Tensorflow安装教程及数字识别应用实例

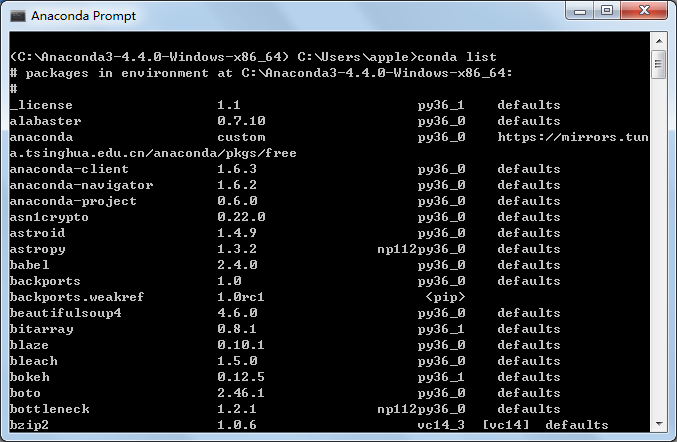
1. 安装Tensorflow

Tensorflow的使用需要python环境，推荐使用Anaconda安装，操作系统：windows

步骤如下：

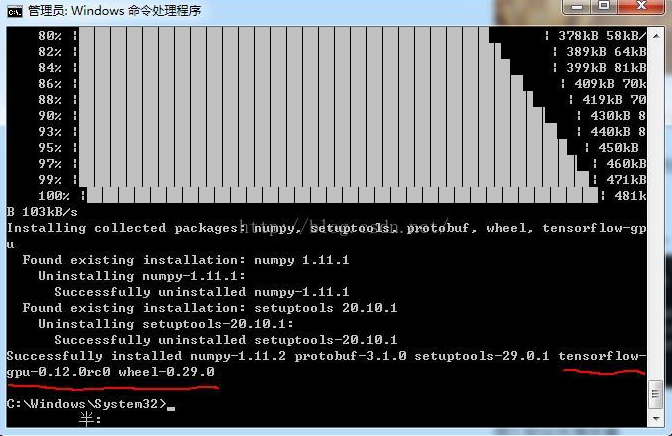
1. 下载Anaconda 官网：<https://www.anaconda.com/download/>，下载最新版本的即可（现在最新的版本是Anaconda 5.0.0for python3.6）；
2. 安装Anaconda，和安装普通的软件一样，全部选择默认即可，自己也可选择安装路径。注意勾选将python3.6添加进环境变量；

Anaconda安装好了以后，我们可以运行 开始菜单->搜索Anaconda Prompt，在命令行里输入：conda list，可见Anaconda已经安装了哪些包：

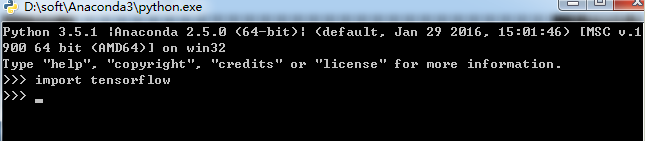


1. 安装Tensorflow

Tensorflow有cpu和gpu两个版本，这里我们安装cpu版本，在命令行键入pip install tensorflow进行安装。安装过程会持续几分钟，安装完成后出现以下信息：



1. 测试tensorflow是否安装成功



引入tensorflow没有报错，说明tensorflow安装成功，至此cpu版的tensorflow安装成功。

1. Tensorflow实现MNIST手写数字识别

Tensorflow基本概念：

➢ 使用图（graphs）来表示计算任务，图中的节点称之为op，

➢ 在被称之为会话（Session）的上下文（context）中执行图

➢ 使用Tensor表示数据，Tensor 看作是一个 n维的数组或列表。

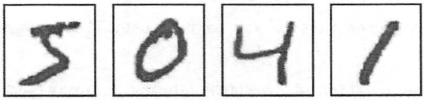
➢一个op获得0个或多个Tensor，执行计算，产生0个或多个Tensor。

➢ 通过变量（Variable）维护状态

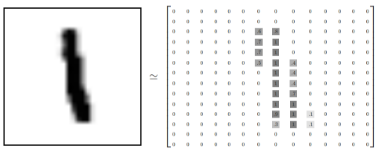
➢ 使用feed和fetch可以为任意的操作赋值或者从其中获取数据

MNIST数据集：

MNIST是一个入门级的计算机视觉数据集，它包含各种手写数字图片，我们的任务就是用一个神经网络训练这些数据集从而对这些手写数字的图片进行分类：



我们的图像是28像素X 28像素大小的灰度图片，我们可以用一个数字数组来表示这张图片：



我们把这个数组展开成一个向量，长度是 28x28 = 784。如何展开这个数组（数字间的顺序）不重要，只要保持各个图片采用相同的方式展开。从这个角度来看，MNIST数据集的图片就是在784维向量空间里面的点, 并且拥有比较[复杂的结构](http://colah.github.io/posts/2014-10-Visualizing-MNIST/)。

展平图片的数字数组会丢失图片的二维结构信息。这显然是不理想的，但由于这个数据集的分类任务比较简单，为了简化问题，因此不需要建立一个太复杂的模型。

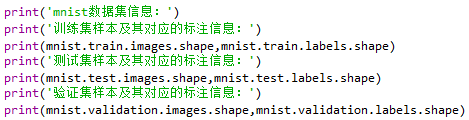
下面我们开始进行对数字的识别和分类，首先对MNIST数据进行加载：

import input\_data

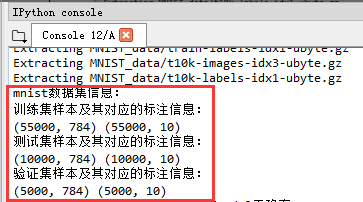
mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)

下载下来的数据集被分成两部分：60000行的训练数据集（mnist.train）和10000行的测试数据集（mnist.test）其中训练集中又划分出5000个样本作为验证集。这样的切分很重要，我们在训练集上训练模型，在验证集上检验效果并决定何时完成训练，最后在测试集上评测模型的效果，从而更加容易把设计的模型推广到其他数据集上（泛化）。

我们可输入如下代码查看这个MNIST数据集的情况：



输出结果如下：



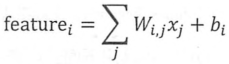
可以看到，训练集的shape是一个55000 X 784 的Tensor，第一个维度是图片的编号，用来索引图片，第二个维度是图片中像素点的编号，用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于0和1之间。同样也可以看到测试集和验证集Tensor的shape及其样本的个数。

每一个样本都有它对应的标注信息，即label。MNIST数据集的label是介于0到9的数字，用来描述给定图片里表示的数字，因此可以看到训练集、测试集和样本集label的第二个维度都是10。这里对10个种类进行了one-hot编码，label是一个10维的向量，一个one-hot向量除了某一位数字是1以外，其余维度数字都是0，比如数字0对应的label就是([1,0,0,0,0,0,0,0,0,0,0])。

Softmax回归介绍：

准备好数据后，接下来就要设计算法了，这里使用Softmax回归算法训练手写数字识别的分类模型。当模型对一张图片进行预测时，softmax回归会对每一种类别估算一个概率，最后取概率最大的那个数字作为输出结果。它的工作原理简单地说就是将可以判定为某类的特征相加，然后将这些特征转化为判定是这一类的概率。上述特征通过对所有像素求一个加权和得到，而权重是模型根据数据自动学习、训练出来的。比如某个像素的灰度值大代表很可能是数字n时，这个像素的权重就很大。我们将这些特征写成如下公式：

i代表第i类，j代表一张图片的第j个像素，bi是bias，顾名思义就是这个数据本身的一些倾向，比如大部分数字都是0，那么0的特征对应的bias就会很大：



接下来就是对所有特征计算Softmax，先对各个类的特征求exp函数，然后对他们标准化，使得和为1，特征值越大的类，最后输出的概率也越大。softmax回归的实现用一行简洁的公式来表达即：y=Softmax( wx + b )

现在我们就使用Tensorflow实现一个softmax回归：

首先导入Tensorflow库，并创建一个新的InteractiveSession，然后创建一个占位符placeholder，即输入数据的地方，第一个参数是数据类型，第二个参数[None, 784]代表tensor的shape，即数据的尺寸，None代表不限条数的输入，784代表每条输入是一个784维的向量：

IMG_256

接下来给Softmax回归模型中的权重值weights和偏置量bias创建Variable对象，Variable是用来存储模型参数的，它可以长期存在并在每轮迭代中被更新。这里我们把weights和bias全部初始化为0，w的shape是[784, 10]，784是输入特征的维数，后面的10代表有10类：

IMG_256

然后用一行简单的代码定义Softmax回归算法：

IMG_256

Softmax是tf.nn下面的一个函数，tf.matmul是Tensorflow中的矩阵乘法函数。

为了训练我们的模型，在机器学习中，我们需要定义一个损失函数（loss）来描述模型对问题的分类精度，loss越小，说明模型越精确。对分类问题，通常使用“交叉熵”（cross-entropy）作为损失函数。

首先需要添加一个新的占位符用于输入正确值（label），然后计算交叉熵cross\_entropy：

IMG_256

现在采用经典的随机梯度下降算法，并根据反向传播算法进行训练，在每一轮迭代时更新参数来减小loss。Tensorflow给我们提供了一个封装好的优化器，只需每轮迭代时feed数据给它就行。

如下，直接调用tf.train.GradientDescentOptimizer，并设置学习率为0.5，优化目标为cross\_entropy，得到进行训练的操作train\_step：

IMG_256

下一步执行全局参数初始化：

IMG_256

最后，迭代的执行训练操作train\_step，这里训练1000次，使用批梯度下降，每次随机抓取训练数据中的100个批处理数据点，并feed给placeholder，运行train\_step对这些样本进行训练：

IMG_256

采用小部分数据进行随机梯度下降会比全样本训练的收敛速度快很多。

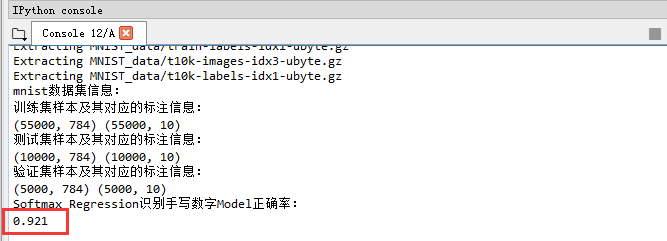
完成训练后，对模型进行评估，tf.argmax是从一个tensor中寻找最大值的序号，比如tf.argmax(y,1)返回的是模型对于任一输入x预测到的标签值，而tf.argmax(y\_,1)代表正确的标签，tf.equal用来检测我们的预测是否真实标签匹配：

IMG_256

最后计算准确率，运行测试，输出结果：

IMG_256

Softmax Regression识别手写数字Model正确率结果：



以上便是使用Tensorflow实现MNIST手写数字识别的一个简单的例子，这可以算作是一个没有隐含层的最浅的神经网络。使用Tensorflow进行算法设计、训练的核心步骤总结如下：

A.加载数据

B.定义输入placeholder、变量、输出

C.定义损失函数loss：

D.选定优化器优化loss

E.迭代地对数据进行训练

F.在测试集或验证集上对准确率进行评估