# 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

# 20180801 k近邻kNN

## kNN代码

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 createDataSet():

group = np.array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])

labels = ['A','A','B','B']

return group,labels

def classify0(sample,dataSet,labels,k):

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]

def simpleKnnTest():

group,labels=createDataSet()

r = classify0([0,0],group,labels,3)

print('class is ',r)

## 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)

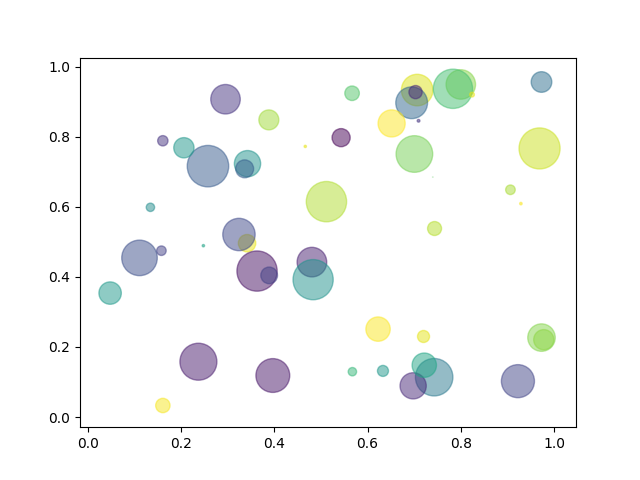
y = np.random.rand(N)

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



# 20180707感知机perceptron

## （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]])