

**模式识别大作业**

题 目 猫狗分类器问题

学 院 信息科学与工程

专 业 控制科学与工程

组 员 黄靖

指导教师 赵海涛

**完成日期： 2017 年 10 月23日**

**模式识别作业报告——猫狗分类器问题**

组员：黄靖

经过第二次课程的模式识别学习，在赵海涛老师的辛勤指导下，我对模式识别的CNN算法有了一定的了解，并通过本次针对猫狗分类数据集的评分预测实验来巩固所学内容。下面将详细说明我的解决过程。

**一、猫狗分类器题目描述**

给出一张猫或狗的图片，识别出这是猫还是狗。这种识别具有很重要的意义，比如：

Web服务为了进行保护，会防止一些计算机进行恶意访问或信息爬取，进而设立一些验证问题，这些验证问题对于人来说很容易做，但是对于计算机这很困难。这样的方法称为CAPTCHA（完全自动公开的图灵测试）或HIP（人类交互证明）。 HIP有很多用处，例如减少垃圾邮件，防止暴力破解密码等。

比较有名的Asirra（用于限制访问的动物图像识别）就是一个HIP，它会让用户识别图片信息，比如识别出图片中是猫还是狗。对于人来说这很容易，但是对于计算机很困难。以下是Asirra的一个例子：

寻找流浪宠物为其提供住所的网站——http://Petfinder.com，向微软研究院提供了超过三百万张猫和狗的图像，这些图片由美国各地成千上万的动物收容所手动分类。

对于要入侵的计算机，随机猜测一般是最简单的攻击方法。图片识别并不容易，因为图片之间不同的的背景，角度，姿势，亮度等都存在着巨大的差异，很难识别。

不过随着机器学习——尤其是神经网络的发展，这项工作精度可以达到60%以上。而60％分类器就已经能将12幅图像的猜测概率从1/4096提高到1/459。

本次实验主要使用的数据文件是train.zip（训练集）和test.zip（测试集）。其中train.zip文件包含了20000个猫和狗的jpg图片这些图片大小不尽相同，在这些图片上进行训练。(1=狗，0=猫)的标签其中id即为'.jpg'前面的文件编号。test.zip是用来预测的测试集数据。

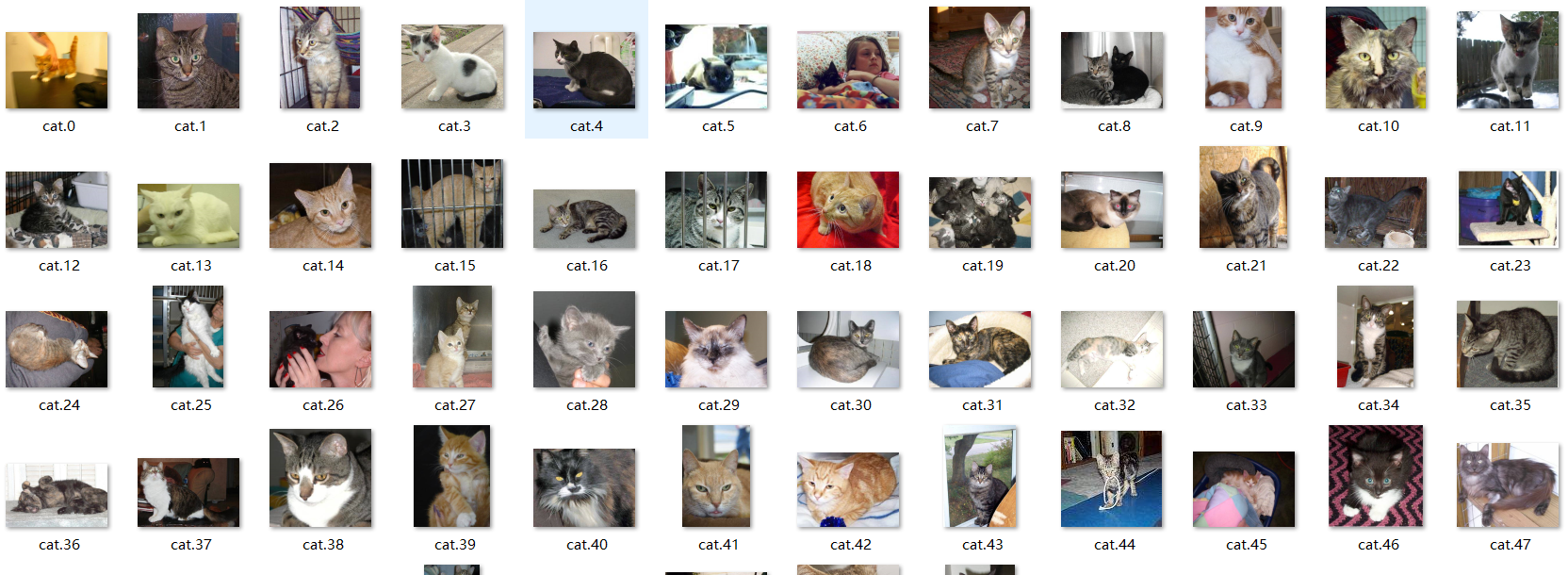
本次实验主要做的是前半部分（即预测评分），而后将预测的评分与test中的评分对比，且通过计算两者的RMSE（均方根误差）来评价本次预测的好坏。

**二、整体解决方案**

**2.1 下载，探索数据**

这道题提供了25000张猫狗图片，分为训练集（10000猫，10000狗），测试集（5000张，猫狗未标注）

图片大概如下所示：



这里使用python将所有图片都缩放到224\*224大小，保存到另一个文件夹中，做完这一步后可以把原始数据删掉，代码如下：

# 修改图片大小

for i in range(10000):

img\_path = '.\猫狗分类器data\train\cat.%d.jpg'%(i)

im = Image.open(img\_path)

resizedIm = im.resize((224, 224))

im.save(r'.\猫狗分类器data\\reshape\_train\cat.%d.jpg'%(i))

这里用到的缩放方法为按固定尺寸缩放图片，除此之外，还可以按照宽度或高度进行所需比例的缩放；按照生成图片文件大小进行处理（单位为KB）；按照图片长宽比进行分割。下面给出部分代码：

# coding=utf-8

import Image

import shutil

import os

class Graphics:

infile = 'D:\\myimg.jpg'

outfile = 'D:\\adjust\_img.jpg

@classmethod

def resize\_by\_width(cls, w\_divide\_h):

"""按照宽度进行所需比例缩放"""

im = Image.open(cls.infile)

(x, y) = im.size

x\_s = x

y\_s = x/w\_divide\_h

out = im.resize((x\_s, y\_s), Image.ANTIALIAS)

out.save(cls.outfile)

@classmethod

def resize\_by\_size(cls, size):

"""按照生成图片文件大小进行处理(单位KB)"""

size \*= 1024

im = Image.open(cls.infile)

size\_tmp = os.path.getsize(cls.infile)

q = 100

while size\_tmp > size and q > 0:

print q

out = im.resize(im.size, Image.ANTIALIAS)

out.save(cls.outfile, quality=q)

size\_tmp = os.path.getsize(cls.outfile)

q -= 5

if q == 100:

shutil.copy(cls.infile, cls.outfile)

**2.2数据预处理**

如果想要用这2W张训练图片直接训练出一个分类器的话，由于测试样本太少，分类器的泛化能力很差，因此我们为了提升泛化能力，就需要动用一些训练好的模型。

这里对分类器的泛化能力做简要介绍：在机器学习方法中，泛化能力通俗来讲就是指学习到的模型对未知数据的预测能力。在实际情况中，我们通常通过测试误差来评价学习方法的泛化能力。

2.2.1 泛化误差的定义：如果学到的模型是*f*^,那么用这个模型对未知数预测的误差即为泛化误差（generalization error）

泛化误差反映了学习方法的泛化能力，如果一种方法学习的模型比另一种方法学习的模型有更小的泛化误差，那么这种方法就更有效。事实上，泛化误差就是所学习到的模型的期望风险。

2.2.2 泛化误差上界

学习方法的泛化能力分析往往是通过研究泛化误差的概率上界进行的，简称为泛化误差上界（generalization error bound）。具体来说，就是通过比较两种学习方法的的泛化误差上界的大小来比较它们的优劣。

泛化误差上界的性质：它是样本容量的函数，当样本容量增加时，泛化上界趋于0；它是假设空间容量（capacity）的函数，假设空间容量越大，模型就越难学，泛化误差上界就越大。

这里使用VGG19网络，提取出VGG19网络最后一层卷积层的输出数据，所谓卷积层（Convolutional layer）是指，卷积神经网络中每层卷积层由若干卷积单元组成，每个卷积单元的参数都是通过反向传播算法最佳化得到的。卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网路能从低级特征中迭代提取更复杂的特征。初次运行时，可能需要下载VGG模型。每张照片经过模型，转换为了[7,7,512]大小的数据。

将2W张猫狗图片全部经过这样的处理后，保存到一个train\_data\_x.npy数组里面去，训练标签就设置为一个2w x 1大小的向量，猫用0表示，狗用1表示。同理对于5k张测试集数据，也这样保存到test\_data\_x.npy里面去。

代码如下：

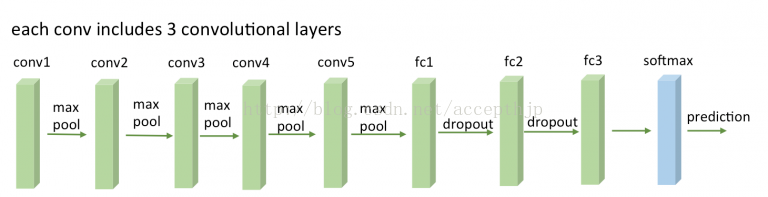
|  |
| --- |
| #数据预处理  #导入训练好的VGG模型  model=VGG19(weights=’imagenet’,include\_top=false)  train\_data\_x=np.zeros([20000,7,7,512]).astype(“float16”)  train\_data\_y=np.zeros([20000,]  train\_data\_y[10000:]=1  #猫的数据  for i in range(10000)  img\_path=’.\猫狗分类器data\\reshape\_train\cat.%d.jpg’%(i)  img=image.load\_img(img\_path,target\_size=(244,244))  x=image.image.img\_to\_array(img)  x=np.expand\_dims(x,axis=0)  x=preprocess\_input(x)  features=model.predict(x)  train\_data\_x[i,::]=features  printf(i)  #狗的数据  for i in range(10000):  img\_path=’.\猫狗分类器data\\reshape\_train\cat.%d.jpg’%(i)  img=imag.load\_img\_to\_array(img)  x=imag.img\_to\_array(img)  x=np.expend\_dims(x,axis=0)  x=preprocess\_input(x)  feature=model.predict(x)  train\_data\_x[i+10000,::]=features  print(i+10000)  np.save(“./猫狗分类器data/train\_data\_x.npy”,train\_data\_x)  np.save(“./猫狗分类器data/train\_data\_y.npy”,train\_data\_y)  #测试数据  Test\_data\_x=np.zeros([5000,7,7,512]).astype(“float”)  for i in range(5000);  img\_path=’.\猫狗分类器data\\reshape\_test\%d.jpg’%(i)  img=image.load\_img(img\_path,target\_size=(244,244))  x=image.img\_to\_array(img)  x=np.expand\_dims(x,axis=0)  x=preprocess\_input(x)  features=model.predict(x)  test\_data\_x[i,::]=features  print(i)  np.save(“./猫狗分类器data/test\_data\_x.npy”,test\_data\_x) |

**2.3建立模型，并训练**

在训练之前先读取数据，并且将7\*7\*512的数组平摊成25088长度的向量。

由于在原版的vgg网络中，卷积层后面也都是全连接层，这里的模型也都使用全连接层。首先将25088->2048->512->64，每一步都是全连接层+dropout+relu。

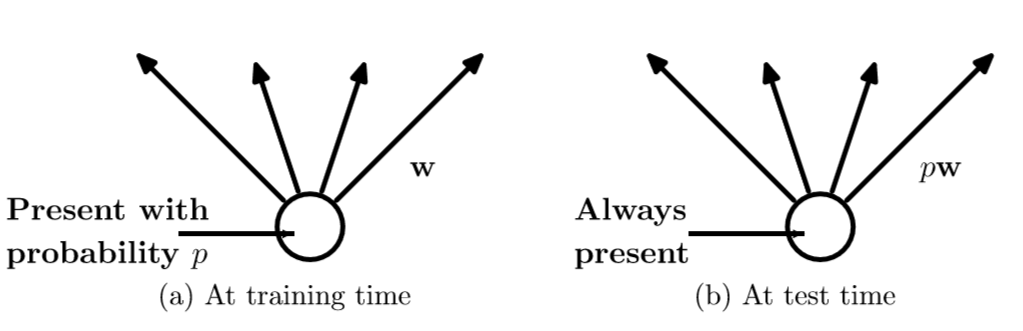
Vgg网络说一种CNN，Vgg非常深，通常有16-19层，卷积核大小为 3 x 3，16和19层的区别主要在于后面三个卷积部分卷积层的数量，模型结构如下：

  
可以看到VGG的前几层为卷积和maxpool的交替，每个卷积包含多个卷积层，后面紧跟三个全连接层。激活函数采用Relu，训练采用了dropout。

2.3.1.dropout简单介绍

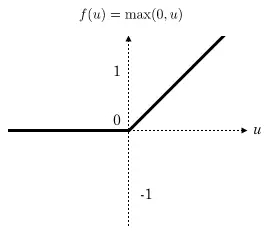
Dropout是一种在深度学习环境中应用的正规化手段。它是这样运作的：在一次循环中我们先随机选择神经层中的一些单元并将其临时隐藏，然后再进行该次循环中神经网络的训练和优化过程。在下一次循环中，我们又将隐藏另外一些神经元，如此直至训练结束。

在训练时，每个神经单元以概率p被保留(dropout丢弃率为1-p)；在测试阶段，每个神经单元都是存在的，权重参数w要乘以p，成为：pw。测试时需要乘上p的原因：考虑第一隐藏层的一个神经元在dropout之前的输出是x，那么dropout之后的期望值是E=px+(1−p)0，在测试时该神经元总是激活，为了保持同样的输出期望值并使下一层也得到同样的结果，需要调整*x*→*px。其中p是Bernoulli分布（0-1分布）中值为1的概率。示意图如下：*

  
2.3.2 激活函数Relu

最近几年卷积神经网络中，激活函数往往不选择sigmoid或tanh函数，而是选择relu函数。Relu函数的定义是：

Rulu函数图像如下：



与sigmoid作为激活函数相比，Relu函数其实就是一个max（0，x），计算代价小很多，所以表现为速度更快。并且能减轻梯度消失问题，计算梯度的公式为，其中，为sigmoid函数导数，在使用反响传播算法进行梯度计算时，每经过一层sigmoid神经元，梯度都要乘以一个。因此会导致梯度越来越小。这对于深层网络的训练是个很大的问题，而relu函数的导数为1，不会导致梯度变小。虽然导致梯度减小的原因并非全部来自激活函数，但relu对比sigmoid还是更好。

模型的定义代码如下：

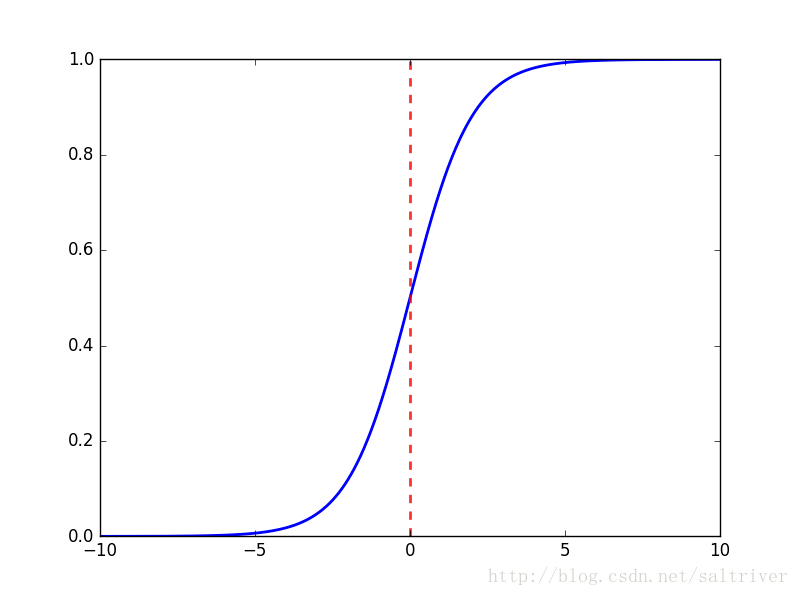
|  |
| --- |
| #定义maxpool  def maxPoolLayer(x, kHeight, kWidth, strideX, strideY, name, padding = "SAME"):  """max-pooling"""  return tf.nn.max\_pool(x, ksize = [1, kHeight, kWidth, 1],  strides = [1, strideX, strideY, 1], padding = padding, name = name)  #定义dropout层  def dropout(x, keepPro, name = None):  """dropout"""  return tf.nn.dropout(x, keepPro, name)  #定义全连接层  def fcLayer(x, inputD, outputD, reluFlag, name):  """fully-connect"""  with tf.variable\_scope(name) as scope:  w = tf.get\_variable("w", shape = [inputD, outputD], dtype = "float")  b = tf.get\_variable("b", [outputD], dtype = "float")  out = tf.nn.xw\_plus\_b(x, w, b, name = scope.name)  if reluFlag:  return tf.nn.relu(out)  else:  return out  #定义卷积层  def convLayer(x, kHeight, kWidth, strideX, strideY,  featureNum, name, padding = "SAME"):  """convlutional"""  channel = int(x.get\_shape()[-1]) #获取channel数  with tf.variable\_scope(name) as scope:  w = tf.get\_variable("w", shape = [kHeight, kWidth, channel, featureNum])  b = tf.get\_variable("b", shape = [featureNum])  featureMap = tf.nn.conv2d(x, w, strides = [1, strideY, strideX, 1], padding = padding)  out = tf.nn.bias\_add(featureMap, b)  return tf.nn.relu(tf.reshape(out, featureMap.get\_shape().as\_list()), name = scope.name) |

在进行具体的识别问题时，还需要定义VGG19。

在“64”层后面利用sigmoid转换成0-1之间的数据

Sigmoid函数是一个有着优美S形曲线的数学函数，在逻辑回归、人工神经网络中有着广泛的应用，其函数表达式为：

函数图像如下：



可以看出，sigmoid函数连续，光滑，严格单调，以(0,0.5)中心对称，是一个非常良好的阈值函数。当x趋近负无穷时，y趋近于0；趋近于正无穷时，y趋近于1；x=0时，y=0.5。当然，在x超出[-6,6]的范围后，函数值基本上没有变化，值非常接近，在应用中一般不考虑。Sigmoid函数的值域范围限制在(0,1)之间。这样，猫狗的分类问题就可以表示为0与1的概率问题，越靠近0，表示预测猫的可能性更大，反之越靠近1，表示预测狗的可能性更大。

本节python内容包括读取数据与构建模型两部分，具体代码如下：

|  |
| --- |
| #读取数据  train\_data\_x=np.load(“./猫狗分类器data/train\_data\_x.npy”)  train\_data\_y=np.load(“./猫狗分类器data/train\_data\_y.npy”)  test\_data\_x=np.load(“./猫狗分类器data/test\_data\_x.npy”)  print(train\_data\_x.shape,train\_data\_y.shape,test\_data\_x.sshape)  #(20000,7,7,512)(20000,)(5000,7,7,512)  train\_data\_x=np.reshape(train\_data\_x,[train\_data\_x.shape[0],-1])  test\_data\_x=np.reshape(test\_data\_x,[test\_data\_x.shape[0],-1)  print(train\_data\_x.shape,train\_data\_y.shape,test\_data\_x.shape)  #(20000,25000)(20000,)(5000,25088)  #构建模型  model=Sequential()  model.add(Dense(2048,activation=”relu”,input\_shape=(25088,)))  model.add(dropout(0,5))  model.add(Dense(512,activation=”relu”))  model.add(Dropout(0.5))  model.add(Dense(64))  model.add(Dense(1,activation=”sigmoid”))  model.summary()  model.compile(loss=’binary\_crossentropy’,optimizer=’adadelta’,metrics=[‘accuracy’])  Model.fit(x=train\_data\_x,y=train\_data\_y,batch\_size=2048,nb\_epoch=15,verbose=1) |

**2.4预测，提交数据**

用训练好的模型预测猫狗，并写到submission.csv里面。具体代码如下：

|  |
| --- |
| # 预测答案  test\_y = model.predict\_classes(test\_data\_x)  answer = pd.read\_csv(open("./猫狗分类器data/sampleSubmission.csv")) answer["label"] = test\_y  answer.to\_csv("./猫狗分类器data/submission.csv",index=False) # 不要保存引索列 |

**三、小组分工**

程序设计及编写：黄靖

程序调试：黄靖

实验报告：黄靖

**四、作业总结**

由于这是第一次接触到深度学习的题目，并且对使用到的工具python接触到较晚，不能熟练掌握其使用。并且对于整个学习过程以及最后的仿真过程均借助了别人的代码。在次基础上，做出了自己的一些理解与改变。本文使用的CNN训练数据，并且采用了keras训练好的VGG19模型，这都是读者之前未曾接触的或接触很少的内容。

通过查阅资料理解到，如果条件允许，可以使用aws上的EC2，结合keras框架，对数据集采用ImageNet上模型如VGG16/ResNet50/Inception/Xception进行训练。当然本文使用的VGG19模型。

这次的大作业让增加了我对模式识别的兴趣，同时原来神秘的推荐算法变得立体清晰。感谢赵老师教学帮助，并且在作业过程中给了我们大家很多指导。