基于深度学习的数字图像识别技术设计

物理学基地班　　王意昂　　2015301020069

摘要：　本文介绍了图像识别和深度学习基本原理和流程，深入分析了 softmax 回归算法，设计了数字图像的识别模型，在 TensorFlow 上使用 softmax 回归算法实现了数字图像的识别。实验表明，使用 TensorFlow 能够快速实现图像识别、自然语言处理等相关的机器学习和深度学习算法，可靠性和准确性较高。

关键词：　数字图像识别；深度学习；TensorFlow；softmax 回归

1 引言

1.1 图像识别技术

图像识别是模式识别技术的一种，特指对图像类模式的识别。图像识别过程主要包括训练和测试两个阶段。训练阶段对图像特征进行训练得到分类模型，测试阶段可以利用已训练模型得到识别结果。以上两个阶段主要由图像预处理、特征提取及分类三个环节组成。

在图像识别中，首先要对获取的图像进行预处理，目的是去掉图像中的噪声，把它变成一幅清晰的点阵图，以便于提取正确的图像特征。预处理主要对图像进行几何变换、格式变换、图像滤波，得到满足需要的改进形式。其次要对图像中的目标进行分析，提取正确代表不同目标物特点的特征参数（图像特征）。对图像进行特征提取后，具备将一个目标或场景的不同影像进行匹配。最后要对图像中目标物进行识别和解释。设计分类器，建立分类模型，对图像中目标物进行识别和分类。

机器学习作为图像识别实现的主要方法，其实现方法、工具、性能仍难以满足当前需要，应用时也存在一定的困难，在信息化，智能化的时代背景下，本文主要利用 Google 第二代人工智能平台 TensorFlow，使用 softmax 回归算法实现数字图像的识别。

1.2 机器学习及实现流程

机器学习能够通过先验学习获取经验，自动分析数据获取规律，重新组织已有知识、结构，并不断完善自身性能，让计算机在无事先明确编程的情况下做出正确的响应。机器学习主要从已知数据学习获取正确经验或规律，然后利用该经验或规律推理其中未知的、潜在的概率分布等重要信息，提示数据样本中变量（特征）之间的关系。

机器学习主要有两个方向：一是监督学习，它主要解决分类和回归问题。分类主要任务是将实例数据划分到合适的分类中，而回归主要用于预测数值型数据；二是无监督学习，它是将数据集分成由类似对象组成的多个类的过程，即聚类；或者将寻找描述数据统计值的过程，即密度估计。在构建机器学习模型选择算法，首先要明确机器学习算法的目的。如果想要预测，则选择监督学习算法，再考虑目标变量类型，如果预测值是离散的，则选择分类算法，如果预测值为连续的，则选择回归算法。如果不想预测，则选择无监督学习，如果需要将数据划分为离散的组，则使用聚类算法；然后要分析数据集。由于机器学习的同类算法很多，为了缩小算法的选择范围，需要充分了解数据，对实际数据了解越充分，才能选出最切近该问题的算法。

机器学习应用程序开发过程中，首先需要收集数据。如制作网络爬虫从网站爬取数据等；接着要格式化数据源，确保数据格式符合要求；然后将格式化的数据输入到算法，即训

练算法，从中抽取信息；算法训练好后，测试算法的工作效果，检验算法的成功率；若不满意可修改前面步骤，最后将机器程序转换为应用程序，执行实际任务。

1.3 TensorFlow

2015 年 9 月，Google 发布了其第二代人工智能系统——TensorFlow， 一个开源机器学习软件资源库。它是目前最受欢迎的机器学习算法框架，支持多种开发语言（本文使用的为 python），支持异构分布式系统部署。TensorFlow 即支持深度学习算法，也实现了很多其他算法，例如线性回归、逻辑回归等。

2 数字图像识别模型的设计与实现

2.1 数据采集

本模型主要用 MNIST 数据集进行训练，而验证和测试部分采用不同来源的数字图片，从而验证正确率。MNIST 数据集在 TensorFlow 官网上下载。MNIST 数据集由训练集、验证集和测试集三部分构成，其中训练集有 55000 个样本，验证集有 5000 个样本，测试集有 10000 个样本。每一个样本都有它对应的标签信息(label)用来描述样本中的数字。MNIST 样本描述的是一个28×28像素的灰度图片，可用一个长度为 784 的数组来表示一张图片。图片样本虽然丢弃了图片二位结构的信息，但由于分类任务比较简单，这种数据简化，可以选择比较简单的分类算法实现模型。这里我们对官方程序稍作改进：训练样本仍用 MNIST，而验证测试部分采用处理后的任意数字图片。

训练样本的数据特征是一个55000×784的 Tensor。第一维是图片编号，第二维是对应图片的像素点,其值表示某个像素点的灰度值。训练样本的标签是一个55000×10的 Tensor，用来描述每个样本代表的数字，每个样本的 label 是一个 10 维向量，只有一个值为 1，其余为 0；那个数字样本图片的 label 变量对应位置为 1，其余为 0。例如数字 0 的样本 label 就是[1,0,0,0,0,0,0,0,0,0]。

由于 MNIST 数据集所有样本描述的数字图片值为 0—9，所有样本就分为 10 类，即 0,1,2,3,4,5,6,7,8,9，数字图像识别问题就是计算测试集中第 i 个样本是数字 j（j=0,1,2…9）的概率问题，可以采用 softmax 回归算法实现。

2.2 softmax 回归算法

Softmax 回归将只能解决二分类问题的 Logistic 回归扩展至能够解决多分类问题。通过梯度下降算法计算参数，得到训练好的 softmax 回归模型。然后将测试集中的样本数据代入假设函数计算其所属类别的概率，选择概率最大的类别作为预测结果。

3 TensorFlow 实现数字图像识别

3.1 MNIST 简介

MNIST 数据库是一个手写数字的数据库，它提供了六万的训练集和一万的测试集。它的图片是被规范处理过的，是一张被放在中间部位的$28\text{px} \times 28\text{px}$的灰度图。总共 4 个文件：

train-images-idx3-ubyte: training set images

train-labels-idx1-ubyte: training set labels

t10k-images-idx3-ubyte: test set images

t10k-labels-idx1-ubyte: test set labels

图片都被转成二进制放到了文件里面，所以，每一个文件头部几个字节都记录着这些图片的信息，然后才是储存的图片信息。

3.2 准备要识别的图片

按照 mnist 的标准需要格式为28×28的灰度图片，因此需要将采集来的图片进行转化。首先需要将任何一张带有数字的图片，按照长宽 1:1 的比例，以单个数字为中心进行裁剪，注意需要保留一个完整的数字；将图片文件按照规则重命名，比如一张为 3 的图片应该命名为 3_4（最后一位可以随意命名，为了区分不同版本的图片，示例中最后一位表示版本号）然后将第一步处理后的所有图片放入目录‘C:\TensorFlow_train\input’中，运行程序 mnist_train.py ，该程序可以将这些图片自动将该目录中的正方形图片转化为28×28的灰度图片，以供测试阶段使用。程序 mnist_train.py 如下：

```
# 将input目录中的图片经缩小并转化为灰度图片后再存入identify目录
import os
from PIL import Image

def get_path(path):
    return [os.path.join(path, f)for f in os.listdir(path)]

for path in get_path(r'C:\TensorFlow_train\input'):
    outfile = 'C:\TensorFlow_train\identify\ '+ os.path.basename(path)
    print(outfile)
    # 写入图片，并将其转化为灰度图
    im = Image.open(path).convert('L')
    # 缩小图片尺寸到28*28
    im.thumbnail((28, 28))
    im.show()
    im.save(outfile)
```

3.3 矩阵转化

将28×28图片转化成28×28矩阵，再将矩阵转化为 1 维矩阵（即把第 2,3,4,5....行矩阵依次接入到第一行的后面）通过子程序来完成。

3.4 样本标签

在前文 2.1 中，已经初步介绍了样本的 label。Softmax 回归处理后会生成一个1×10的数组，该数组表示的就是这张图片是哪个数字的概率（已经归一化处理过）因此实际上概率最大的那个数字就是我们预测的值。标准标签就是表示图片对应数字的概率是 100%，而别的数字为 0。

3.5 softmax 回归

Softmax 回归：为了得到一张给定图片属于某个特定数字类的证据（evidence），我们对图片像素值进行加权求和。如果这个像素具有很强的证据说明这张图片不属于该类，那么相应的权值为负数，相反如果这个像素拥有有利的证据支持这张图片属于这个类，那么权值是正数。同时，需要引入一个额外的偏移量（bias），因为输入往往会带有一些无关的干扰量。对于给定输入图片 x 它代表的是数字 i 的证据可以表示为

$$evidence_i = \sum_j w_{i,j}\, x_j + b_i$$

其中，$w_{i,j}$表示权重，b_i代表数字 i 类的偏移量，j 代表给定图片 x 的像素索引用于像素求和。然后用 softmax 函数可以把这些证据转换成概率 y：

$$Y = \text{softmax(evidence)}$$

这里的 softmax 可以看成是一个激励函数或链接函数，把我们定义的线性函数的输出转换成我们想要的格式，也就是关于 10 个数字类的概率分布。因此，给定一张图片，它对于每一个数字的吻合度可以被 softmax 函数转换成为一个概率值。softmax 函数可以定义为：

$$\text{Softmax(x) = normalize(exp(x))}$$

展开等式右边的子式，可以得到：

$$Softmax(x)_i = \frac{\exp(x_i)}{\sum_j exp(x_j)}$$

但是更多的时候把 softmax 模型函数定义为前一种形式：把输入值当成幂指数求值，再正则化这些结果值。这个幂运算表示，更大的证据对应更大的假设模型（hypothesis）里面的乘数权重值。反之，拥有更少的证据意味着在假设模型里面拥有更小的乘数系数。假设模型里的权值不可以是 0 值或者负值。Softmax 然后会正则化这些权重值，使它们的总和等于 1，以此构造一个有效的概率分布。

具体见官网：http://tensorfly.cn/tfdoc/tutorials/mnist_beginners.html

3.6 图片识别

训练结束后，即可使用这个模型进行图片识别了。最后会输出一个 input 值和一个 output 值，其中 input 来自于图片名第一位，即准确值；output 为使用训练模型识别出的数字。由于 mnist_softmax 程序读取图片是按照编号依次读取的，比较即可看出图片识别正确与否。

4 结果与讨论

4.1 模型评估

tf.argmax 函数可以用于得到某个 tensor 对象在某一维上的其数据最大值所在的索引值。由于标签向量是由 0、1 组成，因此最大值 1 所在的索引位置就是类别标签，比如 tf.argmax(y,1) 返回的是模型对于任一输入 x 预测到的标签值，而 tf.argmax(y_,1) 代表正确的标签，我们可以用 tf.equal 来检测我们的预测是否真实标签匹配(索引位置一样表示匹配)。

这里我们测试两种不同的算法实现的训练模型。第一种来源于官方教程介绍的最简单的模型（MNIST For ML Beginners），下一部分测试代码也由此改编而来；

下面是运行结果：

```
Extracting /tmp/tensorflow/mnist/input_data\train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\t10k-labels-idx1-ubyte.gz
2017-12-31 16:23:41.042956: I C:\tf_jenkins\home\workspace\rel-win\M\wind
0.9172

Process finished with exit code 0
```

前面四行表示写入压缩文件；后面的值 0.9172 表示识别正确率为 91.72%，这个结果不太让人满意。

第二种是多层卷积网络模型。细节内容就不一一赘述了，详情见官网 运行结果如下：

刚开始循环时：

```
Extracting MNIST_data\train-images-idx3-ubyte.gz
Extracting MNIST_data\train-labels-idx1-ubyte.gz
Extracting MNIST_data\t10k-images-idx3-ubyte.gz
Extracting MNIST_data\t10k-labels-idx1-ubyte.gz
2017-12-31 20:32:45.449389: I C:\tf_jenkins\home\
WARNING:tensorflow:From C:\Users\wangya\PycharmPr
Instructions for updating:
Use `tf.global_variables_initializer` instead.
step 0, training accuracy 0.04
step 100, training accuracy 0.92
step 200, training accuracy 0.94
step 300, training accuracy 0.94
step 400, training accuracy 0.9
step 500, training accuracy 0.98
step 600, training accuracy 0.92
step 700, training accuracy 0.98
step 800, training accuracy 0.92
```

可以看出，第一次正确率非常之低，只有 4%；经过 100 次循环后迅速上升到 92%，已经超过了第一种简单的模型。随后在 0.9 附近涨落，总体而言正确率在上升。

一段时间后：

```
step 5300,  training accuracy 1
step 5400,  training accuracy 0.98
step 5500,  training accuracy 0.96
step 5600,  training accuracy 1
step 5700,  training accuracy 0.94
step 5800,  training accuracy 0.96
step 5900,  training accuracy 1
step 6000,  training accuracy 1
step 6100,  training accuracy 1
step 6200,  training accuracy 1
step 6300,  training accuracy 0.98

Process finished with exit code 1
```

从图中可以看出，正确率和开始相比有了明显的进步，很多情况下都到了 1（当然，由于精度有限，所以这个 1 不一定是 100%）但也有不小的涨落。显然这种模型是非常有效的，远远优于第一种了。

4.2 对数字图像的识别

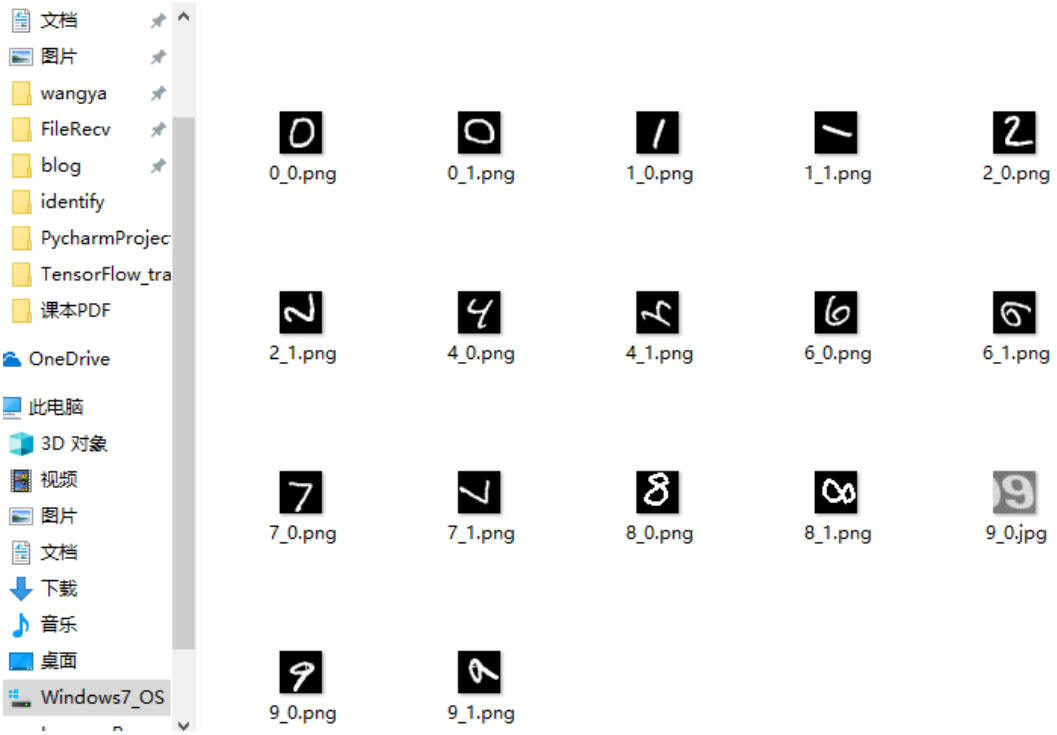
首先，将数字图片按照 3.2 所述制成标准格式。然后运行 mnist_softmax 文件，部分运行结果如下：

```
input: 0   input: 1   input: 4   input: 6   input: 8
output: 0  output: 4  output: 4  output: 0  output: 7


input: 0   input: 2   input: 4   input: 7   input: 9
output: 0  output: 2  output: 3  output: 7  output: 5


input: 1   input: 2   input: 6   input: 7   input: 9
output: 1  output: 4  output: 6  output: 4  output: 9
```

图中，input 来自于图片名第一位，即准确值；output 为使用训练模型识别出的数字。比较即可看出图片识别正确与否。图片顺序是从上到下，为了节省空间将一张竖直图片截成 5

张从左到右依次排列。

从图上看，似乎正确率并不高——明显不到 90%。不过，原因大概是测试过于苛刻。



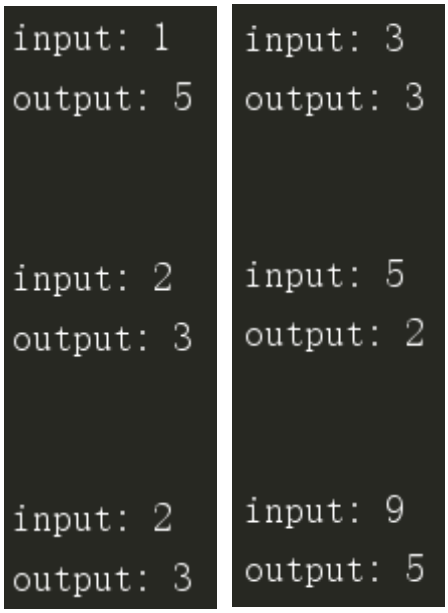
其中，以 0 结尾的图片，是从原来的测试集中得出来的。

从识别结果来看[1]，训练模型对于测试集中的图片识别做到了绝对的正确。同时，测试了旋转 90 度以后有没有可能识别。结果显示，部分倾斜的图片可能由于特征比较明显，也是可以识别的，比如 0。识别图片格式似乎也没有限制，既可以是 png 也可以是 jpg。

再来一个更加困难的测试。图片分别来源于网上和手机照片：

经过处理后用于识别，结果如下：

[1] 需要声明的是，第二部分数字图片的识别整个都是建立在第一部分简单模型的代码之上的。因为多层卷积模型训练时间比较长，修改代码比较复杂，因此还是用简单模型比较方便。



对于这种复杂数字图形的识别，这种模型已经力不从心了，仅有一个识别结果是正确的。

4.3 讨论

采用第一种模型比较简单，模型训练速度比较快，也基本实现了识别的目的。但是，也存在相当多可以改进的部分：对于图片预处理不够，有些还是需要手动操作；算法比较简单，正确率远远不够，而且遇到复杂图像更是无能为力。可以期待，诸如多层卷积网络、卷积神经网络、递归神经网络等更加有效的模型，必将使得识别更加有效。

遗憾的是，这项工作距离最终实用，还有相当远的距离。如何识别多个数字？怎样将一张含有数字的图片都识别出来？以目前的情况来看，距离这一步还是相当遥远的，可能需要构建一套全新的模型，可能已经远远超出了 TensorFlow 可以处理的范围了。因此，与其忧心于人工智能的飞速发展，不如先好好考虑如何改进，这个领域还有很多问题待解决。

5 补充材料

本程序已全部开源到 Github 平台：
https://github.com/youngwang-whu/computational_physics_N2015301020069/tree/master/TensorFlow%E5%85%A5%E9%97%A8

6 参考资料

[1] 吴忠，朱国龙，黄葛峰等.基于图像识别技术的手写数字识别方法[j].计算机技术与发展，2011,21(12):48-51.

[2] 章敏敏，徐和平，王晓洁等.谷歌 TensorFlow 机器学习框架及应用[J].微型机与应用，2017.36(10):58-60