



## 验证码破解技术四部曲之使用K近邻算法（三）

Sep 22, 2016

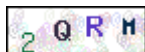
### 前言

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。

### 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。

```
# 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)
```

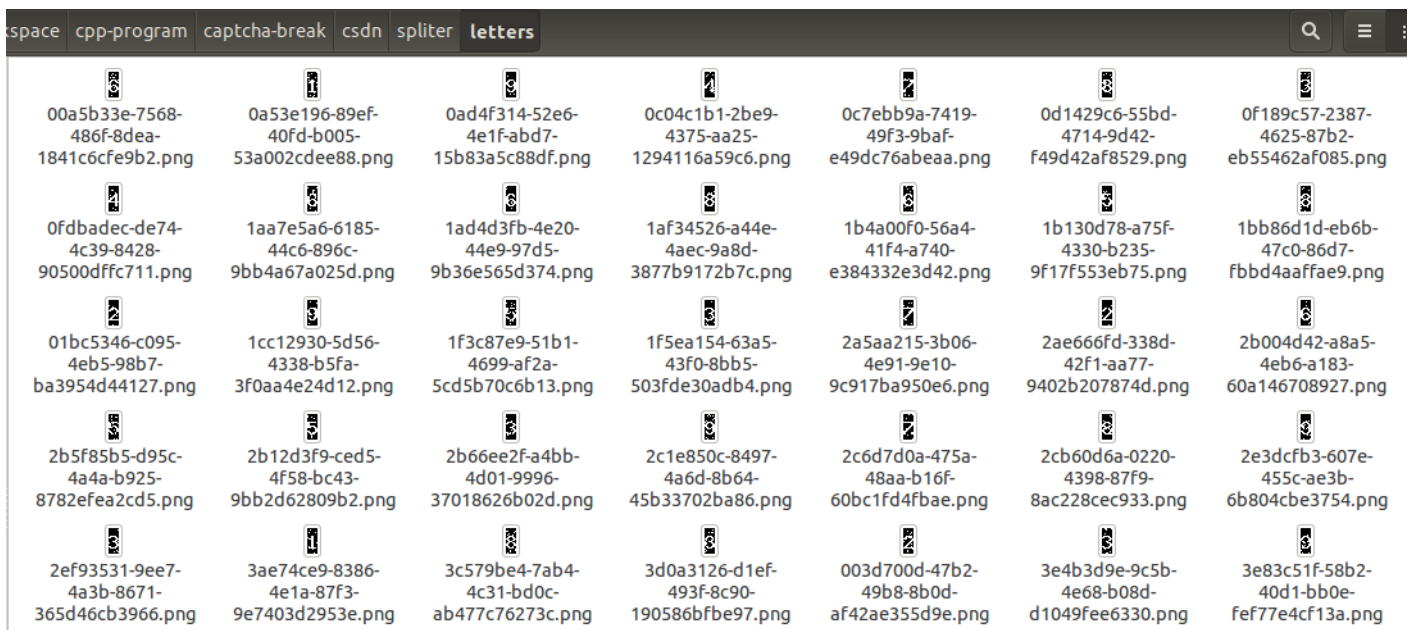
PYTHON

```
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
```

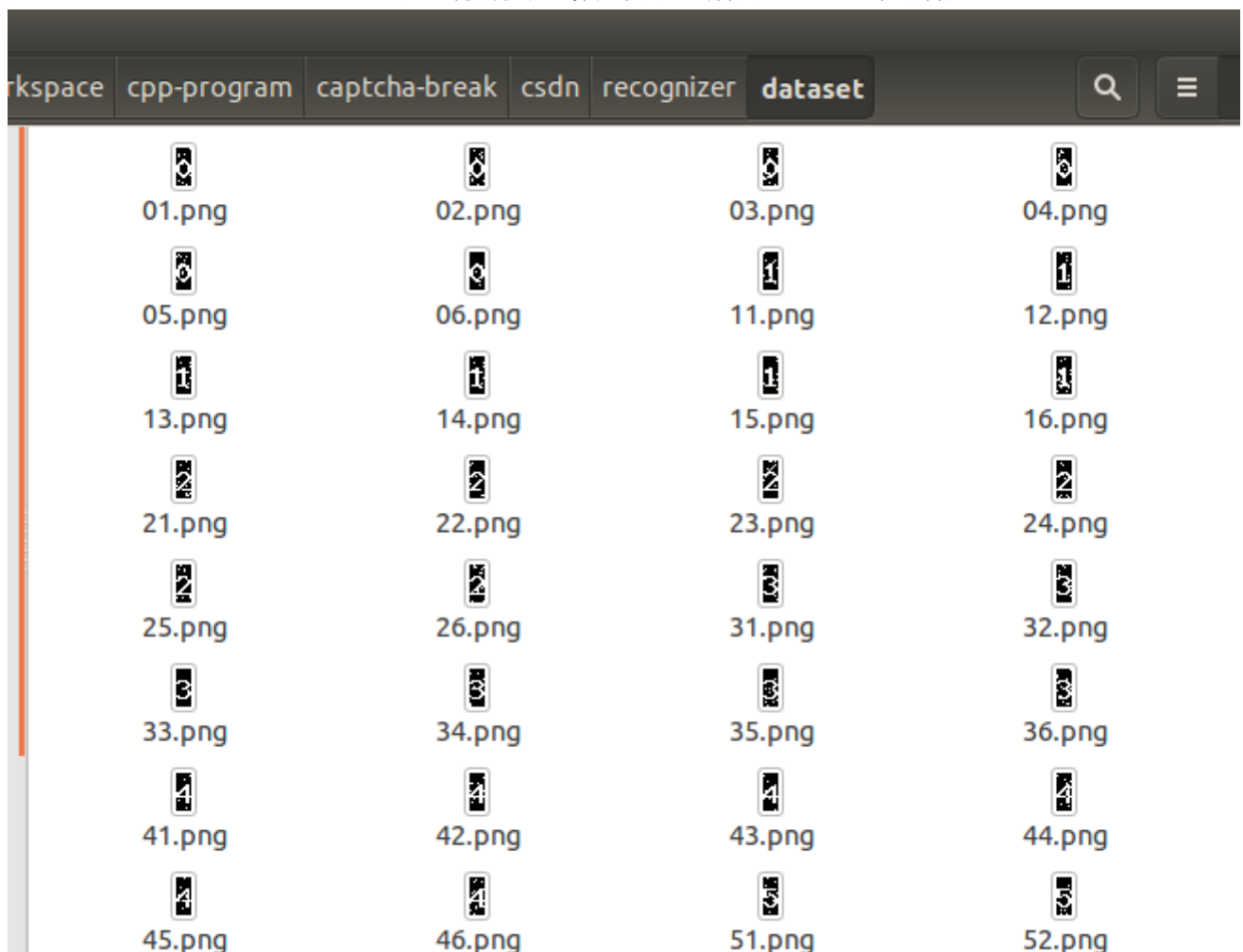
## 分割字符

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。



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。



## 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于某一个类别，则该样本也属于这个类别。

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。

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

## 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：

```
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);
    }
}
```

C++

加载样本数据标签：

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

C++

## 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

```
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];
}
```

C++

### 2、对距离进行排序

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

C++

### 3、获取K个距离最近的图片的类别

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

C++

### 4、利用map记录所有类别中出现k\_nearest的次数

```
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]]++;
}
```

C++

### 5、得到出现最多的类别

```
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;
```

C++

## 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。

```
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:";
    for(int i = 0; i < 4; i++){
        Mat letter = test_image.colRange(col_ranges[i]);
        cout << knn_classify(letter, dataset, labels, 5);
    }
    cout<<endl;
}
```

C++

## 效果

识别图片：



test1.png



test2.png



test3.png



test4.png

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

社交帐号登录:    微信    微博    QQ    人人    更多»



说点什么吧...

发布

0条评论

最新   最早   最热

还没有评论，沙发等你来抢

True's ME正在使用多说