目录

[Python教程 3](#_Toc62233539)

[python教程： 3](#_Toc62233540)

[1. 如何使用全局变量： 3](#_Toc62233541)

[2. 如何获得list中某个元素的序号：index=list.index(‘内容’) 3](#_Toc62233542)

[3. 如何将list中的每个元素都执行相同的操作，比如有一个list面存的都是datetime类型，你想将datetime转成str类型，怎么操作？ 4](#_Toc62233543)

[4. 如何求取两个list的并集： 4](#_Toc62233544)

[5. 5, 如何赋空值？ A=np.nan 4](#_Toc62233545)

[6. 6, 如何连接list 4](#_Toc62233546)

[7. 如果要在一行写多个代码： 4](#_Toc62233547)

[8. list转np：a=np.array(b) 5](#_Toc62233548)

[9. 向量的內积也就是点乘： 5](#_Toc62233549)

[10. 如何生成指定长度的list: 5](#_Toc62233550)

[11. 次幂： 5](#_Toc62233551)

[12. 设置主文件路径： 5](#_Toc62233552)

[Numpy教程： 5](#_Toc62233553)

[13. np array的拼接： 5](#_Toc62233554)

[14. np 保存文件与加载文件: 6](#_Toc62233555)

[15. 3，np array的创建： 6](#_Toc62233556)

[16. 3，向量的比较： 8](#_Toc62233557)

[17. 4, 向量的赋值 9](#_Toc62233558)

[18. 5，nparray中最大值的位置： 9](#_Toc62233559)

[19. 6，绘制直方图： 9](#_Toc62233560)

[20. 7，np array的顺序重排说着说着洗牌 11](#_Toc62233561)

[21. 向量的逻辑运算操作： 11](#_Toc62233562)

[22. 11](#_Toc62233563)

[23. 发 11](#_Toc62233564)

[24. 发 11](#_Toc62233565)

[Matplot教程： 11](#_Toc62233566)

[Pytorch教程： 12](#_Toc62233567)

[1. **torch**.**utils**.**data**.**DataLoader**(dataset,batch\_size=64,shuffle=False(洗牌),num\_workers=0(使用几个进程)) 12](#_Toc62233568)

[2. TensorDataset(data\_tensor=x,target\_tensor=y) 13](#_Toc62233569)

[3. 如果处理多输入网络？ 13](#_Toc62233570)

[4. 如何查看tensor的shape：tensor\_a.size() 14](#_Toc62233571)

[5. 如何初始化一个tensor: 14](#_Toc62233572)

[6. 如何初始化网络中的层？ 14](#_Toc62233573)

[7. np中的array如何与torch中的tensor互相转换 17](#_Toc62233574)

[8. 关于gpu的使用 17](#_Toc62233575)

[9. device的初始化： 18](#_Toc62233576)

[10. pytorch tensor的拼接： 19](#_Toc62233577)

[11. model的参数: 19](#_Toc62233578)

[12. reshape 与 view 与transpose三者都是改变数据形状的函数。 20](#_Toc62233579)

[13. 随机数列： 20](#_Toc62233580)

[14. dropout的使用:作用是方式过拟合 20](#_Toc62233581)

[15. torch的tensor 与 np的array 相互转换 21](#_Toc62233582)

[CV2教程 21](#_Toc62233583)

[Pandas教程 22](#_Toc62233584)

[1. 创建DataFrame: 22](#_Toc62233585)

[2. 操作DataFrame 22](#_Toc62233586)

[训练经验 23](#_Toc62233587)

[1. 输入数据应该归一化,如果不做归一化处理会导致梯度计算错误nan 23](#_Toc62233588)

[2. 每次迭代前必须梯度清零 24](#_Toc62233589)

[3. 线性不要与relu一起用，会有0梯度 24](#_Toc62233590)

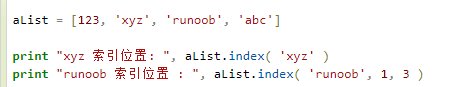
# Python教程

## python教程：

1. 如何使用全局变量：

在外面定义一个变量，然后在函数内以global重新声明一个变量。

1. 如何获得list中某个元素的序号：index=list.index(‘内容’)



1. 如何将list中的每个元素都执行相同的操作，比如有一个list面存的都是datetime类型，你想将datetime转成str类型，怎么操作？

List没有一次性转换的函数，需要对list中的每一个元素都进行一次操作，那么就要使用map命令。

B=list(map(f,a)) 它的意思是 对a中的每一个元素都用f函数运算一次得到一个b，很多个b组成一个B

1. 如何求取两个list的并集：

set(a)将a变成一个无序不重复元素及，a,union(b)

返回a，b的并集。

1. 5, 如何赋空值？ A=np.nan
2. 6, 如何连接list

Append 是将一个list作为一个元素连接起来，比如a=[1,2] b=[3,4] a=a.append(b)

A 是[1,2,[3,4]],如果要把其中的元素提取出来再连接起来就要使用a=a.extend(b),a是

[1,2,3,4]

1. 如果要在一行写多个代码：

可以用”；”分好，隔开。

1. list转np：a=np.array(b)
2. 向量的內积也就是点乘：

np.dot(a,b) a\*b是对应元素相乘

在ab是矩阵时候dot是叉乘，\*是点乘。

1. 如何生成指定长度的list:

a=[i for i in range(10)]

1. 次幂：

math.pow( x, y )

1. 获得文件夹下所有文件名称：

#获取某个文件夹下所有文件名称

import os

filePath='data/stocktestdata/'

for i,k in enumerate(os.walk(filePath)):

print(i,k[2])

1. 如何在C工程（VS）中使用Anaconda中的python

这里我们应该先说一个前提，anaconda是通过配置环境变量来实现不同的env的，这个是通过实验得到的，最关键的知道来自于[https://numpy.org/devdocs/user/troubleshooting-importerror.html#check-environment-variables](https://numpy.org/devdocs/user/troubleshooting-importerror.html" \l "check-environment-variables)。当我们使用python遇到找不到第三方库的时候应该首先关注环境变量。

1. 将要使用的python.exe找到，并将其路径设置为home目录

Py\_SetPythonHome(L"D:/software/anaconda/envs/py3.7-gpu");

1. 将python的lib include 还有numpy（在site-packages）的 lib include配置到VS的工程下，特别注意这里应该配置x64的，因为python是64bit如果不一致运行不了。（release应该是最稳定的），还有一个需要注意，配置debug的时候可能找不到python37\_d.lib这时候可以自己复制改名。
2. 执行单条python语句：PyRun\_SimpleString("import sys,os");

这里首先应该执行的就是把sys和os加载进来，因为后面需要把你的py文件的路径添加早sys.path（注意这个path是找python文件的路径和安装的第三方库的路径PATH是不同的）

4, import: PyRun\_SimpleString("from MyDLfunc\_yolo3D import\*");

这里会遇到很大的困难，单独测试python语句都好用，但是一旦使用到第三方库的话就不好用了。这就是开头说的问题，首先我们要确定PATH是否出现了错误（前提是我们在anaconda中打开的一个终端里import是一定要好用的）

使用如下命令可以打印当前的

print("PATH:", os.environ.get('PATH'))

我们分别在anaconda终端、和 VS调试中 打印PATH这个路径，对比二者我们就可以知道PATH是否相同。如果不同那么就是说明我们的环境变量错了。

然后就是修改这个PATH了，修改函数如下：

这里有大坑，因为SimpleString我们必须在最外边有““，如果里面还用””的话就要用/作为转义符号，这样做就错了，因为里面如果出现一个u开头的路径比如\usr这样就无法解析出正确路径了。正确的做法是里面的字符串用’’括起来，[\\用/](file:///\\\\用/)替代，这样就可以正确修改环境变量了。这么多环境变量我们很慢意义修改，我们先把正确的复制过来，然后用编辑工具把\改成/就可以了。

PyRun\_SimpleString("os.environ['PATH']='D:/software/anaconda/envs/py3.7-gpu;D:/software/anaconda/envs/py3.7-gpu/Library/mingw-w64/bin;D:/software/anaconda/envs/py3.7-gpu/Library/usr/bin;D:/software/anaconda/envs/py3.7-gpu/Library/bin;D:/software/anaconda/envs/py3.7-gpu/Scripts;D:/software/anaconda/envs/py3.7-gpu/bin;D:/software/anaconda/condabin;D:/software/anaconda;D:/software/anaconda/Library/mingw-w64/bin;D:/software/anaconda/Library/usr/bin;D:/software/anaconda/Library/bin;D:/software/anaconda/Scripts;C:/Program Files/Eclipse Foundation/jdk-8.0.302.8-hotspot/bin;C:/Program Files/VanDyke Software/Clients;C:/Windows/system32;C:/Windows;C:/Windows/System32/Wbem;C:/Windows/System32/WindowsPowerShell/v1.0;C:/Windows/System32/OpenSSH;C:/Program Files/dotnet;C:/Program Files/Microsoft SQL Server/130/Tools/Binn;C:/Program Files/Microsoft SQL Server/Client SDK/ODBC/170/Tools/Binn;E:/MyWorks/OpenCv/opencvDebug/install/bin;D:/software/anaconda/Scripts;D:/software/Git/cmd;C:/Users/Administrator/AppData/Local/Microsoft/WindowsApps;C:/Users/Administrator/.dotnet/tools;E:/MyWorks/OpenCv/opencvDebug/install/bin;D:/software/anaconda/Scripts;.;C:/Users/Administrator/AppData/Local/Programs/Microsoft VS Code/bin'");

输入：

输入都要用Tuple类型的，用PyTuple\_SetItem(ArgArray, 0, PyArray);可以将nparray转换成Tuple。PyObject\* PyArray = PyArray\_SimpleNewFromData(3, Dims, NPY\_UBYTE, img->imageData);用这个可以将图像数据生成numpyarray。

输出： <https://blog.csdn.net/stu_csdn/article/details/69488385>

基本上就是先判断是什么类型的输出，然后用相应的函数去解析。

特别注意：在得到pyArray的时候，通过demasion获得的row col是可能有误的，这里一定要自己清楚输出的尺寸。

特别注意:在C端制作图像的时候要4字节对齐，因为在转换array的时候是没用“空隙”的。

1. 字符串a在末尾删除一部分，这一部分是b

#字符串的尾部删除另外一个字符串

str1='a.txt';str2='.txt'

str1=str1[:len(str1)-len(str2)]

print(str1)

1. Lambda函数：是python中一种函数定义方法

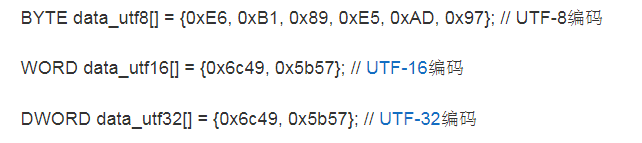
最干的一句话：lambda argument\_list:expersion

Argumentlist是参数裂变，中间用,隔开 。：后面是执行的内容；

用处：如果你需要一个简单的函数作为参数的话，单独定义一个是不是太麻烦了。

1. Python中的encode和decode
   1. 什么是unicode？称为万国码 就是国际通用的字符编码形式，它给世界上所有语言的所有字符都赋予了唯一的二进制编码。其全称是Universal Multiple-Coded CharacterSet（UCS）现在同样的是UCS-2 2个字节编码，也就是16bit可以表达2的16次方个符号。UTF-8中UTF是USC Transformation Format的缩写就是 万国码的转换格式。

比如“汉字”在USC中对应的数字是0x6c49(27721),0x5b57(23383) 那么在UTF-8 到 32中有什么区别的，现在其实就是如何储存这两个数字的问题了。

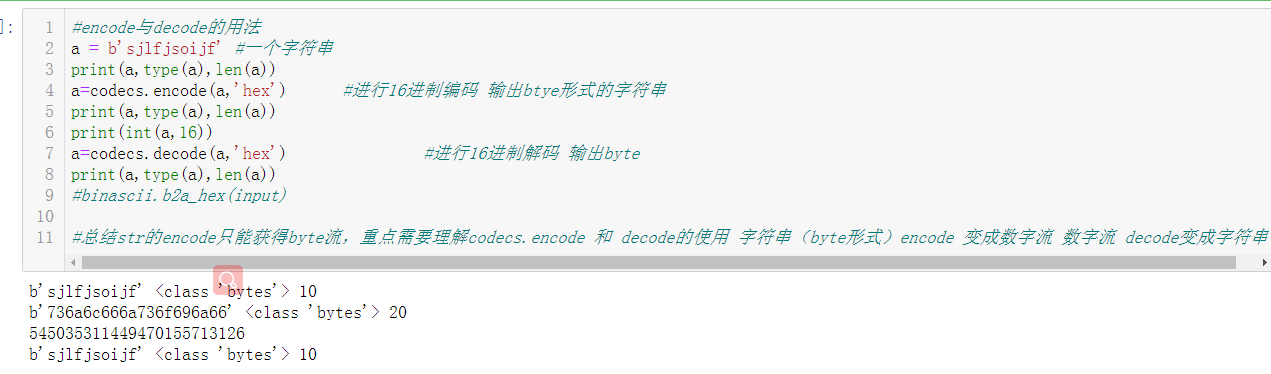


简单介绍一下UTF-8：并不是简单的储存，先判断数字的范围，如果在0x00-0x7F之间则直接储存。知道了区间然后选择使用哪个模板，确定模板以后把数字写成2进制形式并填到模板的x位置上组成最终的编码。

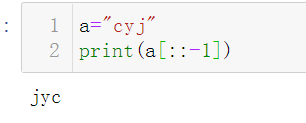
* 1. Python中字符的编码方式是UTF-8

Python中 Str -> byte 的转换称为编码str.encode 就是把字符串转换成数字并储存在一些字节中。Byte.decode()是将数字转换成字符串。

需要注意的是codecs.encode也是一种编码解码的函数，它可以转换为hex等编码方式，但是它的输入需要是byte而不是str。需要注意的是同样是byte的储存方式同样也存在“字符串”和“数字”的区别，比如：b'b901ef'这个保存的实际上是字符串而 b'\xb9\x01\xef' 这个就是实际储存的数据。字符串里面实际是用UTF码保存的，二者本质是不同的。比如b'b901ef' 世界使用codecs.encode(b'b901ef','hex') 得到的是b'623930316566'。而 codecs.encode(b'\xb9\x01\xef','hex')得到的就是b'b901ef'。Str.encode是把



1. A[::-1]的意义：逆序切片



1. 语法及运算符:
   1. python / 和 % 和 //（地板除）用于对数据进行除法运算。

python中与除法相关的三个运算符是// 和 / 和 %，下面逐一介绍。  
“/”，这是传统的除法，3/2=1.5  
“//”，在python中，这个叫“地板除”，3//2=1  
“%”，这个是取模操作，也就是区余数，4%2=0，5%2=1

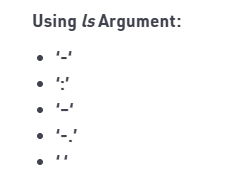
* 1. d

1. plot

plot 功能主要是可视化数据，最常用的就是画函数曲线和散点图。

1，曲线：Plt.plot(x,y,ls=’-’,lw=2,label=’plot figure’)

X,y:就是一般意义上的自变量和因变量，或者说是x轴和y轴上的数据。

Ls 表示线的风格

Lw 表示线的宽度

Label 表示标签文字

PS：

1.1，如果要设定描画时的取值范围需要用：plt.ylim(min，max);

1.2, 可以用color=‘green’来设定颜色。

2，散点图：

from pylab import plot, show

plot(x,y,样式)

eg：

plot(data[target=='setosa',0],data[target=='setosa',2],'bo')#data来自一个数据集

plot(data[target=='versicolor',0],data[target=='versicolor',2],'r+')

plot(data[target=='virginica',0],data[target=='virginica',2],'g\*')

3，是用figure画图：

3.1 创建画板 figure ：

figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True)

num:图像编号或名称，数字为编号 ，字符串为名称

figsize:指定figure的宽和高，单位为英寸；

dpi

facecolor:背景颜色

edgecolor:边框颜色

frameon:是否显示边框

2，创建子图：

2.1 plt.subplot(221)创建 2x2个子图 现在描画第一个 然后 plt.subplot(222)描画第二个；

2.2 fig,axlist=plt.subplots(subplotNum,subplotNum,figsize=(20,20),dpi=300) 创建子图矩阵 axlist

然后 ax=axlist[int(i/subplotNum)][int(i%subplotNum)]

ax.cla()

在然后 ax.plot(x,Open,linewidth=lw);

1. Isinstance（object，type）：判断object是否为指定的type
2. 时间datetime：import datetime
3. 获取当前时间：curr\_time = datetime.datetime.now()
4. 时间转换为字符串：最简单的就是str(curr\_time)这样会消失到微妙；也可以用符号获取想要的部分，
5. 字符串转时间，

d = datetime.datetime.strptime(timestr,'%Y-%m-%d %H:%M:%S.%f') f是毫秒

d = datetime.datetime.strptime(timestr,'%Y-%m-%d %H\_%M\_%S')

1. 大小写转换：

#小写转大写

coinname = 'btc'

coinname = coinname.upper()

print(coinname)

s.upper() #把所有字符中的小写字母转换成大写字母

s.lower() #把所有字符中的大写字母转换成小写字母

s.capitalize() #把第一个字母转化为大写字母，其余小写

s.title() #把每个单词的第一个字母转化为大写，其余小写

1. 如何找到一个浮点数在一个等差整数列（由小到大）中的位置？那么时间序列中呢

f= 102.52

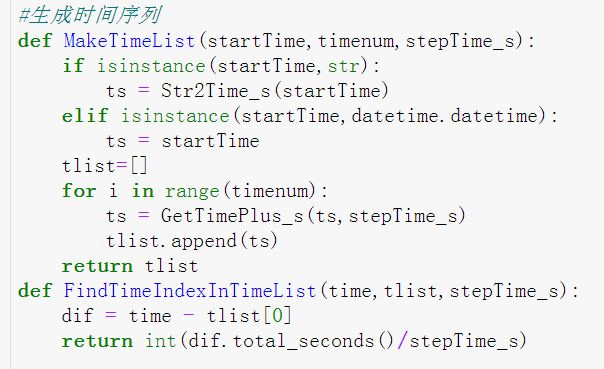
inlist = np.arange(0,200,5)

print(inlist,f)

index = int((f-inlist[0])/(inlist[1]-inlist[0]))

print(inlist[index])

#print(TradeList)



1. 制作定时器：

#计算时间差

import datetime,time

timestart = datetime.datetime.now()

while 1:

timenow = datetime.datetime.now()

time.sleep(1) #等待一秒钟

if (timenow-timestart).total\_seconds() > 10: #10是定时的时间大小，每这么长时间作一次。

timestart = timenow

print(timenow)

#将函数当做参数输入定时器

# def TimerProcesser(timestep\_s,searchtimestep,fuc):

# import datetime,time

# timestart = datetime.datetime.now()

# while 1:

# timenow = datetime.datetime.now()

# time.sleep(searchtimestep) #等待一秒钟

# if (timenow-timestart).total\_seconds() > timestep\_s:

# timestart = timenow

# fuc()

1. Dict的使用

Dict是字典，可以理解为一张表，但是索引不是数字而是str，比如“姓名”，这个属性称为键值。

初始化：dict = {['Name']: 'Zara', 'Age': 7} Name 和 Age就是键值 Zara 和 7 是数据

简单操作：

del dict['Name'] # 删除键是'Name'的条目

dict.clear() # 清空字典所有条目

del dict # 删除字典

[cmp(dict1, dict2)](https://www.runoob.com/python/att-dictionary-cmp.html) 比较两个字典元素。

[dict.copy()](https://www.runoob.com/python/att-dictionary-copy.html) 返回一个字典的浅复制

[dict.items()](https://www.runoob.com/python/att-dictionary-items.html) 以列表返回可遍历的(键, 值) 元组数组

[dict.keys()](https://www.runoob.com/python/att-dictionary-keys.html) 以列表返回一个字典所有的键

[dict.update(dict2)](https://www.runoob.com/python/att-dictionary-update.html) 把字典dict2的键/值对更新到dict里

[pop(key[,default])](https://www.runoob.com/python/python-att-dictionary-pop.html) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。

1. 获取包的安装位置：

Import <packagename>

packagename.\_\_file\_\_ 就是包安装的路径

1. D
2. 设置主文件路径：

python所有的相对路径都是相对于主文件路径的，如果你要直接调用某些工程，其中写了大量的相对路径，那就用os.chdir 设置一个新的主路径，但是记得要重新设置回来。oldmainpath=os.getcwd()

13.类型转换：比如float 转 int int是一个类型转换函数

A=int(a)

14.list 删除： del self.pz[:-10]

15，python 类的成员变量在 init的时候必须使用self赋值初始化，否则定义的变量将作为所有对象的共有变量，你对其中一个赋值，就对所有的都赋值了。

16，文件的读写：打开fo=open(‘test.txt’,”w”);写入fo.write(str);fo.close();

## Numpy教程：

1. np array的拼接：

stocksinfo=np.append(price,shizhi,axis=2) 这个函数是吧price和shizhi拼接到一起，axis可以指定沿着哪一个轴拼接。

1. np 保存文件与加载文件:

保存：np.save('stockssamples/stocksinfo.npy',stocksinfo)

加载：a=np.load("/stockssamples/stockstocksinfo.npy")

1. 3，np array的创建：

# from numpy.random import Generator, PCG64

# rg = Generator(PCG64(12345))

#初始化一个np.array

# a=np.arange(100).reshape(10,10) #从0到99 构成10x10的矩阵

# print(a)

# a=np.zeros((10,10)) #初始化10x10的0矩阵

# print(a)

# a=np.ones((10,10)) #初始化10x10的全是1的矩阵

# print(a)

# a=np.full((10,10),3.14) #用3.14去填充10x10的矩阵

# print(a)

# a = rg.random((10,10)) #rg是一个随机数生成器，rg由Generator(种子)生成 种子一般使用PCG64 Generator和PCG64 从numpy.radom import

# print(a)

# a=np.random.normal(0,1,(10,10))#创建一个10x10的矩阵，里面的数据符合均值为0方差为1的分布

# print(a)

# a=np.random.randint(0,10,(10,10)) #创建一个10x10的矩阵，元素都是由0到10之间的int型

# print(a)

# a=np.eye(10) #10x10 的单位矩阵

# print(a)

# a=np.empty((10,10)) #创建10x10空矩阵，矩阵中的内容取决于内存中当前锁存的内容

# print(a)

# a=np.array([0,1,2,3,4,5,6,7,8,9])#array是一个函数，其参数是一个序列，序列使用[]表示的，不能使用多个数字作为参数

# print(a)

# a = np.array([(1.5,2,3), (4,5,6)]) #一个()表示一行,两行用','隔开

# print(a,a.shape)

# a=np.linspace(0,2,9) #初始化一个序列，从0开始到2中间等分9个算子

# print(a)

Np 的基本操作：

#np的基础操作

a = np.array( [20,30,40,50] )

b = np.arange( 4 )

print(a,b)

#减法

c=a-b

print(c)

#平方

print(b\*\*2)

#三角函数

print(10\*np.sin(a))

#判断得到的是bool类型的序列

print(a<35)

A = np.array( [[1,1],[0,1]] )#2x2矩阵A

B = np.array( [[2,0],[3,4]] )#2x2矩阵B

print(A\*B) #矩阵点乘，就是矩阵对应位置元素相乘得到的结果放到一个新矩阵的相同位置

print(A@B) #矩阵叉乘

print(A.dot(B)) #另一种矩阵叉乘

A\*=2 #矩阵自乘

print(A)

类型转换：a=np.arange(10) #这样得到的是一个int的，要改成float的要转换类型

a=a.astype(float)

1. 3，向量的比较：

I=(a==b) 这样得到一个尺寸与ab相同的bool向量。

上面的方法在判断整形的时候比较好，对于浮点型这种判断经常因为某些精度的原因出错，比如两个数因为不同平台的精度稍有差别或者保存读取的时候有点损失，最终数量上查了10的负10几次方，这个差别可以忽略不计，所以比较两个浮点型向量的时候不如作差绝对值的加和。dif=np.abs(szbig\_a-sz\_1s\_1a).sum()

另外如何判断向量中元素是否都为nan或者null?

1. 4, 向量的赋值

正常情况a是一个np array 如果想要用b给a赋值:a[:]=b[:] 或 a[0:5]=b[1:6] a[:]=0 a[:]=np.nan 都可以。但是特别注意,当a中的数据类型不固定，或者里面有str类型的东西，这种赋值方式将会失效。

1. 5，nparray中最大值的位置：

pos = np.unravel\_index(np.argmax(da),da.shape)

print(da[pos[0],pos[1]])

1. 6，绘制直方图：

一、直方图绘制参数详解

1、plt.hist(x, bins=10, range=None, normed=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False)

* x：指定要绘制直方图的数据。
* bins：指定直方图条形的个数。
* range：指定直方图数据的上下界，默认包含绘图数据的最大值和最小值。
* normed：是否将直方图的频数转换成频率。
* weights：该参数可为每一个数据点设置权重。
* cumulative：是否需要计算累计频数或频率。
* bottom：可以为直方图的每个条形添加基准线，默认为0。
* histtype：指定直方图的类型，默认为bar，除此之外，还有barstacked、step和stepfilled
* align：设置条形边界值的对齐方式，默认为mid，另外还有left和right
* orientation：设置直方图的摆放方向，默认为垂直方向
* rwidth：设置直方图条形的宽度
* log：是否需要对绘图数据进行log变换
* color：设置直方图的填充色
* edgecolor：设置直方图边框色
* label：设置直方图的标签，可通过legend展示其图例
* stacked：当有多个数据时，是否需要将直方图呈堆叠摆放，默认水平摆放

1. 7，np array的顺序重排说着说着洗牌

random.shuffle(a) 不返回

1. 向量的逻辑运算操作：

在python中and or这种逻辑运算只能用于一个数，在对np array使用的时候就不行了，必须使用np的函数进行bitwise\_and、bitwise\_or、bitwise\_not，可以实现这个功能，对两个bool类型的数据进行逻辑运算操作。

1. 搜索一个特定向量：

def SearchOneArrayInDf(df,array):

if df.values.shape[1]!=len(array):

return

a,b = np.where(df.values==array)

b = ''.join(str(i) for i in b) #数组转字符串

t = np.arange(0,len(array))

t = ''.join(str(i) for i in t)

return a[b.find(t)] #在一个大字符串中找到小字符串

return a

特别注意where获得的是分别对比每一项的结果，并不是获得了完全相等的index，后面需要自己处理。

1. 二维矩阵的reisze

Newarray = np.resize(oldarray,newshape)

1. 二维矩阵的描画：

import matplotlib.pyplot as plt

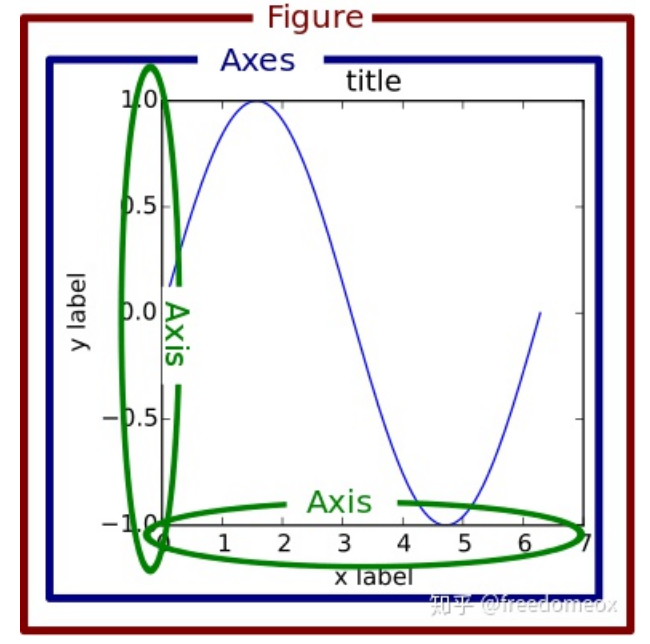
plt.figure("Image") *# 图像窗口名称*

plt.imshow(img)

1. 的

## Matplot教程：

参考：<https://zhuanlan.zhihu.com/p/93423829>



1. 创建画板：figure
   1. 创建：fig=plt.figure(num=,figsize=(w,h单位英寸),facecolor=(0.5,0.5,0.5))
   2. 这是标题：fig. suptitle (“name”,fontsize=12)
2. 创建子图：ax

2.1 初始化ax1=fig. add\_subplot(2,2,1) 表示生成2x2个子图，这是其中第一个

2.2 设置标题：ax1.set\_title(‘title’)

2.3 给坐标轴命名: ax1.set\_xlabel(name,fontsize,fontfamily,fontstyle)

1. D

## Pytorch教程：

### **torch**.**utils**.**data**.**DataLoader**(dataset,batch\_size=64,shuffle=False(洗牌),num\_workers=0(使用几个进程))

dataset:数据集 用TensorDataset(x\_data,y\_data)

batch\_size:是一次用训练用几个样本，注意当迭代取数据的时候一次也是去除dataset个数据的整体，并不是分开的。

shuffle:是否重新洗牌

问题：如果输入数据是多个，如何同时使用多个dataloader？

使用 c=zip(a,b) 可以把a,b合并到一起，for data in c: 得到的data是个tuple（元组: 速度快，安全，不可变<https://zhuanlan.zhihu.com/p/92527888>）而如果对a,b迭代只会得到list类型这点是不同的。

特别注意：当使用zip把多个dataloader合并到一起的时候，这个dataloader在遍历一次之后将不能够再次迭代。就是一般训练使用的epoch概念，训练一轮后dataloader中的样本都使用过了，当再次进入到新一轮迭代的时候dataloader就去不到数据了。解决办法就是每次迭代的时候都zip一次。

### TensorDataset(data\_tensor=x,target\_tensor=y)

2.1 x,y必须都是tensor也就是torch的数据类型，如果想使用np array作为输入的话必须是用torch.from\_numpy进行转换。

2.2 DataSet是一个特殊的数据类型，不能直接打印出结果，那么如何测试呢？

构造dataloader ->迭代dataloser->在迭代中打印data和lebel

迭代：for batch\_idx, (data, target) in enumerate(train\_loader):

2.3:x和y在第一个维度上一定要一样。

### 如果处理多输入网络？

3.1如何创建网络

3.2如何构造dataloader？

用zip可以将连个dataloader合并

3.3如何读取数据？

Zip过的dataloader中获得的是一个tuple类型，需要用index去获得data的tenor和target的tensor 然后才能向device中推。

for i,data in enumerate(traindataloader):

d1,l1=data[0][0].to(device),data[0][1].to(device)

d2,l2=data[1][0].to(device),data[1][1].to(device)

print(d1,l1,type(d1),type(l1))

### 如何查看tensor的shape：tensor\_a.size()

### 如何初始化一个tensor:

5.1 使用直接数据初始化tensor

tensor=torch.Tensor([2,3])

5.2 初始化指定形状的tensor

tensor=torch.Tensor(2,3)这样初始化的tensor中的数据是随机的。

Shape=(2,3) zeros=torch.zeros(size=Shape) ones=torch.ones(size=Shape)

Size=[2,3] randn=torch.randn(size)

### 如何初始化网络中的层？

6.1 卷积层conv：Conv2d(inc,outc,ksize,stride=1,padding=0,padding\_mode,dilation=1,groutps,blas)

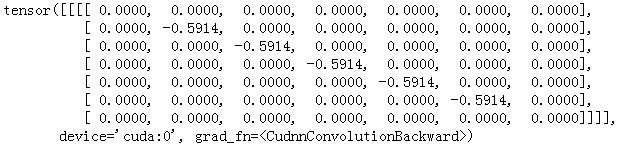
Inc 输入通道 outc 输出通道 stride 间隔 ksize 是卷积核的大小

Padding 0是输出的WH缩小 1-n是先将原数据的WH扩展padding行列之后再卷积

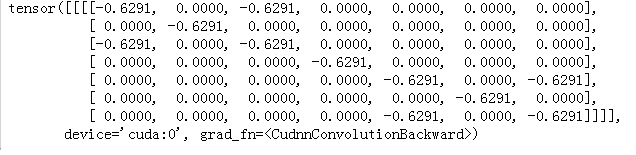
Padding\_mode

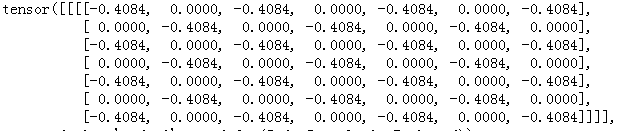
|  |  |
| --- | --- |
| Zeros | 用0填充扩展的行列 |
| reflect | 与原有边缘对称的数据填充 |
| Replicate 或者 circular | 将原有边缘处的数据复制过去或者是最近一行列的循环。 |

Padding\_mode =zeros 5x5单位矩阵

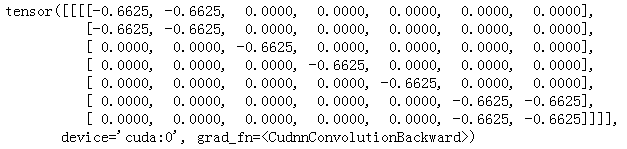


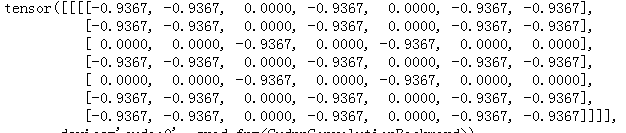
Padding\_mode = reflect 隔着取



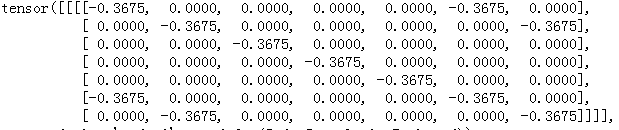


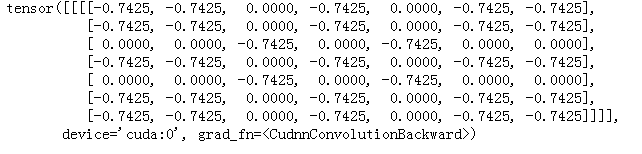
Padding\_mode = replicate





Padding\_mode =circular





Dilation 卷积核之间的空间，比如空洞卷积时要大于1



Groups 1就是正常的情况 input通过一个卷积输出output

2 指的是两个串联的卷积核，每一个只操作一半的channels同时也输出一半的channels

Inc 如果groups的参数与输入的通道数相同，那么就意味着每个通道都适用自己的卷积核其输出的通道数平均分为(outc/intc)

Blas 是给output加一个可学习的偏移

注意：像kernel\_size,stride,padding,dilation这种偏移或者说步长类的东西是可以在两个轴上设置不同的数的，如果参数输入为一个数字那么就默认是宽高相同，如果要在两个方向上设置不同的数值就要输入一个tuple(y,x)这样就有了非矩形的卷积核或者步长。

### np中的array如何与torch中的tensor互相转换

7.1 tensor转array a=np.array(tensor)

这里需要特别注意，tensor是一个需要求导的变量的时候是不可以用这个方法转的，要先用tensor.detach()把tensor变成不需求导的才可以。

7.2 array转tensor torch.from\_numpy(a)

### 关于gpu的使用

import torch

torch.cuda.set\_device(id)

如何查看gpu使用情况？在linux中使用nvidia-smi查看gpu使用情况 top查看cpu的使用情况。

可以使用os.environ['CUDA\_VISIBLE\_DEVICES'] = '0,3'来设置可见的gpu

如何使用多gpu：model=nn.DataParallel(model) model=model.cuda()

Code:model = Model(input\_size, output\_size)

if torch.cuda.device\_count() > 1:

print("Let's use", torch.cuda.device\_count(), "GPUs!")

# dim = 0 [30, xxx] -> [10, ...], [10, ...], [10, ...] on 3 GPUs

model = nn.DataParallel(model)

model.to(device)

注意：在使用多gpu的时候具体使用几个gpu取决于batchsize，而与dataloader关系不大。也就是说训练很少量的样本设置好batchsize也可以使用多个gpu。

如何释放显存？torch.cuda.empty\_cache()

关于显存释放的问题：torch.cuda.empty\_cache()可以释放显存，但是前提是必须是不再使用的显存。如果不加with torch.no\_grad():这个的的话就必须backward求导后整个grad才会被认为是不用的内存，才会被回收。也就是说每一轮训练(计算loss并求导)后都是需要进行这个操作的（当计算梯度时，在loss.backward之后才会释放显存）。

### device的初始化：

torch.device(‘cuda:0’)指定使用0号gpu

torch.device(‘cuda’) 使用当前gpu

torch.device(‘cpu’)使用cpu

1. 损失函数：<https://blog.csdn.net/rosefun96/article/details/88058708>

首先说最重要的，每个损失函数都有两个参数：reduction 和 size\_average。

reduction:是否降维，降维就返回标量、不降维就返回矢量。默认是True 此时size\_average无效。

size\_average:是否计算平均误差，如果返回值是标量的话，当size\_average为True时返回数据的平均误差，否则返回总误差。

注意要先声明一个函数变量然后再使用之loss\_fn = torch.nn.MSELoss()

具体损失函数介绍：

1,nn.L1loss：output(预测值) 与 GT(真值也就是标注值)的绝对值误差。

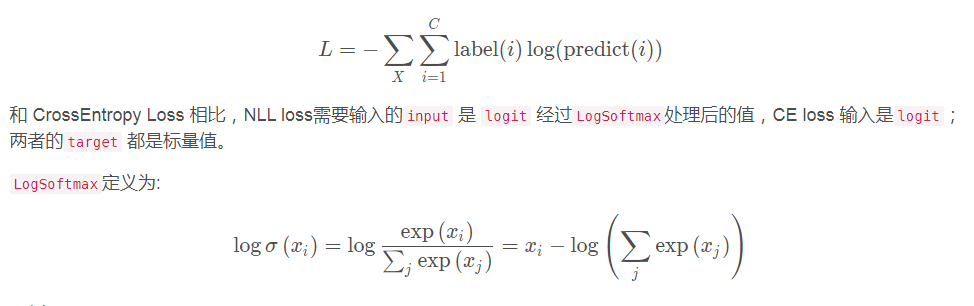
2,nn.SmoothL1loss：当output与GT绝对值差在1之内时是L2（平方损失）损失，其他时候是L1损失。

3，nn.MSELoss L2损失

4，nn.CrossEntropyLoss:交叉熵 主要刻画概率分布之间的距离。待续…

5，nn.BCELoss 二分类交叉熵 待续…

6，nn.NLLLoss 负对数释然损失函数 用于分类 待续…



7,nn.NLLLoss2d：nn.NLLLoss的多维度实现。

### pytorch tensor的拼接：

torch.cat((a,b,c),dimindex)。dimindex表示在第几个维度上拼接。

Torch.stack((a,b,c),dimindex)。Dimindex表示维度，stack与cat的区别是cat的dimindex必须是已经有的维度，而stack是创建一个新的维度并在这个维度上进行拼接。

### model的参数:

pars=model.parameters() 可以获得模型的参数，是一个generator结构体。

for parameters in model.parameters():

print(type(parameters),parameters.shape)

其中parameters的数量就是模型的总层数。

每个parameters有自己的shape就可以看到参数的具体数量了。

### reshape 与 view 与transpose三者都是改变数据形状的函数。

x = torch.randn(3,4)

x = x.view(4,3) 使用view的前提是数据必须是连续的，在内存上连续。

使用x.is\_contiguous()可以验证数据是否连续

x = x.reshape(4,3) 不需要数据是连续的

x = x.transpose((1,0,2))参数是一个tuple记录的是新的维度序号的顺序，比如上面的代码就是把rows与channels互换。Transpose与reshape\view 不同。Transpose是改变数据实际排列顺序的。

注意，reshape\view 都是不改变数据总量的 而resize是改变数据总量的。

### 随机数列：

整数型随机数列：random.randint(start,end)这里注意end是可以被取到的这与切片时候的左闭右开有很大区别。

### dropout的使用:作用是方式过拟合

测试dropout的作用

### torch的tensor 与 np的array 相互转换

a是一个tensor 通过调用numpy函数转成array换b=a.numpy()。

a 是一个array 通过torch.form\_numpy()转换成torch

b=torch.from\_numpy(a)。

### Python回传参数给C：不支持list嵌套 除非里面有nparray

16.1：回传tuple：

**int ok;**

**int i, j;**

**long k, l;**

**char \*s;**

**int size;**

**ok = PyArg\_ParseTuple(args, "");/\* No arguments \*/**

**/\* Python call: f() \*/**

**ok = PyArg\_ParseTuple(args, "s",&s); /\* A string \*/**

**/\* Possible Python call: f('whoops!')\*/**

**ok = PyArg\_ParseTuple(args,"lls", &k, &l, &s); /\* Two longs and a string \*/**

**/\* Possible Python call: f(1, 2,'three') \*/**

**ok = PyArg\_ParseTuple(args,"ii", &i, &j);**

**ok = PyArg\_ParseTuple(args,"(ii)s#", &i, &j, &s, &size);**

**/\* A pair of ints and a string, whosesize is also returned \*/**

**/\* Possible Python call: f((1, 2),'three') \*/**

**{**

**char \*file;**

**char \*mode = "r";**

**int bufsize = 0;**

**ok = PyArg\_ParseTuple(args,"s|si", &file, &mode, &bufsize);**

**/\* A string, and optionally anotherstring and an integer \*/**

**/\* Possible Python calls:**

**f('spam')**

**f('spam', 'w')**

**f('spam', 'wb', 100000) \*/**

**}**

**{**

**int left, top, right, bottom, h, v;**

**ok = PyArg\_ParseTuple(args,"((ii)(ii))(ii)",**

**&left, &top,&right, &bottom, &h, &v);**

**/\* A rectangle and a point \*/**

**/\* Possible Python call:**

**f(((0, 0), (400, 300)), (10, 10)) \*/**

**}**

**{**

**Py\_complex c;**

**ok = PyArg\_ParseTuple(args,"D:myfunction", &c);**

**/\* a complex, also providing a functionname for errors \*/**

**/\*** Possible Python call:myfunction(1+2j) \*/

}

17，Pycocotools安装：

在ubuntu上 pip install pycocotools就可以了，但是在windows上需要参考这个网址

注意可能需要安装VS C++ build tools

若出现VS2015报“安装包丢失或损坏”问题，则原因为：microsoft root certificate authority 2010、microsoft root certificate authority 2011证书未安装，导致文件校验未通过，下载并安装这两个证书即可。

## CV2教程

#### 读取\保存图像：

读取：cv2.imread('timg.jpg')

保存：cv2.imwrite(path,img)

#### 图像基础处理：

#### 描画：

3.1 画线：cv2.line(img,ps,pe,color,thickness,lineType,shift)

3.2 矩形：cv2.rectangle(img,ps,pe,color,thickness,lineType,shift)

3.3 圆:cv2.circle(img,center,radius,color,thickness,lineType,shift)

3.4 多边形:cv2.polylines(img,pts,inClosed,color,thickness,lineType,shift)

3.5 文字:cv2.putText(img,text,org,fontFace,fontScale,color,thickness,lineType,bottomLeftOrigin) org:文字在图像中的左下角坐标；fontFace:字体类型，FONT\_HERSHEY\_SIMPLEX

## Pandas教程

1. 创建DataFrame:

创建DataFrame有很多种方法这里直说几种非常实用的。

* 1. 从csv等文件直接读取:df=pd.read\_csv(path)
  2. 用dict初始化DataFrame：
  3. 已知columns和data用list类型初始化（少数几行数据）：

Df=pd.DataFrame([[0,1,2,3]],columns=[c1,c2,c3,c4])注意标颜色的是实际数据的list，它外边还要再扩一个[]，这表示输入数据可以有很多行。这在只需要一行数据的时候很容易忽略。

* 1. 用np.array初始化 当我们有了一块大数据的时候用

Df=pd.DataFrame(data=np.arrage(100).reshape(10,10),columuns=cs,index=ins)

1. 操作DataFrame

2.1 截取DataFrame

2.1.1 截取一行:如果知道要截取行的需要可以用a=df.iloc[rowindex,:];如果知道键名可以用a=loc[rowname,:];这里必须注意这个键名必须是Df的索引一个df只有一个列是索引列其他的不是，如果不是索引可以用下面的代码编程索隐列:df.index=df.colname这样就把colname这个列设置为索引列了。

注意：有的时候直接用属性[index]是无法索引到的，必须用iloc

2.2 删除行列:

price=price.drop(names,axis=1)

names 就是行或者列的键值的序列，axis是维度的序号。

* 1. 插入列：insert(index,’colname’,data)

Eg: frame.insert(0, 'num', np.ones(5))

* 1. 插入行：

Dftmp = pd,DataFrame()

Df.append(Dftmp)

按条件筛选和多条件筛选：df=df[(df.A>1)&(df.A<41)]必须用括号括起来，只能用&不能用and。

* 1. 保存成csv。Dfname.to\_cst(“name.csv”)
  2. Series是按照index的数值来索引的而不是一定从0 开始

比如你从df中截取了一个切片 df1 = df[df.colname = ‘x’] 这时候df1.colname这个serices就不是从0开始计数的了。比如你是从index=1-10 那么df1.colname[0]就是不存在的，更笨没有这个数据。如果要想访问第一个那就㤇用df1.colname.values[0]这样就是第一个数据了，因为values是np.array类型的。

* 1. 数据逆序：

伪逆序：只是改变了index的顺序，实际里面的数据没有改变，对有些函数不好使，比如plot。A=A[::-1]

真逆序：

**for s in ETHmin.columns:**

**ETHmin[s].values[:] =ETHmin[s].values[::-1]**

#将dataframe中的每一个属性的数据（np.array）进行逆序，这样整个数据就逆序了

ETHmin.open.values[:] = ETHmin.open.values[::-1]

ETHmin.close.values[:] = ETHmin.close.values[::-1]

ETHmin.id.values[:] = ETHmin.id.values[::-1]

ETHmin.low.values[:] = ETHmin.low.values[::-1]

ETHmin.high.values[:] = ETHmin.high.values[::-1]

ETHmin.amount.values[:] = ETHmin.amount.values[::-1]

或者：ETHmin[open]. values[:]

* 1. 提取列：DataFrameNmae.colname
  2. dataframe转numpy：DataFrameName.to\_numpy()

## 训练经验

1. 输入数据应该归一化,如果不做归一化处理会导致梯度计算错误nan

是因为有某一个维度的数值非常巨大，导致的。最科学应该还是分别归一化。

1. 输出labels如果不归一化可能会导致梯度消失

2.1 要验证是否与labels中含有负数有关

|  |  |  |  |
| --- | --- | --- | --- |
|  | Loss | 梯度消失迭代书 |  |
| 原始labels | 9.790630 | 47 | 9.790630 |
| 归一化N | 0.317315 | 57 | 9.72614 |
| 标准化S | 16.773136 | 45 | 9.8151 |
| N+S | 16.773136 | 45 | 8.7868 |
| S+N | 0.317315 | 57 |  |
| sigmoid | 0.317058 | 57 |  |
| Data分别归一化 | 0.317058 | 57 |  |
| 2层 | 0.319790 |  |  |
| 3层 | 0.317622 |  |  |
| 4层 | 0.317079 |  |  |
| 没有relu leaky | 0.316613 |  |  |
| 多维度数据 |  |  |  |

1. 每次迭代前必须梯度清零
2. 线性不要与relu一起用，会有0梯度

## PyGears

Gears是齿轮的意思，这个工具是使用python设计硬件逻辑的工具包。硬件逻辑可以以几何图形的形式输出。是一种免费的开源硬件描述语言 (HDL)，可以编译为Verilog语言。

下载github安装：

PyGears有一下依赖：

1. PythonOCC：使用python上画图的，是用来生成电路图的
2. wxPython GUI图形库
3. matplotlib 画线统计的

其最大的好处是，可以像函数复用一样复用电路。但是基本的符合芯片计算逻辑的函数设计是必不可省的。还有就是可以转换成verilog语言。