

代码内容审核建议 9 (2.5.1, 2.5.2)




- 2.5.1 中类似的代码，改进的函数，
可以着重标记一下改动的不同，可以更加方便的让读者找出不同

• 图像显示

为方便查看图像，或者排布图像用于研究文章的图表说明，定义图像排布显示的函数。图像处理过程，例如打开、调整图像大小均使用PIL图像处理库，在调整大小时要保持图像的R、G、B三个通道不变。

```
def imgs_layoutShow(imgs_root,imgsFn_lst,columns,scale,figsize=(15,10)):  
    ...import math,os  
    ...import matplotlib.pyplot as plt  
    ...from PIL import Image  
    ...  
    ...function - 显示一个文件夹下所有图片，便于查看。  
    ...  
    ...Paras:  
    ...    imgs_root - 图像所在根目录  
    ...    imgsFn_lst - 图像名列表  
    ...    columns - 列数  
    ...  
    ...rows=math.ceil(len(imgsFn_lst)/columns)  
    ...fig,axes=plt.subplots(rows,columns,sharex=True,sharey=True,figsize=figsize) ... #布局多个子图，每个子图  
    ...ax=axes.flatten() ... #降至1维，便于循环操作子图  
    ...for i in range(len(imgsFn_lst)):  
    ...    img_path=os.path.join(imgs_root,imgsFn_lst[i]) #获取图像的路径  
    ...    img_array=Image.open(img_path) #读取图像为数组，值为RGB格式0-255 ...  
    ...    img_resize=img_array.resize([int(scale * s) for s in img_array.size]) #传入图像的数组，调整图片大小  
    ...    ax[i].imshow(img_resize) #显示图像  
    ...    ax[i].set_title(i+1)  
    ...fig.tight_layout() #自动调整子图参数，使之填充整个图像区域 ...  
    ...fig.suptitle("images show",fontsize=14,fontweight='bold',y=1.02)  
    ...plt.show()  
  
    imgs_fn=util_misc.filePath_extraction('./data/default_20170720081441',"jpg")  
    imgs_root=list(imgs_fn.keys())[0]  
    imgsFn_lst=imgs_fn[imgs_root]  
    columns=6  
    scale=0.2  
    imgs_layoutShow(imgs_root,imgsFn_lst,columns,scale)
```

且，正如黄色框标出的那样，参数列表修改，函数功能注释也需要跟随修改

上述定义的图像显示函数，存在空白的子图。以及图像文件路径处理的参数包括了两个，一个是根目录；一个是文件名列表。下述更新的函数则解决了上述两个问题。

```
def imgs_layoutShow_FPList(imgs_fp_list,columns,scale,figsize=(15,10)):  
    import math,os  
    import matplotlib.pyplot as plt  
    from PIL import Image  
    ...  
    function - 显示一个文件夹下所有图片，便于查看。  
    ...  
    Paras:  
    imgs_root - 图像所在根目录  
    imgsFn_lst - 图像名列表  
    columns - 列数  
    ...  
    rows=math.ceil(len(imgs_fp_list)/columns)  
    fig,axes=plt.subplots(rows,columns,figsize=figsize,) #布局多个子图，每个子图显示一幅图像  
    ax=axes.flatten() #降至1维，便于循环操作子图  
    for i in range(len(imgs_fp_list)):  
        img_path=imgs_fp_list[i] #获取图像的路径  
        img_array=Image.open(img_path) #读取图像为数组，值为RGB格式0-255  
        img_resize=img_array.resize([int(scale * s) for s in img_array.size]) #传入图像的数组，调整图片大小  
        ax[i].imshow(img_resize,) #显示图像 aspect='auto'  
        ax[i].set_title(i+1)  
    invisible_num=rows*columns-len(imgs_fp_list)  
    if invisible_num>0:  
        for i in range(invisible_num):  
            ax.flat[-(i+1)].set_visible(False)  
    fig.tight_layout() #自动调整子图参数，使之填充整个图像区域  
    fig.suptitle("images show",fontsize=14,fontweight='bold',y=1.02)  
    plt.show()  
    import glob  
    imgs_dougong_fps=glob.glob('./data/default_20170720081441/*.jpg')  
    columns=6;scale=0.2  
    imgs_layoutShow_FPList(imgs_dougong_fps,columns,scale)
```

Pyth

2. 参数和注释信息不对应

```
def img_exif_info(img_fn, printing=True):
    from PIL import Image
    from PIL.ExifTags import TAGS
    from datetime import datetime
    import time

    function - 提取数码照片的属性信息和拍摄数据, 即可交换图像文件格式(Exchangeable image file format, Exif)

    Paras:
        img_fn - 一个图像的文件路径
```

3. 键值重复, 需要删除上面一行

在Exif信息提取函数中, 根据'DateTimeOriginal:2017:07:20 09:16:58'时间信息, 计算时间戳, 用于图像按拍摄时间排序, 可以绘制图片拍摄的行走路径。此次绘制使用Plotly库提供的go方法实现。该方法在调用地图时, 不需提供mapbox地图数据的访问许可(access token)。

```
imgs_ChicagoDowntown_coordi=[]
for fn in imgs_ChicagoDowntown_fps:
    img_exif_2, geo_coordinate = img_exif_info(fn, printing=False)
    imgs_ChicagoDowntown_coordi.append((geo_coordinate['GPSLatitude'], -geo_coordinate['GPSLongitude'], geo_coordinate['GPSAltitude'], img

import pandas as pd
pd.set_option('display.float_format', lambda x: '%.5f' % x)
import plotly.graph_objects as go
imgs_ChicagoDowntown_coordi_df = pd.DataFrame(data=imgs_ChicagoDowntown_coordi, columns=['lat', 'lon', 'elevation', 'timestamp']).sort_val

fig = go.Figure(go.Scattermapbox(mode = "markers+lines", lat=imgs_ChicagoDowntown_coordi_df.lat, lon=imgs_ChicagoDowntown_coordi_df.lon
fig.update_layout(
    margin = {'l':0, 't':0, 'b':0, 'r':0},
    mapbox = {
        'center': {'lon': 10, 'lat': 10},
        'style': "stamen-terrain",
        'center': {'lon': -87.62401, 'lat': 41.88205},
        'zoom': 14})
#fig.add_trace()

fig.show()
```

4. 此部分需要添加详细描述

3) RGB色彩的三维图示

包含色彩信息(RGB)的数据投射到三维空间中, 可以通过判断区域色彩在三维空间域中的分布情况来把握Red、Green和Blue色彩分量的变化情况。

```
def imgs_colorSpace(imgs_root, imgsFn_lst, columns, scale, figsize=(15,10)):
    import math, os
    import matplotlib.pyplot as plt
    from PIL import Image
    import numpy as np
    from tqdm import tqdm
    ...

    function - 将像素RGB颜色投射到色彩空间中, 直观感受图像颜色的分布。

    Paras:
        imgs_root - 图像所在根目录
        imgsFn_lst - 图像名列表
        columns - 列数
    ...
```

代码和图像, 结果描述较少

2.5.1.2 聚类(Clustering)

监督学习(Supervised learning), 是机器学习的一种方法, 可以学习训练数据集建立学习模型用于预测新的实例。回归属于监督学习, 所用到的数据集被标识分类, 通常包括特征值(自变量)和目标值(标签, 因变量)。非监督学习(Unsupervised learning), 则没有给定事先标记的数据集, 自动对数据集进行分类或分群。聚类则属于非监督学习。

聚类是把相似的对象通过静态分类的方法分成不同的组别或者更多的子集(subset), 这样让在同一个子集中的成员对象都有相似的一些属性。聚类涉及的算法很多, K-Means是常见算法的一种, 通过尝试将样本分离到 n 个方差相等的组中对数据进行聚类, 最小化指标(criterion), 例如聚类内平方和指标(within-cluster sum-of-squares criterion), 其公式可表达为: $\sum_{i=0}^n \min(\|x_i - \mu_j\|^2) : \mu_j \in C$ 。Python Machine Learning 对K-Means算法解释的非常清晰, 引用其中的案例加以说明。

图中假设了两个簇 C_0 和 C_1 , 即 $k = 2$ 。首先随机放置两个质心(centroid), 并标识为old_centroid。通过逐个计算每个点分别到两个质心的距离, 比较大小, 将点归属于距离最近的质心(代表簇或组分类), 例如对于点 a , 其到质心 C_0 距离 d_0 小于到质心 C_1 的距离, 因此点 a 归属于质心 d_0 所代表的簇。将所有的点根据距离远近归属于不同的质心所代表的类之后, 由所归属簇中点的均值作为新的质心, 图中用new_centroid假设标识。分别计算旧质心和新质心的距离, 如果所有簇质心的新旧距离值为0, 则意味质心没有发生变化, 即完成聚类; 否则, 用新的质心重复上一轮的计算, 直至质心新旧距离值为0为止。只要有足够的时间, K-Means总是收敛的, 但它可能是一个局部最小值。这高度依赖于质心的初始化。因此, 计算通常要进行多次, 并对质心进行不同的初始化。

K 均值聚类算法 百度百科 [https://baike.baidu.com/item/K 均值聚类算法](https://baike.baidu.com/item/K_均值聚类算法) (google.com)

- 聚类算法比较

Sklearn官网聚类部分提供了一组代码，比较多个不同聚类算法，其归结如下：

这部分的代码和结果全部采用英文，是否应该翻译为中文

7. 2.5.2 包含如下内容

2.5.2 图像分类器、识别器与机器学习模型

2.5.2.1 [Flask] 构建实验用网络应用平台

2.5.2.2 问卷调查

2.5.2.3 视觉词袋与构建图像映射特征

2.5.2.4 决策树 (Decision trees) 与随机森林 (Random forests)

2.5.2.5 图像分类_识别器

2.5.2.1-3 和本章主题有点串联困难，建议在 2.5.2.1 前介绍各部分目的内容进行章节内容关系串联介绍

2.5.2.4 部分中，决策树中对信息论有关的公式内容介绍，可以单独开辟一小节或者添加引用链接讲解关键公式，将重要理论部分单独拿出，可以更加方便理解

信息论中的基本概念 - ZingpLiu - 博客园

<https://www.cnblogs.com/zingp/p/10265983.html>

信息论基础与决策树基础_Star 星屹程序设计的博客-CSDN 博客

https://blog.csdn.net/weixin_42067873/article/details/110207354

决策树基础(信息论)——AI 之机器学习实战 (Peter Harrington) - 知乎

<https://zhuanlan.zhihu.com/p/143552898>

讨论和建议

对于长篇幅的引用以及注释的使用，应该保持语言一致性，即，尽可能的采用一种语言，使目标读者容易接受

且对于长代码的运行结果，可以稍加评价和讲解，这样可以丰富内容，使代码和结果平易近人，易于理解

对于图像，横纵坐标，图像中点或者其他标记的大小和颜色，应该在图上或者后面的解释中说明，这样可以提高图像的易读性

对于计算样例，可以采用（单行）1，（单行多列）2,3，（多行多列）4,6,8 副图像来做显示，太多的举例会造成运行时间的加长和视觉疲劳，可以在代码循环中注释如何修改，来调整图片的数量