

**《机器学习》**

**期末实验报告**

作 者**： 王昊**

导 师**： 李征 副研究员**

系别年级**： 心理学部 2018级**

学科专业**： 计算机应用技术**

学 号**： 201821061107**

# 基于SVM的图片识别器

## 一、实验介绍

### 1.1 实验内容

SVM（支持向量机）是一种常用的机器学习分类算法。本课程使用HOG+SVM算法和OpenCV实现一个图片分类器，通过自己训练分类器，达到可以判断任意图片是否是猫的效果。

### 1.2 实验知识点

* HOG+SVM分类器的基本原理
* OpenCV处理图片
* 训练分类器，以得到适合自己项目的分类器
* Python文件操作

### 1.3 实验环境

* python2.7
* Xfce终端

## 二、实验原理

SVM（支持向量机）分类器的原理是利用“分类超平面”来实现数据分类。在利用“分类超平面”对数据进行划分时，遵循“间距最大”原则。例如，将二维平面内的两组数据分类，可以确定很多个“分类超平面”，在二维维度下，超平面退化为一条直线。

如果分类3维数据，我们就使用一个平面来分割数据。如果分类4维数据，我们将会使用一个体来分割数据。以此类推，如果分类1024维数据，我们将使用1023维平面来分割数据。所以这个时候，将1023维度的平面命名为“分类超平面”。

SVM是一个由分类超平面定义的判别分类器。也就是说给定一组带标签的训练样本，算法将会输出一个最优超平面对新样本(测试样本)进行分类。

这也是监督类型机器学习的特点，即，把一堆带有标签的数据输入到机器中，让机器根据给定的数据计算出规则，再利用这个规则，去对未知数据进行分类。说白了，就是先积累几年工作经验，然后去工作。

本实验是读入输入图片的灰度图，即黑白的。然后计算该图片的hog值，将计算得到的结果作为向量来代表该图片。对由很多张图片组成的向量集进行计算，找到最大间距的分类超平面，进而分类数据。

hog的全称是Histogram of Oriented Gradient, HOG，即方向梯度直方图。它是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。Hog特征结合SVM分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。HOG+SVM进行行人检测的方法是法国研究人员Dalal在2005的CVPR上提出的，今天的很多行人检测算法基本都是以HOG+SVM的思路。

* 主要思想：在一副图像中，局部目标的表象和形状（appearance and shape）能够被梯度或边缘的方向密度分布很好地描述。（本质：梯度的统计信息，而梯度主要存在于边缘的地方）。
* 具体的实现方法是：首先将图像分成小的连通区域，我们把它叫细胞单元。然后采集细胞单元中各像素点的梯度的或边缘的方向直方图。最后把这些直方图组合起来就可以构成特征描述器。
* 提高性能：把这些局部直方图在图像的更大的范围内（我们把它叫区间或block）进行对比度归一化（contrast-normalized），所采用的方法是：先计算各直方图在这个区间（block）中的密度，然后根据这个密度对区间中的各个细胞单元做归一化。通过这个归一化后，能对光照变化和阴影获得更好的效果。
* 优点：与其他的特征描述方法相比，HOG有很多优点。首先，由于HOG是在图像的局部方格单元上操作，所以它对图像几何的和光学的形变都能保持很好的不变性，这两种形变只会出现在更大的空间领域上。其次，在粗的空域抽样、精细的方向抽样以及较强的局部光学归一化等条件下，只要行人大体上能够保持直立的姿势，可以容许行人有一些细微的肢体动作，这些细微的动作可以被忽略而不影响检测效果。因此HOG特征是特别适合于做图像中的人体检测的。

## 三、开发准备

打开Xfce终端，下载并安装OpenCV的相关依赖。

1. $ sudo pip install numpy
2. $ sudo apt-get install python-opencv

遇到是否安装的询问时，输入y，按回车键继续安装。安装时间较长，并且视网络状态而定。

下载实验所需的图片数据：

1. $ wget http://labfile.oss.aliyuncs.com/courses/794/cat.zip
2. $ wget http://labfile.oss.aliyuncs.com/courses/794/other.zip
3. $ wget http://labfile.oss.aliyuncs.com/courses/794/predict.zip

这三组数据分别是含有猫的图片，没有猫的图片，以及用于测试SVM分类器的数据集。

下载后，解压得到图片：

1. $ unzip cat.zip
2. $ unzip other.zip
3. $ unzip predict.zip

这些图片都是从网上下载的。如果想使用自己下载的图片，也没有问题。需要注意爹是，输入到分类器的图片都是固定像素的。我们需要对下载的图片数据进行处理，使其符合我们程序的要求。将大图片裁减成固定像素的小图片的程序如下：

1. *# -\*- coding: utf-8 -\*-*
2. import numpy as np
3. import cv2
4. from os.path import dirname, join, basename
5. from glob import glob
7. num=0
8. for fn in glob(join(dirname(\_\_file\_\_)+'\other', '\*.jpg')):
9. img = cv2.imread(fn)
10. res=cv2.resize(img,(64,128),interpolation=cv2.INTER\_AREA)
11. cv2.imwrite(r'D:\ECLIPSE-PROJECT\Python\my\_opencv\other\_64\_128\test'+str(num)+'.jpg',res)
12. num=num+1
14. print 'all done!'
16. cv2.waitKey(0)
17. cv2.destroyAllWindows()

使用程序时，请替换输出路径为一个已存在的路径，即替换这一句中的路径：

cv2.imwrite(r'D:\ECLIPSE-PROJECT\Python\my\_opencv\other\_64\_128\test'+str(num)+'.jpg',res)

这段代码会扫描Python脚本所在的文件夹的子文件夹other文件夹下的所有.jpg文件，然后使用OpenCV读取图片数据，并按照指定的大小进行缩放，将缩放后的结果写入到指定目录下的指定图片中。

## 四、实验步骤

### 4.1 训练数据集

首先，我们根据已经分类好的数据集来对分类器进行训练。我们的cat文件夹下全是猫的照片，而other文件夹下全不是猫，已经完成了贴标签这个过程了。让程序从这两组数据里学习，计算分类的方法。

使用HOG+SVM算法进行训练前，需要先计算每张图片的HOG值以得到供SVM分类器使用的输入向量。计算该值的算法实现的一般过程为：

* 灰度化（OpenCV处理图像时，一般都处理为灰度图像，忽略颜色干扰）
* 采用Gamma校正法对输入图像进行颜色空间的标准化（归一化）；目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音的干扰；
* 计算图像每个像素的梯度（包括大小和方向）；主要是为了捕获轮廓信息，同时进一步弱化光照的干扰。
* 将图像划分成小cells（例如6\*6像素/cell）；
* 统计每个cell的梯度直方图（不同梯度的个数），即可形成每个cell的descriptor；
* 将每几个cell组成一个block（例如3\*3个cell/block），一个block内所有cell的特征descriptor串联起来便得到该block的HOG特征descriptor。
* 将图像image内的所有block的HOG特征descriptor串联起来就可以得到该image（你要检测的目标）的HOG特征descriptor了。这个就是最终的可供分类使用的特征向量了。

在本实验中，没有严格按照上述的过程实现，我们采用了下述方法：我们在每个cell内计算X和Y方向的Sobel导数。然后找到每个像素的梯度和方向。此梯度被量化为16＊16个整数值。把每个图像分成四个子图方块。对于每个子正方形，计算加权其幅度的方向（16＊16bins）的直方图。因此，每个子图给我们一个包含16＊16个值的向量。四个这样的向量（分别代表四个子图的16\*16向量）一起给我们一个特征向量包含1024个值。这就是我们用来训练数据的特征向量。这部分的代码如下所示：

1. bin\_n = 16\*16 *# Number of bins*
3. def hog(img):
4. x\_pixel,y\_pixel=194,259
5. gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)
6. gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)
7. mag, ang = cv2.cartToPolar(gx, gy)
8. bins = np.int32(bin\_n\*ang/(2\*np.pi)) *# quantizing binvalues in (0...16)*
9. bin\_cells = bins[:x\_pixel/2,:y\_pixel/2], bins[x\_pixel/2:,:y\_pixel/2], bins[:x\_pixel/2,y\_pixel/2:], bins[x\_pixel/2:,y\_pixel/2:]
10. mag\_cells = mag[:x\_pixel/2,:y\_pixel/2], mag[x\_pixel/2:,:y\_pixel/2], mag[:x\_pixel/2,y\_pixel/2:], mag[x\_pixel/2:,y\_pixel/2:]
11. hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) for b, m in zip(bin\_cells, mag\_cells)]
12. hist = np.hstack(hists) *# hist is a 64 bit vector*
13. *# print hist.shape*
14. *# print type(hist)*
15. return hist

完整的代码如下所示，程序首先扫描cat和other文件夹内的图片，然后用灰度方式读入，计算每个图片的hog值，然后建立SVM分类器，使用输入的数据进行训练，将训练结果保存于svm\_cat\_data.dat文件中。

1. *#file name:train.py*
2. import numpy as np
3. import cv2
4. *#from matplotlib import pyplot as plt*
5. from os.path import dirname, join, basename
6. import sys
7. from glob import glob

10. bin\_n = 16\*16 *# Number of bins*
12. def hog(img):
13. x\_pixel,y\_pixel=194,259
14. gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)
15. gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)
16. mag, ang = cv2.cartToPolar(gx, gy)
17. bins = np.int32(bin\_n\*ang/(2\*np.pi)) *# quantizing binvalues in (0...16)*
18. bin\_cells = bins[:x\_pixel/2,:y\_pixel/2], bins[x\_pixel/2:,:y\_pixel/2], bins[:x\_pixel/2,y\_pixel/2:], bins[x\_pixel/2:,y\_pixel/2:]
19. mag\_cells = mag[:x\_pixel/2,:y\_pixel/2], mag[x\_pixel/2:,:y\_pixel/2], mag[:x\_pixel/2,y\_pixel/2:], mag[x\_pixel/2:,y\_pixel/2:]
20. hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) for b, m in zip(bin\_cells, mag\_cells)]
21. hist = np.hstack(hists) *# hist is a 64 bit vector*
22. *# print hist.shape*
23. *# print type(hist)*
24. return hist
26. *#print glob(join(dirname(\_\_file\_\_)+'/cat','\*.jpg'))*
27. img={}
28. num=0
29. for fn in glob(join(dirname(\_\_file\_\_)+'/cat', '\*.jpg')):
30. img[num] = cv2.imread(fn,0)*#参数加0，只读取黑白数据，去掉0，就是彩色读取。*
31. *# print img[num].shape*
32. num=num+1
33. print num,' num'
34. positive=num
35. for fn in glob(join(dirname(\_\_file\_\_)+'/other', '\*.jpg')):
36. img[num] = cv2.imread(fn,0)*#参数加0，只读取黑白数据，去掉0，就是彩色读取。*
37. *# print img[num].shape*
38. num=num+1
39. print num,' num'
40. print positive,' positive'
42. trainpic=[]
43. for i in img:
44. *# print type(i)*
45. trainpic.append(img[i])
47. svm\_params = dict( kernel\_type = cv2.SVM\_LINEAR,
48. svm\_type = cv2.SVM\_C\_SVC,
49. C=2.67, gamma=5.383 )
50. *#img = cv2.imread('02.jpg',0)*
51. *#hist\_full = cv2.calcHist([img],[0],None,[256],[0,256])*
52. *#print hist\_full*
53. *#plt.plot(hist\_full)*
54. *#plt.show()*
56. *#img1 = cv2.imread('02.jpg',0)*
57. *#temp=img[0].ravel()*
58. *#print temp*
59. *#print len(temp)*
60. temp=hog(img[0])
61. print temp.shape
63. *#hogdata = [map(hog,img[i]) for i in img]*
64. hogdata = map(hog,trainpic)
65. print np.float32(hogdata).shape,' hogdata'
66. trainData = np.float32(hogdata).reshape(-1,bin\_n\*4)
67. print trainData.shape,' trainData'
68. responses = np.float32(np.repeat(1.0,trainData.shape[0])[:,np.newaxis])
69. responses[positive:trainData.shape[0]]=-1.0
70. print responses.shape,' responses'
71. print len(trainData)
72. print len(responses)
73. print type(trainData)
75. svm = cv2.SVM()
76. svm.train(trainData,responses, params=svm\_params)
77. svm.save('svm\_cat\_data.dat')

注意，如果想要运行此程序并得到正确的结果，需要在控制台输入：

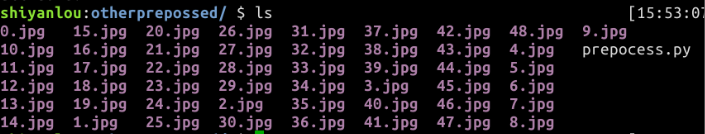
$ python /home/shiyanlou/train.py

如果只输入：

$ python train.py

也能看到输出，只不过执行到一半会报错。

这其中的原因，主要是程序中使用了glob包来枚举文件夹下的某个类型的文件。

正常运行后，可以见到文件夹下生成的数据：

### 4.2 使用训练好的SVM分类器进行分类

机器学习是一个不断迭代的过程。训练的数据集越大越好，训练时间当然也是越长效果越好。当机器认错了图片的时候，我们要把这个图片拿出来，标记正确，输入机器再训练一遍，如此迭代下去。本实验只训练了一次以演示原理。

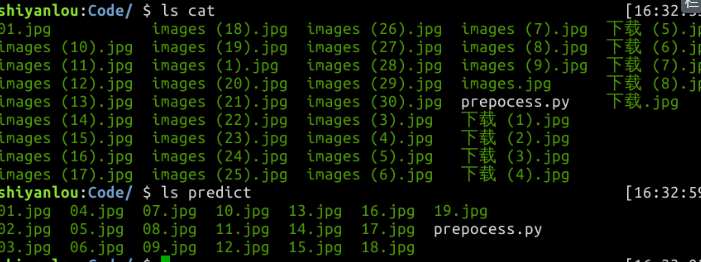
我们得到了训练好的数据svm\_cat\_data.dat后，可以用它来分类测试图片。

建立程序如下：

1. *#file name:predict.py*
2. import numpy as np
3. import cv2
4. *#from matplotlib import pyplot as plt*
5. from os.path import dirname, join, basename
6. import sys
7. from glob import glob
9. *#my\_svm=cv2.SVM()*
10. *#my\_svm*
11. *#bin\_n = 16 # Number of bins*
12. bin\_n = 16\*16 *# Number of bins*
14. def hog(img):
15. x\_pixel,y\_pixel=194,259
16. gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)
17. gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)
18. mag, ang = cv2.cartToPolar(gx, gy)
19. bins = np.int32(bin\_n\*ang/(2\*np.pi)) *# quantizing binvalues in (0...16)*
20. bin\_cells = bins[:x\_pixel/2,:y\_pixel/2], bins[x\_pixel/2:,:y\_pixel/2], bins[:x\_pixel/2,y\_pixel/2:], bins[x\_pixel/2:,y\_pixel/2:]
21. mag\_cells = mag[:x\_pixel/2,:y\_pixel/2], mag[x\_pixel/2:,:y\_pixel/2], mag[:x\_pixel/2,y\_pixel/2:], mag[x\_pixel/2:,y\_pixel/2:]
22. hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) for b, m in zip(bin\_cells, mag\_cells)]
23. hist = np.hstack(hists) *# hist is a 64 bit vector*
24. *# print hist.shape*
25. *# print type(hist)*
26. return hist
28. *#print glob(join(dirname(\_\_file\_\_)+'/cat','\*.jpg'))*
29. img={}
30. num=0
31. for fn in glob(join(dirname(\_\_file\_\_)+'/cat', '\*.jpg')):
32. img[num] = cv2.imread(fn,0)*#参数加0，只读取黑白数据，去掉0，就是彩色读取。*
33. *# print img[num].shape*
34. num=num+1
35. print num,' num'
36. positive=num
37. for fn in glob(join(dirname(\_\_file\_\_)+'/other', '\*.jpg')):
38. img[num] = cv2.imread(fn,0)*#参数加0，只读取黑白数据，去掉0，就是彩色读取。*
39. *# print img[num].shape*
40. num=num+1
41. print num,' num'
42. print positive,' positive'
44. trainpic=[]
45. for i in img:
46. *# print type(i)*
47. trainpic.append(img[i])
49. svm\_params = dict( kernel\_type = cv2.SVM\_LINEAR,
50. svm\_type = cv2.SVM\_C\_SVC,
51. C=2.67, gamma=5.383 )
53. temp=hog(img[0])
54. print temp.shape
56. *#hogdata = [map(hog,img[i]) for i in img]*
57. hogdata = map(hog,trainpic)
58. print np.float32(hogdata).shape,' hogdata'
59. trainData = np.float32(hogdata).reshape(-1,bin\_n\*4)
60. print trainData.shape,' trainData'
61. responses = np.float32(np.repeat(1.0,trainData.shape[0])[:,np.newaxis])
62. responses[positive:trainData.shape[0]]=-1.0
63. *#print responses[40:80]*
64. print responses.shape,' responses'
65. print len(trainData)
66. print len(responses)
67. print type(trainData)
69. svm = cv2.SVM()
70. svm.load('svm\_cat\_data.dat')
72. img = cv2.imread('/home/shiyanlou/predict/01.jpg',0)
73. *#print img.shapes,' img\_test0'*
74. hogdata = hog(img)
75. testData = np.float32(hogdata).reshape(-1,bin\_n\*4)
76. print testData.shape,' testData'
77. result = svm.predict(testData)
78. print result
79. if result > 0:
80. print 'this pic is a cat!'
82. test\_temp=[]
83. for fn in glob(join(dirname(\_\_file\_\_)+'/predict', '\*.jpg')):
84. img=cv2.imread(fn,0)*#参数加0，只读取黑白数据，去掉0，就是彩色读取。*
85. test\_temp.append(img)
86. print len(test\_temp),' len(test\_temp)'
88. hogdata = map(hog,test\_temp)
89. testData = np.float32(hogdata).reshape(-1,bin\_n\*4)
90. print testData.shape,' testData'
91. result = [svm.predict(eachone) for eachone in testData]
92. print result

运行该程序。

程序输出结果如下：



## 五、实验总结

使用SVM机器学习算法和OpenCV实现了一个判断一张图片是否是猫的分类器。通过本课程的学习，学员应理解SVM分类器的原理，可以建立自己的图片分类器，训练分类器达到合适的分类精度。