BOW

2018年2月17日 10:46

```
import numpy as np
import pandas as pd
import sqlite3
import pickle
import cv2
from sklearn.cluster import KMeans

if __name__ == "__main__":
    resizeImgData=getImgPath()
    training_data=load_training_data(resizeImgData)
    scaling_size=100
    kmeans,centroids=FeatureBuilder().get_codewords(training_data,scaling_size)

with open('featK.pkl','wb') as f1: # 使用with结构避免手动的文件关闭操作
      pickle.dump((kmeans,centroids),f1) #存储kmeans模型，以及聚类中心点

feature_map=get_feature_map(training_data,kmeans,centroids,scaling_size)
 with open('feature_map.pkl','wb') as f2:
     pickle.dump(feature_map,f2) #存储特征
```

01 02 03 08

```
'''提取网页图像打分存储到服务器SQLite数据库中的数据'''
def getImgPath():
    connPred = sqlite3.connect('local_data.db')  # 连接数据库
    sql = "select * from imageseval"
    data = pd.read_sql(sql=sql, con=connPred)  #根据pandas读取SQLite数据库参数要求配置参数
    print(data)
    return data
```

从数据库中读取数据为dataframe数据格式

	id	imagename	good	medium	poor	eval	eval_time	image_id
1	1	static/images/imresize/img_0321.jpg	4	0	0	好	2018-01-28 17:43:47.808424	null
2	2	static/images/imresize/img_0323.jpg	0	0	1	差	2018-01-28 17:43:55.915705	null
3	3	static/images/imresize/img_0326.jpg	0	2	0	中	2018-01-28 17:44:05.366660	null
4	4	static/images/imresize/img_0327.jpg	1	0	0	好	2018-01-28 17:44:14.690694	null
5	5	static/images/imresize/img_0331.jpg	1	0	0	好	2018-01-28 17:44:21.485299	null
6	6	static/images/imresize/img_0332.jpg	0	0	1	差	2018-01-28 17:44:28.395802	null
7	7	static/images/imresize/img_0333.jpg	0	1	0	中	2018-01-28 17:44:35.023523	null
8	8	static/images/imresize/img_0334.jpg	2	0	0	好	2018-01-28 17:44:41.900496	null
9	9	static/images/imresize/img_0336.jpg	1	0	0	好	2018-01-28 17:44:47.083740	null
10	10	static/images/imresize/img_0339.jpg	2	0	0	好	2018-01-28 17:44:54.997002	null

	id	imagename	good	medium	poor	eval	eval_time	image_id
0	1	static/images/imresize/img_0321.jpg	4	0	0	好	2018-01-28 17:43:47.808424	None
1	2	static/images/imresize/img_0323.jpg	0	0	1	差	2018-01-28 17:43:55.915705	None
2	3	static/images/imresize/img_0326.jpg	0	2	0	中	2018-01-28 17:44:05.366660	None
3	4	static/images/imresize/img_0327.jpg	1	0	0	好	2018-01-28 17:44:14.690694	None
4	5	static/images/imresize/img_0331.jpg	1	0	0	好	2018-01-28 17:44:21.485299	None

```
'''根据图像打分评价标志'好'，'中'和'差'分离数据，提取对应的图像路径'''
def load_training_data(imgPD):
    training_data=[]
    # print(input_folder)
    goodPD=imgPD[imgPD['eval']=='好'] #先判断获取bool值mask后提取数据
    mediumPD = imgPD[imgPD['eval'] == '中']
    poorPD = imgPD[imgPD['eval'] == '差']
    training_data.append([{'object_class':'good','image_path':i} for i in goodPD['imagename'].tolist()]) #根据评价类标提取对应的图像路径名
    training_data.append([{'object_class': 'moderate', 'image_path': i} for i in mediumPD['imagename'].tolist()])
    training_data.append([{'object_class': 'poor', 'image_path': i} for i in poorPD['imagename'].tolist()])
    training_dataFlat=flatten_lam(training_data)
    print(training_dataFlat[:5])
    return training_dataFlat
```

根据'eval '评价结果，提取对应的'imagename'，为图像路径。

```
[{'object_class': 'good', 'image_path': 'static/images/imresize/img_0321.jpg'}, {'object_class': 'good', 'image_path': 'static/images/imresize/img_0327.jpg'}, {'object_class': 'good', 'image_path': 'static/images/imresize/img_0331.jpg'}, {'object_class': 'good', 'image_path': 'static/images/imresize/img_0334.jpg'}, {'object_class': 'good', 'image_path': 'static/images/imresize/img_0336.jpg'}]
```

```
'''图像特征提取'''
class FeatureBuilder(object):
    '''计算图像特征，返回关键点及特征向量'''
    def extract_features(self,img):
        keypoints=StarFeatureDetector().detect(img)
        print(len(keypoints),keypoints[:5])
        keypoints,feature_vectors=compute_sift_features(img,keypoints)
        print(len(keypoints),keypoints[:5])
        print(len(feature_vectors),[len(i)foriinfeature_vectors[:5]],feature_vectors[0])
        return feature_vectors

    '''提取指定采样图像数量的图像特征，此时图像评价指标有3类'good','moderate'和'poor'。'''
    def get_codewords(self,input_map,scaling_size,max_samples=200): #max_samples为采样数量，例如键为'good','moderate','poor'的分别各自取12个图像，当程序调试正确后，调大该值
        keypoints_all=[]
        count=0
        cur_class=''
        for item in input_map:
            if count>=max_samples:
                if cur_class != item['object_class']:
                    count=0
                else:
                    continue
            count+=1
            if count==max_samples:
                print("Build centroid for",item['object_class'])
            cur_class=item['object_class']
            img=cv2.imread(item['image_path'])
            num_dims=128
            feature_vectors=self.extract_features(img)
            keypoints_all.extend(feature_vectors) #提取每一图像特征，并追加在列表中
        print(len(keypoints_all),keypoints_all[0])
        kmeans,centroids=BagOfWords().cluster(keypoints_all)  #聚类特征
        print(kmeans,centroids.shape,centroids)
        return kmeans,centroids
```

05 获取Star特征检测器提取的关键点

06

```
'''定义类，用于处理Star特征检测相关函数'''
class StarFeatureDetector(object):
    def __init__(self):
        self.detector = cv2.xfeatures2d.StarDetector_create()
    def detect(self,img):
        return self.detector.detect(img)  #对输入图像运行检测器
```

```
194 [<KeyPoint 11D2E610>, <KeyPoint 11D2E750>, <KeyPoint 11D2E5C0>, <KeyPoint 11D2E110>, <KeyPoint 11D2E598>]
294 [<KeyPoint 11D2E138>, <KeyPoint 11D3F318>, <KeyPoint 11D3F340>, <KeyPoint 11D3F368>, <KeyPoint 11D3F390>]
831 [<KeyPoint 11D3F2F0>, <KeyPoint 11E021D8>, <KeyPoint 11E02200>, <KeyPoint 11E02228>, <KeyPoint 11E02250>]
48 [<KeyPoint 11E021B0>, <KeyPoint 11E10598>, <KeyPoint 11E105C0>, <KeyPoint 11E105E8>, <KeyPoint 11E10610>]
367 [<KeyPoint 11E10570>, <KeyPoint 11E10A70>, <KeyPoint 11E10A98>, <KeyPoint 11E10AC0>, <KeyPoint 11E10AE8>]
```

```
'''提取SIFT特征'''
def compute_sift_features(img,keypoints):
    if img is None:
        raise TypeError('Invalid input image')
    img_gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #将图像转为灰度
    keypoints,descriptors=cv2.xfeatures2d.SIFT_create().compute(img_gray,keypoints) #SIFT特征提取器提取特征
    return keypoints,descriptors
```

```
194 [<KeyPoint 11D3E340>, <KeyPoint 11D3E368>, <KeyPoint 11D3E390>, <KeyPoint 11D3E3B8>, <KeyPoint 11D3E3E0>]
194 [128, 128, 128, 128, 128] [ 2.  1. 12.  9.  1.  0. 23. 106.  0.  0. 52. 28.  4.  6. 19. 13.  2.  1. 46. 117. 15.  6.  4.
 5.  6.  0. 15. 77. 30.  7. 34. 42.  0.  0. 14. 39. 21.  3.  1.  5.  5.  2. 98. 117. 117.  9.  1.  4. 11 7. 27. 117. 117. 19.  4.
 2. 91. 77.  4.  5. 10.  0. 11. 32. 115.  0.  0.  7. 47. 36. 18.  3.  1.  6.  1.  6. 62. 117. 117. 79. 2 6. 117.  5.  5. 14.
 22. 39. 88. 117. 68. 26. 33.  7.  5.  6.  2. 31.  0.  0.  0.  1. 13. 24. 18.  6.  0.  0.  0.  0.  1. 22. 85. 21.  3.  0.  0.
 0.  1. 13. 82. 43.  1.  3. 20.  9. 12. 52. 31.  5.]
```

```
1734 [ 2.  1. 12.  9.  1.  0. 23. 106.  0.  0. 52. 28.  4.  6. 19. 13.  2.  1. 46. 117. 15.  6.  4.  5.  6.  0. 15. 77.
 30.  7. 34. 42.  0.  0. 14. 39. 21.  3.  1.  5.  5.  2. 98. 117. 117.  9.  1.  4. 117. 27. 117. 117. 19.  4.  2. 91.
 77.  4.  5. 10.  0. 11. 32. 115.  0.  0.  7. 47. 36. 18.  3.  1.  6.  1.  6. 62. 117. 117. 79. 26. 117.  5.  5. 14.
 22. 39. 88. 117. 68. 26. 33.  7.  5.  6.  2. 31.  0.  0.  0.  1. 13. 24. 18.  6.  0.  0.  0.  0.  1.  22. 85. 21.
  3.  0.  0.  0.  1. 13. 82. 43.  1.  3. 20.  9. 12. 52. 31.  5.]
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=10,    n_clusters=32, n_init=10, n_jobs=1,
compute_distances='auto',  random_state=None, tol=1.0, verbose=0)
(32, 128)
[[17.5  27.56060606 60.8030303  ... 27.53030303 33.39393939  20.01515152]
 [29.26785714 12.48214286 12.85714286 ... 14.58928571 15.125  22.26785714]
 [16.12658228 22.01265823 33.74683544 ... 30.5443038  48.73417722  27.93670886]
 ...
 [ 3.66666667  2.96296296  8.62962963 ... 22.85185185 41.2962963  28.81481481]
 [12.24657534 16.28767123 30.01369863 ... 20.79452055 24.45205479  22.64383562]
 [21.50704225 15.11267606 21.30985915 ... 11.16901408 15.94366197  16.63380282]]
```

07 B.所有图像特征聚类/词袋

A.提取每一幅图像特征(SIFT 128维) 04

```
'''定义类处理词袋模型和向量量化'''
class BagOfWords(object):
    def __init__(self,num_clusters=32):
        self.num_dims=128
        self.num_clusters=num_clusters #KMeans聚类参数：定义聚类数量，关于聚类可以参考之前实验“机器学习-聚类”部分
        self.num_retries=10 #KMeans聚类参数：Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

    '''KMeans聚类'''
    def cluster(self,datapoints):
        kmeans=KMeans(self.num_clusters,n_init=max(self.num_retries,1),max_iter=10,tol=1.0)
        res=kmeans.fit(datapoints)
        centroids=res.cluster_centers_
        return kmeans,centroids #返回KMeans聚类模型，以及聚类中心点

    '''归一化数据。其它方法可以参考之前相关实验处理技术'''
    def normalize(self,input_data):
        sum_input=np.sum(input_data)
        if sum_input>0:
            return input_data/sum_input #单一数值/总体数值之和，最终数值范围[0,1]
        else:
            return input_data

    '''提取SIFT特征'''
    def construct_feature(self,img,kmeans,centroid):
        keypoints=StarFeatureDetector().detect(img) #应用处理Star特征检测相关函数，返回检测出的特征关键点
        keypoints,feature_vectors=compute_sift_features(img,keypoints) #应用SIFT提取特征描述信息
        labels=kmeans.predict(feature_vectors) #对特征执行聚类预测类标
        print(labels)
        feature_vector=np.zeros(self.num_clusters)
        for i,item in enumerate(feature_vectors): #计算特征聚类出现的频数/直方图
            # print(i,labels[i])
            feature_vector[labels[i]]+=1
            # print(feature_vector)
            feature_vector_img=np.reshape(feature_vector,((1,feature_vector.shape[0])))
        return self.normalize(feature_vector_img)
```

11 12 13

C.构造码本 10

```
def get_feature_map(input_map,kmeans,centroids,scaling_size):
    feature_map=[]
    for item in input_map:
        temp_dict={}
        temp_dict['object_class']=item['object_class']
        print("Extracting feature for",item['image_path'])
        img=cv2.imread(item['image_path'])
        # img=resize_image(img,scaling_size)
        temp_dict['feature_vector']=BagOfWords().construct_feature(img,kmeans,centroids)
        if temp_dict['feature_vector'] is not None:
            feature_map.append(temp_dict)
    print(feature_map[0]['feature_vector'].shape,feature_map[0])
    return feature_map
```

09

```
Extracting feature for static/images/imresize/img_0334.jpg
[31 31 31  5 20 20  5  5  5 20 20  5  5 20 20  5  5  5  5  5 20  5  5  5  5  5  5 31  5  5  5 20 31 31 20 20 20
 16 20 20 11 23 23  9  0  0 13]
```

```
'''调整图像大小，因已经处理过图像大小，此处定义函数未使用。一般直接使用misc.imresize()方法调整图像大小更加简便'''
def resize_image(input_img, new_size):
    h, w = input_img.shape[:2]
    scaling_factor = new_size / float(h)
    if w < h:
        scaling_factor = new_size / float(w)
    new_shape = (int(w * scaling_factor), int(h * scaling_factor))
    return cv2.resize(input_img, new_shape) #使用cv2.resize()方法调整图像大小
```

```
(1, 32) {'object_class': 'good', 'feature_vector': array([[0.05154639, 0.01030928, 0.01030928, 0.        , 0.05670103,
        0.11340206, 0.        , 0.01030928, 0.        , 0.01030928,
        0.05154639, 0.00515464, 0.06185567, 0.06701031, 0.01030928,
        0.        , 0.01030928, 0.09793814, 0.02061856, 0.01030928,
        0.11340206, 0.        , 0.        , 0.01030928, 0.02061856,
        0.03608247, 0.0257732 , 0.1185567 , 0.        , 0.07216495,
        0.        , 0.00515464]])}
```