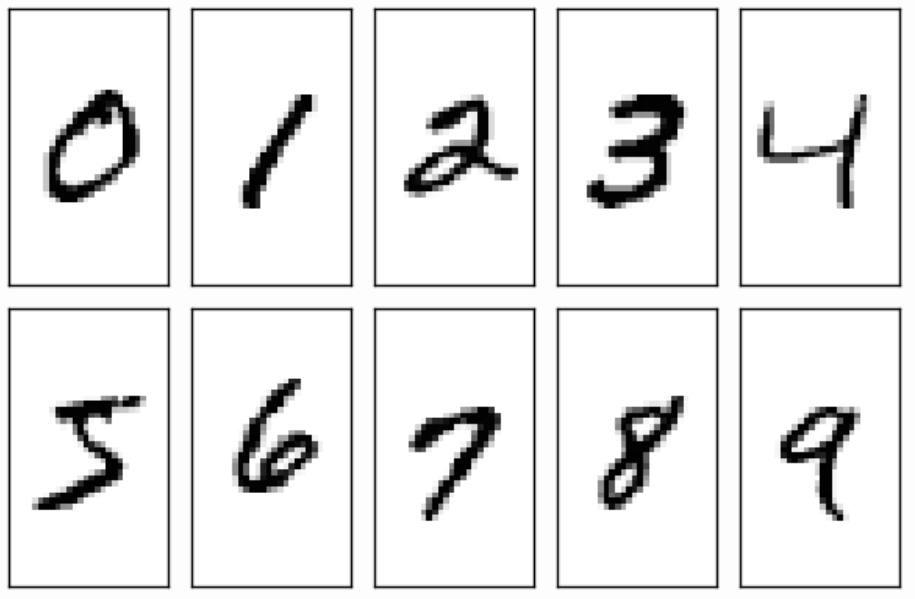
手写字体识别实验

1. 数据集 MNIST

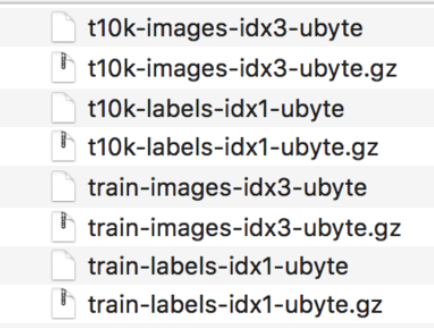


MNIST 数据集可在 http://yann.lecun.com/exdb/mnist/ 获取, 它包含了四个部分:

* Training set images: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)
* Training set labels: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)
* Test set images: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)
* Test set labels: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

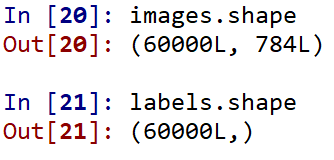
MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST). 训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据.

将数据集下载到 mnist 以后, 解压即可:

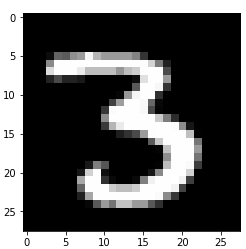


图片是以字节的形式进行存储, 我们需要把它们读取到 NumPy array 中, 以便训练和测试算法.

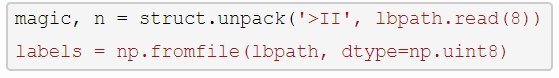
Process\_data.py 中load\_mnist 函数返回两个数组, 第一个是一个 n x m 维的 NumPy array(images), 这里的 n 是样本数(行数), m 是特征数(列数). 训练数据集包含 60,000 个样本, 测试数据集包含 10,000 样本. 在 MNIST 数据集中的每张图片由 28 x 28 个像素点构成, 每个像素点用一个灰度值表示. 在这里, 我们将 28 x 28 的像素展开为一个一维的行向量, 这些行向量就是图片数组里的行(每行 784 个值, 或者说每行就是代表了一张图片). load\_mnist 函数返回的第二个数组(labels) 包含了相应的目标变量, 也就是手写数字的类标签(整数 0-9).



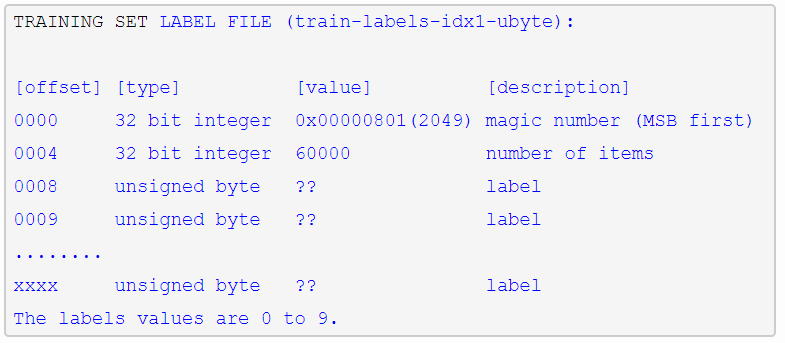
说明训练集中有60000幅图像,对应有60000个标签,每个图像拉成一个向量放在images矩阵中的一行. 我们取出images中的一行可视化一下:



解析:



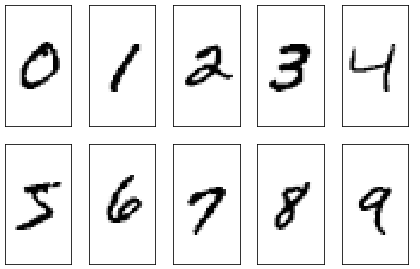
为了理解这两行代码, 我们先来看一下 MNIST 网站上对数据集的介绍:



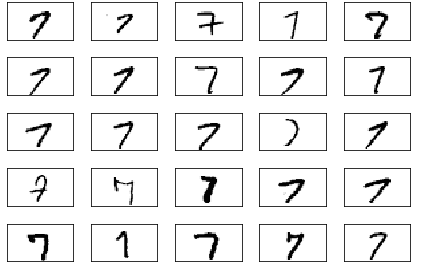
通过使用上面两行代码, 我们首先读入 magic number, 它是一个文件协议的描述, 也是在我们调用 fromfile 方法将字节读入 NumPy array 之前在文件缓冲中的 item 数(n). 作为参数值传入 struct.unpack 的 >II 有两个部分:

* >: 这是指大端(用来定义字节是如何存储的); 如果你还不知道什么是大端和小端, Endianness 是一个非常好的解释. (关于大小端, 更多内容可见<<深入理解计算机系统 – 2.1 节信息存储>>)
* I: 这是指一个无符号整数.

为了了解 MNIST 中的图片看起来到底是个啥, 让我们来对它们进行可视化处理. 从 feature matrix 中将 784-像素值 的向量 reshape 为之前的 28\*28 的形状, 然后通过 matplotlib 的 imshow 函数进行绘制:

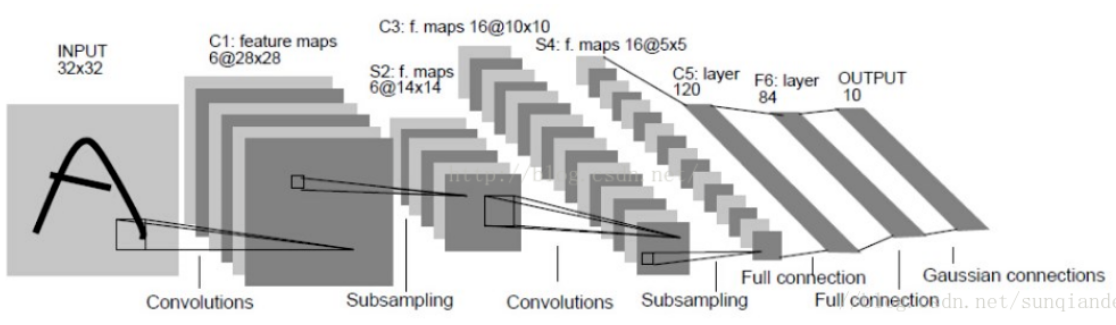


看几个数字7的样本

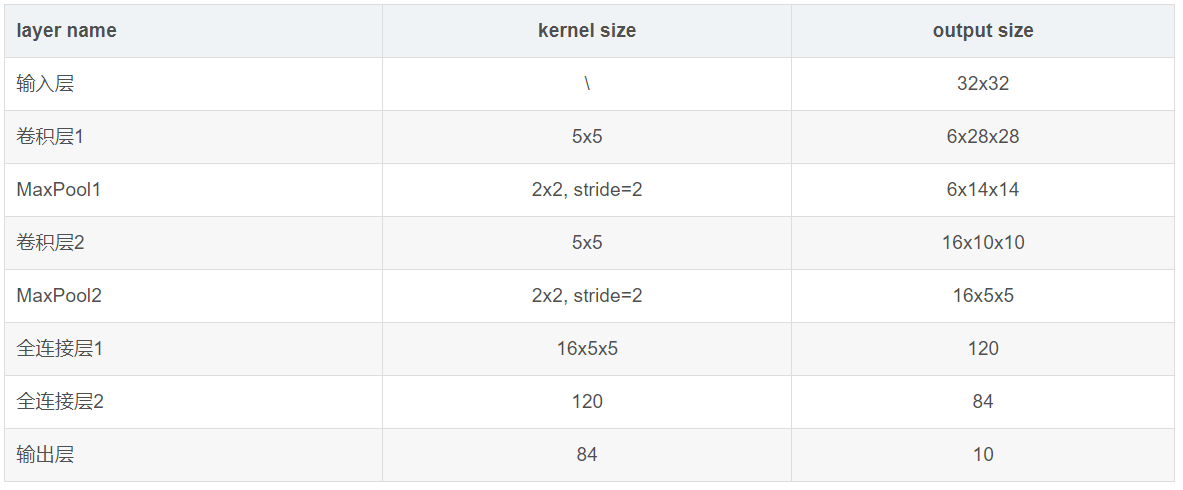


2. 训练: 训练LeNet模型, 完成手写数字识别任务.

LeNet是一个简单的卷积神经网络(Convolution Neural Network)



网络参数



main.py

用pytorch 中torchvision库中自带的mnist数据库读入训练, 测试数据;

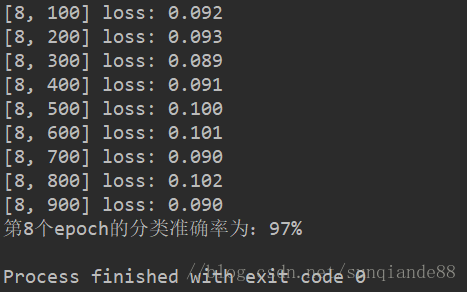
训练8个epoch

Batch\_size: 64

学习率: 0.001

优化方法: SGD

实验结果



思考: 如何画loss曲线?