

ARCHIVE BI OG **GITHUB WFIBO**

验证码破解技术四部曲之使用K近邻算法(三)

Sep 22, 2016

#前言

在上一节中,我们使用了google的开源OCR库来对字符进行识别,这一节以及下一节我们将要使用机器学 习算法来识别验证码。本节的代码都在https://github.com/nladuo/captcha-break/tree/master/csdn可以找 到。

卡下载验证码

在这一节中,将要对CSDN下载的验证码进行破解,就是在http://download.csdn.net/下载东西的时候,短时 间内下载次数过多弹出来的验证码。

做机器学习的第一个步骤就是采集数据,构建训练样本。首先,来看一下CSDN下载中出现的验证码。



在每次刷新的时候,会有以上这两种验证码出现。在本节中,为了方便学习K近邻算法(简称为: KNN), 选择第二种来进行破解,因为第二种的字母分割十分容易,每个字母的位置都是固定的。

由于两种验证码的图片大小不一样,所以可以使用图片大小来判断哪个是第一种验证码,哪个是第二种验 证码,这里使用python进行验证码下载。

PYTHON

```
# coding:utf-8
import requests
import uuid
from PIL import Image
import os
```

url = "http://download.csdn.net/index.php/rest/tools/validcode/source_ip_validate/10.5711163911 for i in range(100):

resp = requests.get(url)

```
filename = "./captchas/" + str(uuid.uuid4()) + ".png"
with open(filename, 'wb') as f:
    for chunk in resp.iter_content(chunk_size=1024):
        if chunk: # filter out keep-alive new chunks
            f.write(chunk)
            f.flush()
    f.close()
im = Image.open(filename)
if im.size != (48, 20):
   os.remove(filename)
else:
   print filename
```

† 分割字符

下载过后,就需要对字母进行分割。机器学习虽然牛逼,但是也需要对样本进行预处理,这里的预处理就 是把字母分割出来,并且分割成同样的尺寸。分割的方式可以使用代码分割,当然也可以通过人用PS等工 具进行手动分割。

我这里使用代码分割,字母分割的代码在spliter文件夹下,我使用了boost库来来读取所有下载的验证码, 对图片进行二值化后,进行定点分割,可以看到分割好的字母如下。

cspace cpp-program	captcha-break csdn sp	oliter letters				Q ≡ :
00a5b33e-7568-	0a53e196-89ef-	0ad4f314-52e6-	0c04c1b1-2be9-	0c7ebb9a-7419-	0d1429c6-55bd-	0f189c57-2387-
486f-8dea-	40fd-b005-	4e1f-abd7-	4375-aa25-	49f3-9baf-	4714-9d42-	4625-87b2-
1841c6cfe9b2.png	53a002cdee88.png	15b83a5c88df.png	1294116a59c6.png	e49dc76abeaa.png	f49d42af8529.png	eb55462af085.png
0fdbadec-de74-	1aa7e5a6-6185-	1ad4d3fb-4e20-	1af34526-a44e-	1b4a00f0-56a4-	1b130d78-a75f-	1bb86d1d-eb6b-
4c39-8428-	44c6-896c-	44e9-97d5-	4aec-9a8d-	41f4-a740-	4330-b235-	47c0-86d7-
90500dffc711.png	9bb4a67a025d.png	9b36e565d374.png	3877b9172b7c.png	e384332e3d42.png	9f17f553eb75.png	fbbd4aaffae9.png
01bc5346-c095-	1cc12930-5d56-	1f3c87e9-51b1-	1f5ea154-63a5-	2a5aa215-3b06-	2ae666fd-338d-	2b004d42-a8a5-
4eb5-98b7-	4338-b5fa-	4699-af2a-	43f0-8bb5-	4e91-9e10-	42f1-aa77-	4eb6-a183-
ba3954d44127.png	3f0aa4e24d12.png	5cd5b70c6b13.png	503fde30adb4.png	9c917ba950e6.png	9402b207874d.png	60a146708927.png
2b5f85b5-d95c-	2b12d3f9-ced5-	2b66ee2f-a4bb-	2c1e850c-8497-	2c6d7d0a-475a-	2cb60d6a-0220-	2e3dcfb3-607e-
4a4a-b925-	4f58-bc43-	4d01-9996-	4a6d-8b64-	48aa-b16f-	4398-87f9-	455c-ae3b-
8782efea2cd5.png	9bb2d62809b2.png	37018626b02d.png	45b33702ba86.png	60bc1fd4fbae.png	8ac228cec933.png	6b804cbe3754.png
2ef93531-9ee7-	3ae74ce9-8386-	3c579be4-7ab4-	3d0a3126-d1ef-	003d700d-47b2-	3e4b3d9e-9c5b-	3e83c51f-58b2-
4a3b-8671-	4e1a-87f3-	4c31-bd0c-	493f-8c90-	49b8-8b0d-	4e68-b08d-	40d1-bb0e-
365d46cb3966.png	9e7403d2953e.png	ab477c76273c.png	190586bfbe97.png	af42ae355d9e.png	d1049fee6330.png	fef77e4cf13a.png

之后,需要人工对字母进行分类,分类好的图片见recognizer/dataset,我这里每个字母需要6个样本,10个 字母,总共60个样本。

		contribution because and		
rkspace	cpp-program	captcha-break csdn	recognizer dataset	٩ 🔳
				8
	01.png	02.png	03.png	04.png
	2	2	I	
	05.png	06.png	11.png	12.png
			1	1
	13.png	14.png	15.png	16.png
1	2	2	2	2
	21.png	22.png	23.png	24.png
	2		Ē	
	25.png	26.png	31.png	32.png
	3	8		2
1	33.png	34.png	35.png	36.png
		2	2	
	41.png	42.png	43.png	44.png
	4	S		
	45.png	46.png	51.png	52.png

‡算法原理

K近邻算法的定义十分简单,在百度百科上有这样的解释:如果一个样本在特征空间中的k个最相似(即特征 空间中最邻近)的样本中的大多数属干某一个类别,则该样本也属干这个类别。

也就是说,需要找到要识别的字母在训练样本中K个最近的字母,然后找出这K个字母中最多的是某个类 的?要识别的图片也就是该类的。

‡实现KNN

† 计算距离

首先,先定义一下距离如何计算,这里可以用各种数学上的距离,欧式距离、马氏距离等等。。

由于我们的图片已经进行了二值化,为了简便起见,这里把两张图片的距离定义为:两张图片灰度不同的 像素点个数。也就是逐个比较图片的相对位置上的灰度值,如果不相同,距离就加一。

```
C++
```

```
int count_distance(Mat mat1, Mat mat2)
    assert(mat1.size().height == mat2.size().height);
    assert(mat1.size().width == mat2.size().width);
    assert(mat1.channels() == 1 && mat2.channels() == 1);
    int distance = 0:
    for(int i = 0; i < mat1.size().width; i++){</pre>
        for(int j = 0; j < mat1.size().height; j++){</pre>
            if(mat1.at<uchar>(j, i)!= mat2.at<uchar>(j, i)){//不相等就加1
                distance++;
        }
    return distance;
}
```

广加载数据

数据的加载需要一个图片数组和一个标签数组,来记录图片数组相应位置的类别。

加载样本数据:

```
C++
void load_dataset(Mat dataset[])
{
    string dataset_dir = "../recognizer/dataset/";
    for(int i = 0; i < 6*10; i++){
        char buffer[255];
        sprintf(buffer, "%d", i/6);
        string image_path = dataset_dir + string(buffer);
        sprintf(buffer, "%d", i%6 + 1);
        image_path += string(buffer) + ".png";
        dataset[i] = imread(image_path, CV_LOAD_IMAGE_GRAYSCALE);
   }
}
```

加载样本数据标签:

```
C++
void create_labels(int labels[])
{
    for(int i = 0; i < 6*10; i++){
       labels[i] = i/6;
}
```

‡算法实现

加载完数据后,就可以开始实现KNN分类了。

1、计算输入图片和所有其他图片的距离

```
C++
int distances[6*10];
int sorted_distances[6*10];
//count distances
for(int i = 0; i < 6*10; i++){
    distances[i] = count_distance(letter, dataset[i]);
    sorted_distances[i] = distances[i];
}
```

2、对距离进行排序

```
C++
sort(sorted_distances, sorted_distances+6*10);
```

3、获取K个距离最近的图片的类别

```
C++
int* k_nearest = new int[k];
for(int i = 0; i < k; i++){
    for(int j = 0; j < 6*10; j++){
        if(distances[i] == sorted_distances[i]){
            k_nearest[i] = labels[j];
            break;
        }
    }
}
```

4、利用map记录所有类别中出现k_nearest的次数

```
C++
map<int, int> labels_map;
for(int i = 0; i < k; i++){
    if(labels_map.find(k_nearest[i]) == labels_map.end())
        labels_map[k_nearest[i]] = 0;
    else
        labels_map[k_nearest[i]]++;
}
```

5、得到出现最多的类别

```
C++
int max_label = -1;
labels_map[max_label] = -1;
map<int,int>::iterator it;
for(it=labels_map.begin();it!=labels_map.end();++it){
    if(it->second > labels_map[max_label]){
        max_label = it->first;
    }
delete[] k_nearest;
return max_label;
```

片识别验证码

最后,我们把验证码的4个字母分割出来,再进行K近邻分类,就可以得到识别结果了。

```
C++
void recognize(string path, Mat dataset[], int labels[])
    Mat test_image = imread(path, CV_LOAD_IMAGE_GRAYSCALE);
    threshold(test_image, test_image, 100, 255, cv::THRESH_BINARY);
    Range col_ranges[4] = {
        Range(5, 5+8),
        Range(14, 14+8),
        Range(23, 23+8),
        Range(32, 32+8)
    }:
    cout<<"Result:";</pre>
    for(int i = 0; i < 4; i++){
        Mat letter = test_image.colRange(col_ranges[i]);
        cout << knn_classify(letter, dataset, labels, 5);</pre>
    cout<<endl;
}
```

;效果

识别图片:



识别结果:

```
kalen@kalen-ThinkPad-S3-S431:~/Workspace/cpp-program/captcha-break/csdn/recognizer$ ./recognizer test1.png
Result:6759
kalen@kalen-ThinkPad-S3-S431:~/Workspace/cpp-program/captcha-break/csdn/recognizer$ ./recognizer test2.png
Result:6281
kalen@kalen-ThinkPad-S3-S431:~/Workspace/cpp-program/captcha-break/csdn/recognizer$ ./recognizer test3.png
Result:5247
kalen@kalen-ThinkPad-S3-S431:~/Workspace/cpp-program/captcha-break/csdn/recognizer$ ./recognizer test4.png
Result:5394
kalen@kalen-ThinkPad-S3-S431:~/Workspace/cpp-program/captcha-break/csdn/recognizer$
```

#练习

通过以上,我们破解了CSDN下载的第二种验证码,第一种验证码的识别过程也是可以使用KNN的,但是第 一种和第二种的分割字母的方式不同,读者可以尝试使用opencv的findCountours函数对字母进行分割,或 者使用垂直投影的方式进行分割,需要注意的是第一种验证码有一个黑色的边框,如果不处理会影响 findCountours函数的效果。

				PREV	NEXT			
社交帐号登	登录: 微信	微博	QQ	人人 更多»				
9,	说点什么吧							
							I	
							发花	व
0条评论						揖	最新 最早	最热
				还没有评论,流	少发等你来抢			
						True	e's ME正在你	吏用多说

© 2015 - 2016 nladuo, powered by Hexo and hexo-theme-apollo.