# 统计学习方法学习笔记

记录（不限于）以下内容：

算法描述、自己的理解、注意事项。

代码实现，包括代码技术细节，例如某个功能如何实现。

老孙 [bzxysjl@163.com](mailto:bzxysjl@163.com) 2018年7月

# 朴素Bayes估计

代码

# Bayes估计 bzxysjl@163.com

import numpy as np

from collections import defaultdict

def tranning(data:np.ndarray, labels:np.ndarray):

"""

bayes训练

:param data: 训练样本（每行为一个样本）

:param labels: 对应的分类

:return:

"""

dimension = data.shape[1]

data\_size = data.shape[0]

priori = {} # 先验概率

conditional = {} # 条件概率

# 以下循环统计次数

for i in range(data\_size):

priori[labels[i]] = priori.get(labels[i],0)+1

if labels[i] not in conditional:

conditional[labels[i]] = {}

for j in range(dimension):

if j not in conditional[labels[i]]:

conditional[labels[i]][j] = {}

k = data[i,j]

conditional[labels[i]][j][k] = conditional[labels[i]][j].get(k,0)+1

# 将次数除以总次数得到对应的概率

y = 1.0 # 对应课本P51的λ值

for i in conditional:

for j in range(dimension):

for k in conditional[i][j]:

conditional[i][j][k] = (conditional[i][j][k]+y) / (priori[i]+len(conditional[i][j])\*y)

for k in priori:

priori[k]=(priori[k]+y)/(data\_size+len(priori)\*y)

return priori, conditional

def classify\_bayes(sample:np.ndarray, priori:dict, conditional:dict):

"""

bayes分类

:param sample: 待分类样本

:param priori: 先验概率

:param conditional: 条件概率

:return: (分类,概率)

"""

max\_probability = -1

max\_label = None

for c in priori:

probability = priori[c]

for j in range(len(sample)):

probability = probability \* conditional[c][j][sample[j]]

if probability>max\_probability:

max\_probability = probability

max\_label = c

return max\_label, max\_probability

def test\_bayes():

# 测试数据来源于《统计学习方法》P50例4.1

all =np.array([[1,'s',-1],[1,'m',-1],[1,'m',1],[1,'s',1],[1,'s',-1],

[2,'s',-1],[2,'m',-1],[2,'m',1],[2,'l',1],[2,'l',1],

[3,'l',1],[3,'m',1],[3,'m',1],[3,'l',1],[3,'l',-1]])

data = all[:,0:2]

labels = all[:,-1]

a,b = tranning(data,labels)

print('training result is:')

print(a)

print(b)

print('test a sample')

sample = np.array([2,'s'])

lable, probability = classify\_bayes(sample, a, b)

print('smaple ',sample ,' is classified as ',lable,' with probability',probability)

# k近邻kNN(20180801)

## kNN代码

### knn排序

版本1.最简单直观的kNN算法，见《机器学习实战》

import numpy as np

import operator

def knn(sample, dataSet, labels, k):

"""

kNN（k近邻）算法，根据最近的k个点的大多数类别来确定要识别样本的类别。

:param sample: 要识别的样本

:param dataSet: 训练样本

:param labels: 训练样本对应的分类

:param k: k值，即取多少个近邻

:return: 要识别的样本所归属的分类

"""

dataSetSize = dataSet.shape[0]

diffMat = np.tile(sample, (dataSetSize,1)) - dataSet

sqDiffMat = diffMat\*\*2

sqDistance = sqDiffMat.sum(axis=1)

distance = sqDistance\*\*0.5

sortedDistance = distance.argsort()

classCount={}

for i in range(k):

label = labels[sortedDistance[i]]

classCount[label] = classCount.get(label,0)+1

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

return sortedClassCount[0][0]

### 归一化

def autoNorm(dataset):

"""

归一化，将样本值归一化成0到1之间的值

:param dataset:样本数据

:return:归一化后的样本数据

"""

minValues = dataset.min(0)

maxValues = dataset.max(0)

rangeValues = maxValues - minValues

n = dataset.shape[0]

diffMat = dataset - np.tile(minValues, (n, 1))

diffMat = diffMat / np.tile(rangeValues, (n, 1))

return diffMat, rangeValues, minValues

### 读取文本文件转换为ndarray

def datingFile2Matrix(filename):

fr = open(filename)

lines = fr.readlines()

lineCount = len(lines)

dataSet = np.zeros((lineCount, 3))

labels = []

index = 0

for line in lines:

line = line.strip()

words = line.split('\t')

dataSet[index, :] = words[0:3]

labels.append(int(words[-1]))

index += 1

return dataSet, labels

### matplotlib显示2维向量点图

def datingDataFigure():

a, b = datingFile2Matrix('data/datingTestSet2.txt')

fig = plt.figure()

ax = fig.add\_subplot(111)

ax.scatter(a[:, 0], a[:, 1], 30, np.array(b), '.')

plt.show()

### knn手写数字识别

# 以下为knn手写数字识别

def knn\_digit():

"""

用knn进行手写数字识别。以trainingDigits文件夹中的文件为分类样本，对testDigits文件夹的文件进行分类测试。

:return:

"""

path = './data/digits/trainingDigits/'

files = os.listdir(path)

traningSize = len(files)

dataset = np.zeros((traningSize, 1024))

labels = np.empty(traningSize)

for i in range(traningSize):

dataset[i, :], labels[i] = digitText2matrix(path + files[i])

path = './data/digits/testDigits/'

files = os.listdir(path)

testSize = len(files)

error = 0

for i in range(testSize):

sample, label = digitText2matrix(path+files[i])

cls = knn(sample, dataset, labels, 5)

if label != cls:

print('sample %s:\t should be %d got %d' % (files[i], label, cls))

error += 1

print('error rate is %f' % (error\*1.0/testSize))

return 0

def digitText2matrix(file):

"""

读取手写数字文本文件（内容为32和32列的0和1），返回1024维向量

:param file:

:return:

"""

fr = open(file)

lines = fr.readlines()

fr.close()

mat = np.zeros(1024)

for row in range(32):

line = lines[row]

for column in range(32):

mat[row\*32 + column] = int(line[column])

label = os.path.split(file)[1].split('\_')[0]

return mat, int(label)

def plotDigit(file):

"""

用matplotlib显示手写数字的二值图像

:param file:

:return:

"""

mat, label = digitText2matrix(file)

mat2 = np.zeros((1024, 3))

for i in range(32):

for j in range(32):

index = i \* 32 + j

mat2[index, 0] = 32 - i

mat2[index, 1] = j

mat2[index, 2] = mat[index]

fig = plt.figure()

axis = fig.add\_subplot(1, 1, 1)

y = mat2[:,0]

x = mat2[:,1]

v = mat2[:,2]

axis.scatter(x, y, v+1, v)

plt.show()

## 技术细节

### tile平铺

numpy.tile(A, reps)

Construct an array by repeating A the number of times given by reps.

If reps has length d, the result will have dimension of max(d, A.ndim).

If A.ndim < d, A is promoted to be d-dimensional by prepending new axes. So a shape (3,) array is promoted to (1, 3) for 2-D replication, or shape (1, 1, 3) for 3-D replication. If this is not the desired behavior, promote A to d-dimensions manually before calling this function.

If A.ndim > d, reps is promoted to A.ndim by pre-pending 1’s to it. Thus for an A of shape (2, 3, 4, 5), a reps of (2, 2) is treated as (1, 1, 2, 2).

### argsort下标排序

numpy.argsort(a, axis=-1, kind='quicksort', order=None)

Returns the indices that would sort an array.

Perform an indirect sort along the given axis using the algorithm specified by the kind keyword. It returns an array of indices of the same shape as a that index data along the given axis in sorted order.

### operator模块

This module exports a set of functions corresponding to the intrinsic operators of Python. For example, operator.add(x, y) is equivalent to the expression x+y. The function names are those used for special methods; variants without leading and trailing '\_\_' are also provided for convenience.

\_\_all\_\_ = ['abs', 'add', 'and\_', 'attrgetter', 'concat', 'contains', 'countOf',

'delitem', 'eq', 'floordiv', 'ge', 'getitem', 'gt', 'iadd', 'iand',

'iconcat', 'ifloordiv', 'ilshift', 'imatmul', 'imod', 'imul',

'index', 'indexOf', 'inv', 'invert', 'ior', 'ipow', 'irshift',

'is\_', 'is\_not', 'isub', 'itemgetter', 'itruediv', 'ixor', 'le',

'length\_hint', 'lshift', 'lt', 'matmul', 'methodcaller', 'mod',

'mul', 'ne', 'neg', 'not\_', 'or\_', 'pos', 'pow', 'rshift',

'setitem', 'sub', 'truediv', 'truth', 'xor']

### Matplotlit绘图

（1）add\_subplot

#引入对应的库函数

import matplotlib.pyplot as plt

from numpy import \*

#绘图

fig = plt.figure()

ax = fig.add\_subplot(349)

#参数349的意思是：将画布分割成3行4列，图像画在从左到右从上到下的第9块

ax.plot(x,y)

plt.show()

(2)scatter

Axes.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, \*, data=None, \*\*kwargs)[source]

A scatter plot of y vs x with varying marker size and/or color.

Parameters:

x, y : array\_like, shape (n, )

The data positions.

s : scalar or array\_like, shape (n, ), optional

The marker size in points\*\*2. Default is rcParams['lines.markersize'] \*\* 2.

c : color, sequence, or sequence of color, optional, default: ‘b’

The marker color. Possible values:

A single color format string.

A sequence of color specifications of length n.

A sequence of n numbers to be mapped to colors using cmap and norm.

A 2-D array in which the rows are RGB or RGBA.

Note that c should not be a single numeric RGB or RGBA sequence because that is indistinguishable from an array of values to be colormapped. If you want to specify the same RGB or RGBA value for all points, use a 2-D array with a single row.

marker : MarkerStyle, optional, default: ‘o’

The marker style. marker can be either an instance of the class or the text shorthand for a particular marker. See markers for more information marker styles.

N = 50

x = np.random.rand(N)

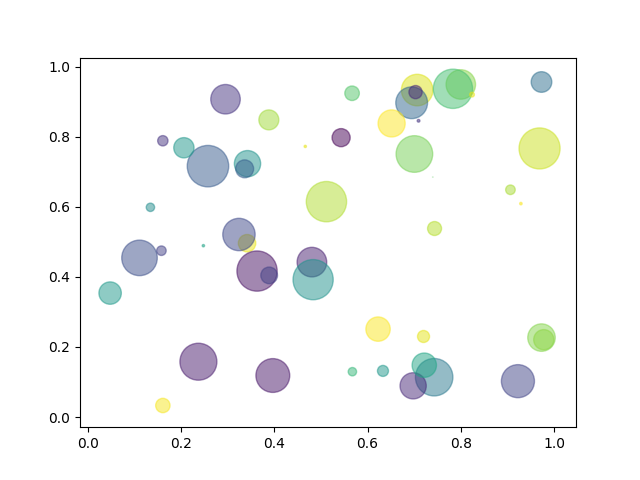
y = np.random.rand(N)

colors = np.random.rand(N)

area = (30 \* np.random.rand(N))\*\*2 # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)

#结果如下图



### Ndarray的sort和python内置sorted区别

Numpy的ndarray的sort方法按照某一行或者列（axis）进行排列，并不保持整行或者整列数据的完整性；而python内置sorted方法保持最大元组的完整性（对于二维来说，就是保持行的完整性，不会破坏一行），可根据某一列的值进行排序，sorted方法行为与C#和Java的排序行为相同。

而且sort方法是原地(in place)排序，而sorted是重新返回一个排序后的列表，不影响原来的列表。

通过下例来看二者区别。

>>>a = np.array([[100,26,33],[61,509,124],[120,90,20]])

>>>a

array([[100, 26, 33],

[ 61, 509, 124],

[120, 90, 20]])

>>>a.sort(axis=0)

>>>a

array([[ 61, 26, 20],

[100, 90, 33],

[120, 509, 124]])

>>>a.sort(axis=1)

>>>a

array([[ 20, 26, 61],

[ 33, 90, 100],

[120, 124, 509]])

>>>a = np.array([[100,26,33],[61,509,124],[120,90,20]])

>>>a

array([[100, 26, 33],

[ 61, 509, 124],

[120, 90, 20]])

>>>sorted(a, key = operator.itemgetter(0) )

[array([ 61, 509, 124]), array([100, 26, 33]), array([120, 90, 20])]

>>>sorted(a, key = operator.itemgetter(1) )

[array([100, 26, 33]), array([120, 90, 20]), array([ 61, 509, 124])]

## Kd树(20180806)

见《统计学习方法》P41

Kd树的实现有现成的库scikit-learn：<http://scikit-learn.org/stable/index.html>

### 构建kd树（递归法）

class KDNode:

"""

kd树结点类

"""

def \_\_init\_\_(self, element=None, dimension=0, left=None, right=None):

self.element = element

self.dimension = dimension

self.left = left

self.right = right

def createKdTree(dataset, depth=0):

"""

创建kd树（递归方法）

:param dataset: 数据集

:param depth: 深度

:return: 根结点

"""

if dataset is None or len(dataset) == 0:

return None

shape0 = dataset.shape

current = depth % shape0[1]

count = shape0[0]

dataset = np.array(sorted(dataset, key=operator.itemgetter(current)))

median = int(count/2)

node = KDNode(dataset[median], current)

if median > 0:

node.left = createKdTree(dataset[0:median], depth+1)

if median < dataset.shape[0]:

node.right = createKdTree(dataset[median+1:], depth+1)

return node

### 打印kd树

def printKdTree(root):

"""

打印kd树

:param root:根结点

:return: 无

"""

if root is None:

print("None")

return

if not isinstance(root, KDNode):

raise Exception("类型不正确，必须是KDNode类型")

return

print('(\t')

print(root.element)

print('left child:')

printKdTree(root.left)

print('right child:')

printKdTree(root.right)

print(')')

### 搜索kd树得到最近邻

def searchKdTree(root: KDNode, element: np.ndarray):

"""

搜索kd树（递归法）得到最近邻(Nearest Neighbour)

:param root: 根结点

:param element: 要查找的元素

:return: tuple(结点,距离)

"""

if root is None:

return None, np.inf

# 先走到叶子结点

current = root

path = []

nearest = None

min\_distance = 99999999

while current is not None:

v1 = current.element[current.dimension]

v2 = element[current.dimension]

distance = np.linalg.norm(current.element - element)

if distance < min\_distance:

min\_distance = distance

nearest = current

print('current nearest :\t {0} with {1:.2f}'.format(current.element, min\_distance))

if distance == 0:

return nearest, min\_distance

path.append(current)

if v2 < v1:

current = current.left

else:

current = current.right

while len(path) > 0 and min\_distance > 0:

current = path.pop()

child = None

if abs(current.element[current.dimension] - element[current.dimension]) >= min\_distance:

continue

if element[current.dimension] < current.element[current.dimension]:

child = current.left

else:

if element[current.dimension] > current.element[current.dimension]:

child = current.right

if not child:

continue

path.append(child)

distance = np.linalg.norm(child.element - element)

if distance < min\_distance:

min\_distance = distance

nearest = child

print('current nearest :\t {0} with {1:.2f}'.format(child.element, min\_distance))

return nearest, min\_distance

### 测试kd树

def testCreateKdTree():

dataset = np.array([[2,3],[5,4],[9,6],[4,7],[8,1],[7,2]])

root = createKdTree(dataset, 0)

printKdTree(root)

def testSearchKdTree():

# dataset = np.array([[2, 3], [5, 4], [9, 6], [4, 7], [8, 1], [7, 2]])

dataset = np.random.rand(20).reshape(10, 2)\*10

print('dataset is:\r\n', dataset)

root = createKdTree(dataset, 0)

query = np.random.rand(2)\*10

# query = np.array([5, 7])

print('query is:\r\n', query)

node, distance = searchKdTree(root, query)

print('final nearest :\r\n {0} with {1:.2f}'.format(node.element, distance))

# 感知机perceptron(20180707)

## （1）学习算法描述

感知机用于分类线性可分的两类样本。（1）二分类器；（2）线性可分。

学习到的是一个超平面，dot(w,x)+b.

学习过程：

(1)给定N个样本(x0,y0)(x1,y1)(x2,y2)…(xN,yN)，其中x是m维列向量，y是-1或+1。

(2)给定一个学习率yita，设定w和b的初值，例如可设为0.

(3)取一个样本xi，计算对应的yi0=w\*xi+b

(4)若yi0的符号与yi相同（即yi0\*yi>0），则继续(3)

(5)w=w+yita\*yi\*x, b=b+yita\*yi

(6)若有误分类点，转3，否则学习结束。

要考虑如果样本线性不可分，那么应该在适当的时候结束学习（例如超过一定次数）。

## （2）python代码实现

函数名：perceptron

输入参数：x(二维数组，以一列表示一个样本),y(一维行向量)

输出：w和b

代码如下：

# 感知机学习算法

# sunjilei 20180708

import numpy as np

def perceptron(x, y, rate):

"""

x为样本数，二维数组，其中每列为一个样本

y为对应的分类，一维数组

返回三元组（是否成功，w，b）

"""

# 参数检查（数组维数和长度）

xshape, yshape = checkShape(x, y)

# 初始化w和b

w = np.zeros(xshape[0])

b = 0

xt = x.T

mistake = True # 是否有误分类样本

count = 0 # 总迭代次数，超过一定数量时认为样本非线性可分，退出算法

while mistake:

if count > 10000:

return False, w, b

mistake = False

# 对样本（x的列）循环，找到一个误分类数据

for i in range(xshape[1]):

sample = xt[i]

if y[i]\*(np.dot(sample, w)+b) <= 0:

mistake = True

w = w + rate \* y[i]\*sample

b = b + rate \* y[i]

count = count + 1

print("第{0}次迭代：x={1},w={2},b={3}".format(count, sample, w, b))

return True, w, b

def perceptron\_dual(x, y, rate):

"""

感知机算法对偶形式

x为样本数，二维数组，其中每列为一个样本

y为对应的分类，一维数组

返回三元组（是否成功，alpha，b）

"""

# 参数检查（数组维数和长度）

xshape, yshape = checkShape(x, y)

# 初始化alpha和b

alpha = np.zeros(xshape[1])

b = 0

xt = x.T

mistake = True # 是否有误分类样本

count = 0 # 总迭代次数，超过一定数量时认为样本非线性可分，退出算法

# 计算gram矩阵

gram = np.zeros((xshape[1], xshape[1]))

for i in range(xshape[1]):

for j in range(xshape[1]):

gram[i][j ] = np.dot(xt[i], xt[j])

print('gram is \r\n{0}', gram)

while mistake:

if count > 10000:

return False, alpha, b

mistake = False

# 对样本（x的列）循环，找到一个误分类数据

for i in range(xshape[1]):

sample = xt[i]

temp = np.zeros(xshape[0])

for j in range(xshape[1]):

temp = temp + alpha[j] \* y[j] \* xt[j]

if y[i]\*(np.dot(temp, sample)+b) <= 0:

mistake = True

alpha[i] = alpha[i] + 1

b = b + rate \* y[i]

count = count + 1

print("第{0}次迭代：x={1},alpha={2},b={3}".format(count, sample, alpha, b))

return True, alpha, b

def checkShape(x, y):

xshape = x.shape

yshape = y.shape

if not(isinstance(x, np.ndarray) or isinstance(y, np.ndarray)):

raise Exception("参数类型不正确，必须为ndarray")

if x.ndim != 2 or y.ndim != 1:

raise Exception("数组维数不正确")

if xshape[1] != yshape[0]:

raise Exception("x,y个数不匹配")

return xshape, yshape

def main():

n = int(input("请输入样本数量："))

m = int(input("请输入样本维数："))

x = np.zeros((n, m))

y = np.zeros(n)

for i in range(n):

sample = input("第{0}个样本值（逗号隔开）：".format(i+1))

t = np.fromstring(sample, dtype=float, sep=',')

if t.size != m:

raise Exception("样本维数不正确")

x[i] = t

y[i] = int(input("第{0}个分类（1或-1)：".format(i+1)))

# result = perceptron(x.T, y, 1.0)

result = perceptron\_dual(x.T, y, 1.0)

if result[0]:

print("分类成功。w={0},b={1}".format(result[1],result[2]))

else:

print("分类失败。")

if \_\_name\_\_ == "\_\_main\_\_":

main()

运行情况

请输入样本数量：3

请输入样本维数：2

第1个样本值（逗号隔开）：3,3

第1个分类（1或-1)：1

第2个样本值（逗号隔开）：4,3

第2个分类（1或-1)：1

第3个样本值（逗号隔开）：1,1

第3个分类（1或-1)：-1

第1次迭代：x=[3. 3.],w=[3. 3.],b=1.0

第2次迭代：x=[1. 1.],w=[2. 2.],b=0.0

第3次迭代：x=[1. 1.],w=[1. 1.],b=-1.0

第4次迭代：x=[1. 1.],w=[0. 0.],b=-2.0

第5次迭代：x=[3. 3.],w=[3. 3.],b=-1.0

第6次迭代：x=[1. 1.],w=[2. 2.],b=-2.0

第7次迭代：x=[1. 1.],w=[1. 1.],b=-3.0

分类成功。w=[1. 1.],b=-3.0

## （3）编码细节

### Numpy数组（矩阵）

Numpy创建数组： a = np.array([[1,2],[3,4]])

a.ndim返回数组维数

获取数组大小： b = a.size

a.size数组大小返回一个tuple对象b

b[0]是tuple的0分量，b[1]是1分量。

### Numpy矩阵乘法

Python中的几种矩阵乘法

1. 同线性代数中矩阵乘法的定义： np.dot()

np.dot(A, B)：对于二维矩阵，计算真正意义上的矩阵乘积，同线性代数中矩阵乘法的定义。对于一维矩阵，计算两者的内积。

2. 对应元素相乘 element-wise product: np.multiply(), 或 \*

参考：<https://blog.csdn.net/u012609509/article/details/70230204>

### Numpy矩阵的轴axis

对于2维矩阵，有2个axis，axis 0是逐行从上往下的1列，axis 1是逐列从左往右的1行。

Axes are defined for arrays with more than one dimension. A 2-dimensional array has two corresponding axes: the first running vertically downwards across rows (axis 0), and the second running horizontally across columns (axis 1).

a = np.array([[1, 2, 3], [4, 5, 6]])

a0 = np.sum(a,axis=0)

a0

array([5, 7, 9])

a1 = np.sum(a,axis=1)

a1

array([ 6, 15])

还有一个很好的参考

<https://blog.csdn.net/qq_29573053/article/details/76998695>

### Python类型

检查变量类型

To check if o is an instance of str or any subclass of str, use isinstance (this would be the "canonical" way):

if isinstance(o, str):

To check if the type of o is exactly str:

if type(o) is str:

### Python异常

Raise抛出异常

### 在[ ]中写代码创建列表

Python一个非常方便实用的语法来创建列表。

例1.基于列表创建另外一个列表

d = [1,2,3]

f = [ e+1 for e in d]

# f now is [2, 3, 4]

例2.列表元素过滤

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]])

a0 = a[ np.sum(a,axis=1) >6 ]

a0

array([[4, 5, 6]])