MLND 机器学习毕业项目

———猫狗识别

苟永亮

2017/11/10

目录

1.问题的定义

1.1 项目概述：

计算机视觉旨在通过设计算法来让计算机自动理解图像的内容，脱胎于人工智能和认知神经科学。得益于最近 deep learning 的发展，基于卷积神经网络（cnn）的各种模型，如 ResNet，Xception 等，在 Imagenet 比赛上很好的解决了图片分类的问题，显示了 deep learning 在计算机视觉上的广阔前景。此次机器学习毕业项目，我选择了猫狗大战，一是因为此项目可是说是很好的入门级项目，有助于深入理解计算机视觉领域的基本问题；二是因为我刚在 kaggle 上入门也是用此项目，一举两得，项目知名度也高，容易被人理解。此次的数据全部都是从 kaggle 上下载而来。 https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

1.2 问题陈述：

Kaggle 竞赛提供的数据包含了一个训练集和一个测试集，训练集有 25000 张图片，猫狗各一半，测试集有 12500 张图片。这些从真实世界中采集而来的猫狗图片，图像分辨率不同，背景环境复杂，猫狗的品种繁多，这些都为我们的分类增加了难度。我们需要搭建模型，用训练集里面得 25000 张图片来训练模型，最后再使用测试集来验证我们的模型，看看对于猫狗类型的判断准确率。

1.3 评价指标

所有的数据集都是从 kaggle 下载的，kaggle 官方的评估标准是 LogLoss，

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

这是一个二分类问题的通用评价指标，

n 代表测试集中的图片数量，

$\hat{y}_i$ 为测试图片是狗的概率，

y_i 如果图片是狗为 1，是猫为 0。

2 分析

2.1 数据的探索

Kaggle 猫狗数据集中，分为两个文件夹，train 和 test，train 共有 25000 张图片，猫狗各一半，test 文件夹共有 12500 张图片，需要我们去标定类别。

2.2 算法和技术

根据分析可知，猫狗识别本是上是一个二分类问题。适用于分类的机

器学习算法很多，比如支持向量机 SVM，随机森林 RF, 决策树 DT，然而这些方法在注明数据及上 Imagenet 数据集上取得的效果一直不是很好，并不能有效的解决图像识别问题，人们急需要一种新的方法来克服这一计算机视觉难题。直到 2013 年 Alex 发明 Alexnet 深度学习神经网络模型取得 Imagenet 比赛第一名，图像识别问题才算解决了。深度学习其实并不是一个全新的概念，早在上世纪 40 年代，就已经提出，但是由于当初的计算级的计算能力小，数据集稀缺，深度学习一度处于低谷。近年来，得益于 GPU 运算能力的爆发，数据集的增多，以及有效算法的开发，深度学习这一技术大放异彩，解决了许多问题，特别是计算机识别领域，使用卷积神经网络（cnn），一种专门用来处理具有类似网格结构的数据的神经网络，可以说基本解决了图像识别的问题。所以在此次项目中，我也采用 CNN 来搭建我的模型框架。

另外，各大机构公司也推出了很多的深度学习平台， 比如 Google 的 tensorflow， Facebook 的 pytorch， 贾杨清开发的 Caffe。本项目中，本着简单快速上手的原则，我使用基于 Google tensorflow 作为 backend 的 keras 来搭建深度学习神经网络模型。

2.3 基准模型

对于图像识别的研究已经非常多了，目前使用深度神经网络来搭建模型，分类图片是业内的趋势。所以此次我也是用深度卷积神经网络来搭框架。为了追求快的运算时间和高效的识别准确率，我计划使用别

人已经训练好的模型，直接导出特征向量，用于我的模型中，而经过多次的图像识别竞赛，著名的模型有考虑残差的 ResNet 模型，VGG 模型，Xception 模型，Inception 模型，我将综合使用这些模型。

3 方法

3.1 数据预处理

首选我们需要新建一个文件夹 train2，包含两个子文件夹 dog 和 cat，将 train 文件夹中的图片，按类别放入 dog 和 cat 文件夹中，方便之后的训练。

3.2 实现

卷积神经网络的核心是对图像特征的提取表示，从最基本的边缘特征逐级提升到抽象层次的对象，CNN 通过各卷积层及其相应的权重来记录这些特征。 从零开始训练一个卷积神经网络，需要非常精心的网络设计，大量的参数优化，和长时间的运算，很耗费资源。其实经过长久的发展，现在已经有很多优秀的网络结构模型可以利用，这样既能很好的表征猫狗的特征，也能节约计算资源。所以如之前所说，我直接应用了 4 个已经预训练好的模型，ResNet50，VGG19，InceptionV3，Xception。代码中只需要载入这些特征向量，并且将它们合并成一条特征向量，这样节得到了我们的 X_train，X_test 和 y_train。代码如图所示：

```
for filename in ["gap_ResNet50.h5", "gap_VGG19.h5",
```

```
"gap_InceptionV3.h5",'gap_Xception.h5']:
    with h5py.File(filename, 'r') as h:
        X_train.append(np.array(h['train']))
        X_test.append(np.array(h['test']))
        y_train = np.array(h['label'])

X_train = np.concatenate(X_train, axis=1)
X_test = np.concatenate(X_test, axis=1)
X_train, y_train = shuffle(X_train, y_train)
```

最后构建模型，一层简单的全连接层。，其中比较重要的一个参数是dropout。代码如图所示：

```
input_tensor = Input(X_train.shape[1:])

x = Dropout(0.25)(input_tensor)

x = Dense(1, activation='sigmoid')(x)

model = Model(input_tensor, x)

model.compile(optimizer='adadelta',loss='binary_crossentropy',

                    metrics=['accuracy'])
```

4 结果

4.1 模型的评价与验证

然后就可以训练模型了：

```
In [13]: model.fit(X_train, y_train, batch_size=128, epochs=10, validation_split=0.2)

Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 1s - loss: 0.0149 - acc: 0.9951 - val_loss: 0.0102 - val_acc: 0.9958
Epoch 2/10
20000/20000 [==============================] - 1s - loss: 0.0151 - acc: 0.9953 - val_loss: 0.0100 - val_acc: 0.9964
Epoch 3/10
20000/20000 [==============================] - 1s - loss: 0.0132 - acc: 0.9959 - val_loss: 0.0098 - val_acc: 0.9960
Epoch 4/10
20000/20000 [==============================] - 1s - loss: 0.0128 - acc: 0.9958 - val_loss: 0.0104 - val_acc: 0.9960
Epoch 5/10
20000/20000 [==============================] - 1s - loss: 0.0125 - acc: 0.9957 - val_loss: 0.0099 - val_acc: 0.9962
Epoch 6/10
20000/20000 [==============================] - 1s - loss: 0.0114 - acc: 0.9964 - val_loss: 0.0099 - val_acc: 0.9962
Epoch 7/10
20000/20000 [==============================] - 1s - loss: 0.0121 - acc: 0.9963 - val_loss: 0.0101 - val_acc: 0.9962
Epoch 8/10
20000/20000 [==============================] - 1s - loss: 0.0105 - acc: 0.9968 - val_loss: 0.0102 - val_acc: 0.9962
Epoch 9/10
20000/20000 [==============================] - 1s - loss: 0.0114 - acc: 0.9961 - val_loss: 0.0099 - val_acc: 0.9960
Epoch 10/10
20000/20000 [==============================] - 4s - loss: 0.0120 - acc: 0.9961 - val_loss: 0.0100 - val_acc: 0.9962

Out[13]: <keras.callbacks.History at 0x116a8240>
```

可以看到，经过 10 epochs 的训练，loss 已经降到 0.012 了，识别准确率到达 0.9961，已经很不错了。

模型训练完了之后，我们使用测试集来验证，将结果输出为 pred_csv 文件，就可以提交到 Kaggle 上面，观察我们最后的得分了，0.04172，还不错的分数，大概能排在 20 名左右。

4.2 合理性分析

最后可以看到，经过结合四种不同模型，合并它们的特征向量，我得到较好的识别结果，其效果大于单独使用其中的一种模型。因为这里的每一个模型都是久经检验的，可以说更有所长，综合它们到一起，可以高度概括出图片当中的内容，所以最后得出的结果较好。

5 结论

5.1 思考

整个工作的流程可以概括为：数据预处理，提取特征向量或者使用预训练的特征向量，综合不同的模型载入特征向量，构建模型并训练，

测试集预测最终结果。决定最后结果的关键点在于第二步，如何提取特征向量，我直接使用了已训练的，久经考验的几个模型，综合使用它们，这样可以节约我自己的训练时间，且可得到较好的结果。我还考虑过这几种模型不同的组合使用，发现四种全用时，效果最好。我还考虑过 dropout 数值对于结果的影响。我们知道 dropout 技巧实际上是一种模型平均，就是把来自不同模型的估计或者说预测通过一定的权重平均起来。每次以一定的概率忽略一些隐层节点，这样每次训练的网络就是不一样的，每次训练都可以看做一个新的模型；此外，隐层节点以一定的概率出现，不能保证哪 2 个节点同时出现，这样就阻止了某些特征关联与其他特征的情况。最后发现，dropout 在 0.5 的时候，效果最好，具体的原因还不太清楚。总的来说，目前的解雇哦还是不错的，高效的识别率，解决了猫狗识别问题。

5.2 改进

想要进一步提高识别率，似乎就需要深入钻研一下神经网络基本结构，不断练习调节参数，积累对于调参的心得。众所周知，深度神经网络的参数非常之多，而且内部具体是如何工作的，目前还是一个黑箱状态，人们了解的并不够透彻，所以不同参数之间对结果有着不同的影响，这个需要更多的时间和经验来掌握。

6 致谢

在这里我要特别感谢一下杨培文大神，他视频和博客里对于猫狗识别

项目的仔细讲解为我入手此项目提供了莫大的帮助，更感谢他在微信群里每天对我们的指导。也要感谢优达学城提供的各种资源支持，是优达学城带我进入了机器学习的大门，其优秀的教学模式，精心设计的教学内容，使人影响深刻。祝愿优达越办越好！