D) alpha = float(0.5*np.log((1.0-error)/error)) #计算当前分类器的权值 beststump['alpha'] = alpha werkclassarr.append(beststump) #更新样本权值 expon = np.multiply(-1*alpha*np.mat(classlabels).T,classest) D = np.multiply(D, np.exp(expon)) D = D/D.sum() print "D:",D.T aggclassest+=alpha*classest aggerrors = np.multiply(np.sign(aggclassest)!= np.mat(classlabels).T,np.ones((m,1))) errorrate = aggerrors.sum()/m if(errorrate==0.0): break return werkclassarr if __name__ == "__main__": D = np.mat(np.ones((5,1))/5) datmat,labelmat= loa

误差率em计算：

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

```
for i in range(0, 10):
    if i<divide:
        g_lst[i]=1
        if y_lst[i]!=1:
            em = em + w_lst[i]
    else:
        g_lst[i]=-1
        if y_lst[i]!=-1:
            em = em+w_lst[i]
```

分类器权值计算：

$$a_m = \frac{1}{2} \ln \frac{1 - e_m}{e_m}$$

```
am = (1/2.0)*mh.log((1-em)/em, mh.e)
```

规范化因子Zm的计算：

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

更新训练样本的权值：

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

```
for i in range(0, 10):
    w2_lst[i]= w_lst[i]/zm*mh.exp(-am*y_lst[i]*g_lst[i])
```

Python实现Adaboost

```
# -*- coding=UTF-8 -*-
import numpy as np
```

dSimData() beststump,minerror,bestclasest = buildStump(datmat,labelmat,D) print "beststump:",beststump print "minerror:",np.mat(minerror).T print "bestclasest:",np.mat(bestclasest).T adbBoostTrainDs(datmat, labelmat, 10)

```
from numpy import
ones

def loadSimData():
    datMat = np.matr
ix([[1.0,2.1],

[2.0,1.1],

[1.3,1.0],

[1.0,1.0],

[2.0,1.0]])
    classLabels = [1
.0,1.0,-1.0,-1.0,1.0
]
    return datMat,
classLabels
def stumpClassify(da
taMatrix,dimen,
threshval,
threshIneq):
    """
    :param dataMatri
x: 训练样本
```

```
        :param dimen: 维
度（x轴或y轴）
        :param threshval
: 阈值（分割线）
        :param threshIne
q: 决定分割线左边是-1
和分割线右边是-1两种情
况
        :return: 返回一个
分类好的np.array
        """
        retArray = np.on
es((np.shape(dataMat
rix)[0],1))
        if threshIneq ==
'lt':
            retArray[dat
aMatrix[:,dimen] <=
threshval] = -1.0
        else:
            retArray[dat
aMatrix[:,dimen] >
threshval] = -1.0
        return retArray
    def buildStump(dataA
rr, classLabels,D):
```

```
    datamatrix = np.mat(dataArr)  #转换成矩阵，方便进行运算
    labelmat = np.mat(classLabels).T  #转置成列向量，因为样本数据就是列向量格式
    m,n=np.shape(datamatrix)    #获取dataMatrix的行，列数
    numsteps = 10.0 # 分割线移动多少次
    beststump = {}
    bestclasEst = np.mat(np.zeros((m,1)))
    minerror =np.inf
    for i in range(n):
        # print "range(n):",range(n)
        rangemin = datamatrix[:,i].min()  # x或者y坐标的最小值
        rangemax =
```

```
        datamatrix[:,i].max()  # x或者y坐标的最大值
        stepsize = (rangemax-rangemin)/numsteps  # 分割线移动的步长
        for j in range(-1, int(numsteps)+1):  #j =[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
            for inqual in ['lt','gt']:    # 分割线左边是-1和分割线右边是-1两种情况

            threshval =(rangemin+float(j)*stepsize) #分割线在[0.9,2]区间内以0.1为步长移动

            predictedvals = stumpClassify(datamatrix,i,threshval,inq
```

```
ual)  #进行分类


errarr = np.mat(ones((m,1)))  #生成一个大小为5*1，元素为1的矩阵

errarr[predictedvals == labelmat] = 0  #预测的分类（predictedvals）和样本的初始分类（labelmat）一样的，也就是预测正确的，就将errarr矩阵中对应位置的元素置0

weightarr = D.T*errarr      #预测正确对应errarr上的位置元素置0了，不正确的为1，  错误率em等于预测错误样本的权值相加

                            if weightarr < minerror:
```

```
        minerror = weightarr

        bestclasest =
        predictedvals.copy()

        beststump['dim'] = i

        beststump['thresh']
        = threshval

        beststump['ineq'] =
        inqual
    return beststump
    ,minerror,
    bestclasest

    def adbBoostTrainDs(
    dataarr,classlabels,
    numit = 40):
        """

        :param dataarr:
    样本数据
        :param classlabe
    ls: 样本分类值
```

```
        :param numit: 迭
代次数
        :return:
        """
        werkclassarr = [
]
        m = np.shape(dat
aarr)[0]
#样本数据多少个
        D = np.mat(np.on
es((m,1))/m)
#初始权值
        aggclassest = np
.mat(np.zeros((m,1))
)
        for i in range(n
umit):
            beststump,er
ror,classest =
buildStump(dataarr,
classlabels, D)
            alpha = floa
t(0.5* np.log((1.0-e
rror)/error))  #计算
当前分类器的权值
            beststump['a
```

```
lpha'] = alpha
                werkclassarr
.append(beststump)

                #更新样本权值
                expon = np.m
ultiply(-1*alpha*np.
mat(classlabels).T,c
lassest)
                D = np.multi
ply(D,np.exp(expon))
                D = D/D.sum(
)

                print "D:",D
.T

                aggclassest
+=alpha*classest

                aggerrors =
np.multiply(np.sign(
aggclassest)!= np.ma
t(classlabels).T,np.
ones((m,1)))
                errorrate =
```

```
        aggerrors.sum()/m

            if(errorrate
==0.0):
                break
    return
werkclassarr
```

```
if __name__ == "__ma
in__":
    D = np.mat(np.on
es((5,1))/5)
    datmat,labelmat=
loadSimData()
    beststump,minerr
or,bestclasest =
buildStump(datmat,la
belmat,D)
    print "beststump
:",beststump
```

```
        print "minerror:
",np.mat(minerror).T
        print "bestclase
st:",np.mat(bestclas
est).T

        adbBoostTrainDs(
datmat, labelmat, 10
)
```

Pre: 机器学习笔记（零）——机器学习初探
Next: 机器学习笔记（三）——决策树和随机森林

0 comments