准备知识

目录

手写数字识别任务

手写数字识别，就是对由人书写出来的形态各异的 0,1,2,3,4,5,6,7,8,9 这 9 个阿拉伯数字进行分类。它与其他手写文本识别任务没有太大的区别，所以一般运用于手写数字识别任务算法，可以经过细微的改动就用到其他文本识别任务中去。

手写数字识别中主要有三个困难：一是输入形态的差异，由于不同的人书写习惯上有较大差异（如图 1），所以同样的数字在实际中会有很大的区别，如何保证算法能准确的抓取到这些区别中的共性；二是如何处理实际手写数字识别中输入的尺度不同，所谓尺度不同就

Figure 1 同一个数字会有不同的形态

是书写的位置、数字的大小的区别，实际书写中没有九宫格之类的标准区域来校准书写，所以处理的数字往往大小不一、甚至会有不同程度的歪斜或者扭曲；三是连续数字情况下的数字分割问题，在实际检测中，往往是连在一起的一串数字进行识别，如何克服其中大小不一、长短不同、连笔、误笔等问题，将这一连串数字切分成单个数字进行识别。

目前卷积神经网络（Convolutional neural networks）是处理困难一和二的最佳算法，困难三的处理则往往归为图像分割领域的问题，不单单是模式识别的问题。感兴趣的同学可以参考这两篇文章进行了解[1,2]。

本次实验的主要目的，是通过手写数字识别任务，锻炼对于后向误差传递和随机梯度下降算法的掌握。为了简化任务难度，本次实验采用**全连接**神经网络，而不是卷积神经网络，这两者的区别主要在卷积神经网络借鉴了生物的视觉系统特性，对同层的网络连接进行一种特殊的共享方式，既多个神经元会共享同一组的网络连接，但是这两者在网络的优化方面都是一致的，既都使用后向误差传递算法（back propagation）和随机梯度下降算法（stochastic gradient descent）。

MNIST 数据集

MNIST 数据集[3]，是一个对于单独的手写数字进行识别数据集。原始的 MNIST 数据包含 6 万个训练样本，和 1 万个测试样本，每个样本都是一个具体的手写数字，同时是一张 28x28 像素的黑白图（见图 1）。

原始的 MNIST 数据，是将所有的原始数据压缩在一个大文件中，如 train 的数据为 train-imges-idx3-ubyte 和 train-labels-idx1-ubyte 这两个文件，一个保存图片，一个保存类标，需要专门用程序按一定的格式读取才能将每张图读取出来。

为了便于编程以及加快运算，我们提供使用 python 处理好的 MNIST 数据集的子集数据 data.pkl （1 万训练数据和 1 千测试数据），可以在 python 中通过如下语句：

```
(train_images, train_labels, test_images, test_labels) = cPickle.load(open('./mnist_data/data.pkl', 'rb'))
```

快速将数据按（train_image, train_label, test_image, test_label）的格式载入内存。

神经网络

人工神经网络（Artificial neural networks，ANN）在人工智能领域简称神经网络（简写 NN），是一种模拟生物神经连接结构设计的计算模型。理论上，包含一个隐藏层的 NN，在隐藏神经元数据足够的情况下，可以近似任意的非线性函数。超过一个隐藏层的 NN 也被称为多层感知机（multilayer perceptron），这个说法主要是因为最初神经网络被提出来时，神经元使用的模型就是 perceptron 模型（一种经典的线性分类器）[4]，但实际上，经过如此多年的发展，现在常用的神经元早就是非线性的（sigmoid、relu 等），所以 multiplayer perceptron 的说法只是种约定俗成的叫法，并不精确。

神经网络模型一般需要预定义好模型结构，然后通过训练数据调整模型中的参数，以使得模型尽量贴近给定的数据分布。一般来说根据网络中是否包含循环连接（recurrent link），会将神经网络分为直连式神经网络（feed-forward NN，FNN）和循环神经网络（recurrent NN）。循环神经网络主要处理带有时序关系的输入信息，如语音等，感兴趣的同学可以参见[5]。本任务主要使用的是 FNN，下面主要对**全连接**（既每一层都是个有向无回二分图结构）的 FNN 进行介绍。

人工神经网络的数学模型

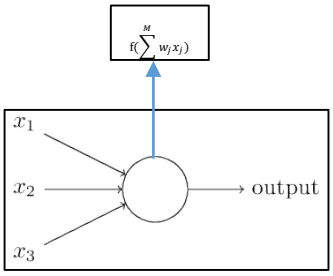


Figure 2 人工神经元

神经网络是以人工神经元为节点的图模型，具体的，每个人工神经元（artificial neuron）的数学模型如下[6]：

$$y(x, w) = f(\sum_{j=1}^{M} w_j x_j) \quad (1)$$

其中 x 是 M 维输入向量，x_j是其在第 j 维度上的值，y 是神经元的输出，w 是权值向量，w 是需要通过训练进行调整的参数，f 是一个非线性函数。见图 2。

为了便于在层级的网络结构中进行推导，我们将每个神经元的输入和输出都称为激活值，因为每个神经元的输入是前一层神经元的输出，而每个神经元的输出又是下一层神经元的输入。对于与输入 x 相连的神经网络层，既第一个隐层（hidden layer），它的第 j 个神经元的输入为：

$$a_j{}^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i \qquad (2)$$

a_j就是第一个隐层第 j 个神经元的输入激活值，$w_{ji}^{(1)}$的上标(1)表示这是属于第一层的权值，D 是说输入 x 向量的维度，然后经过神经元的非线性函数 h 后（对应式（1）中的 f），产生输出激活值$z_j{}^{(1)}$（既式（1）中的 y）：

$$z_j{}^{(1)} = h(a_j) \qquad (3)$$

然后对于第二个隐层层的第 k 个神经元来说，它的输入激活值a_k就等于

$$a_k{}^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j{}^{(1)} \qquad (4)$$

上式中，M 是第一个隐层含有的神经元数量，M 个神经元会有 M 个激活输出激活值$\langle z_1, z_2, \dots, z_M \rangle$。

假设一共有 L 个隐藏层，那么最后整个神经网络的输出，就是以第 L 个隐藏层的输出激活值为输入，进行的一个变换（线性变换或者非线性变换，主要看应用场景）。以回归问题为例，假设神经网络要预测一个 K 维的实值向量，那么第 k 维的预测输出为：

$$y_k = \sigma(a^{(L)}) \qquad (5)$$

上式中$a^{(L)}$是一个向量，它包含第 L 个隐藏层的所有神经元的输出，其中的每个值由(2)(3)(4)式迭代产生。σ也一个连续的函数。整个结构如下图 3 所示：

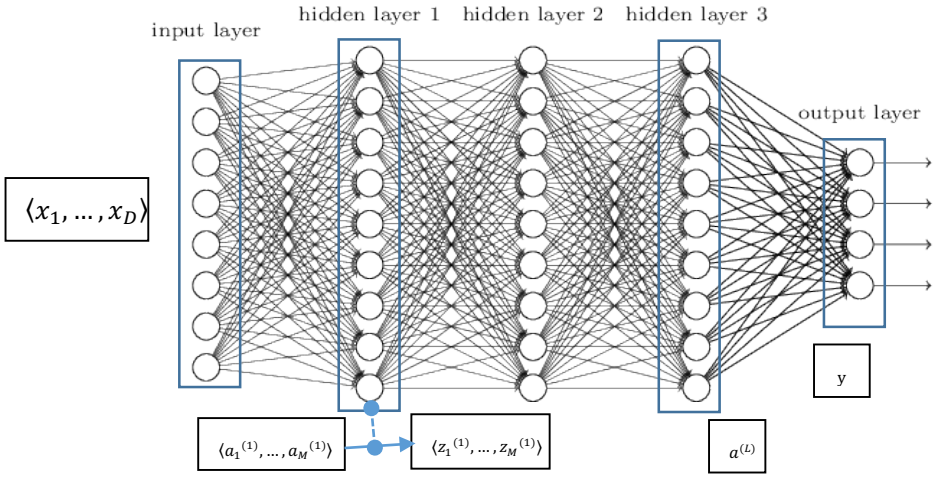


Figure 3 直连式全连接神经网络

将式（2）~（5）联立起来，就是一个神经网络的数学模型，比如，对于一个 L=2 的神经网，它的第 k 维输出为：

$$y_k(x,w) \overset{(5)}{\Rightarrow} \sigma\left(a^{(2)}\right) \overset{(4)}{\Rightarrow} \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} z_j{}^{(1)}\right) \overset{(3)}{\Rightarrow} \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h(a_j)\right) \overset{(2)}{\Rightarrow} \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} \sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)$$

上式中，x 是输入向量，w 是$w^{(1)}$和$w^{(2)}$构成的高维矩阵。对于 L≥ 2的情况，上式中的（4）和（3）式需要不断迭代，直至到达输入层再使用（2）式；对于 L=1 的情况，则没有上面的（4）和（3）的过程。

可以看出，在直连式的神经网络中，由 x 计算 y 的过程，可以看成是输入信息通过网络向前传播的过程，并在传播过程中通过权值 w 和非线性函数 h 进行不断的变换，逐渐将其中需要的信息抽取出来，并最后通过σ函数组合成需要的 y。这么个过程有个特点，就是网络结构中没有回路，信息是一路传递到输出的，因此拥有这种特性的网络都被称为 feed forward network。

神经网络的输出与评价指标

对于给定的输入样本$\langle x, y^t \rangle$，x 是输入数据，y^t是需要预测的值，y^t可以是离散的分类类标或者连续的实值。而神经网络是个将给定的输入向量 x 通过逐层非线性转换变化成输出向量 y 的函数，y 和y^t的维度保持一致。一般来说，这个 y 是实值函数，就是说可以直接应用于回归预测，同时如果我们将 y 的输出看成是待预测的类别的概率的话，也可以转化成预测离散的分类问题。

因为我们需要处理的手写数字识别是个互斥分类问题，这里专门说明下如何用神经网络处理这种问题，处理的关键是从概率的角度来看待神经网络的输出值。手写数字识别一共有 10 个类，对于一个输入图片x，它是个二维的向量（28×28像素），我们可以将之转成一个一维向量（1×784）,这就是神经网络的输入向量；对于 x，它的正确类别y^t是 0~9 中的一个数字（互斥的），我们将之转换为维度为 10 的 one-hot 向量，既只有一个维度为 1，其他都为 0 的向量，维度大小和类别数一致。比如y^t是 5，那么

$$y^t = \langle 0,0,0,0,1,0,0,0,0,0 \rangle$$

。这个 one-hot 向量可以理解为，输入向量 x 为类别 5 的可能性是 100%，为其他类别的可能性为 0%。这样如果我们**设置**神经网络的输出 y 是 x 属于 0~9 个类的概率，那么学习目标就是最大化 x 属于类y^t的条件概率值$\mathrm{p}(y^t|\mathrm{W},\mathrm{x})$，其中 W 是网络所有连接权值构成的参数集合。

在这个考虑下，我们可以直接限制输出函数$\boldsymbol{\sigma}$的值域在[0,1]之间，这样网络预测的 y 就是各个维度都在这个值域范围内的向量，如 y= ⟨0.1,0.2,0.3,0.4,0,0,0,0,0,0⟩；对于这样一个 y，可以很容易的确定预测的结果，既概率最高的那个维度对应的类别（在这个例子里是数字“3”）就是神经网络预测出来的结果。

特别的，因为对于手写数字识别任务来说，对于一个给定的 x，它只可能属于 0~9 个类中的一个，所以这是个互斥多类的分类问题。因此，对于模型的输出函数$\boldsymbol{\sigma}$，不仅要求它的值域为[0,1]之间，同时它输出的结果应该是个 0~9 之间的概率分布（既类别之间有相互依赖的关系），既预测得到的向量 y 的各维度之和为 1。这里我们使用 softmax 函数，softmax 函数对于输入激活值$\mathrm{a} = \langle a_1, \ldots, a_K \rangle$进行如下计算：

$$\mathrm{softmax(a)} = \langle \frac{\exp(a_1)}{\sum_j \exp(a_j)}, \ldots, \frac{\exp(a_K)}{\sum_j \exp(a_j)} \rangle$$

但是我们需要有个度量来衡量神经网络预测的 y 和实际的y^t之间的差别，有了这个差别我们才好有依据的调整神经网络。在进行预测时，我们可以直接通过网络模型对于给定样本集合的预测正确率来判断这个模型的好坏（既给定的样本中，有多少分对了），但是在训练时这个标准无法衡量出模型对于具体类的概率预测的有多准（既给定一个样本 x，它预测的$\mathrm{p}(y^t|\mathrm{W},\mathrm{x})$够不够高）。因为我们的模型直接预测的是，对于给定类的概率高低，所以我们在训练时，用来衡量模型好坏的标准就应该是它对于给定数据的类别分布与正确的类别分布是否一致，既对于给定的数据集合 x，模型对正确类别预测出来的联合概率和实际的正确的概率（既是正确的是 1，既训练提供的类标是完全正确的）是否一样。

在给定网络模型结构（层数、隐藏单元数、隐藏单元之间的连接方式、非线性激活函数，

输出激活函数等不带参数的部分）的情况下，模型的权值参数 W 决定了模型的预测结果，既对于一个给定的 x，模型预测其为类 k 的可能性（也就是向量 y 的第 k 维的值）$y_k(x,w)$是一个条件概率$\mathrm{p}(\mathrm{k}=1|\mathrm{x},\mathrm{w})$，

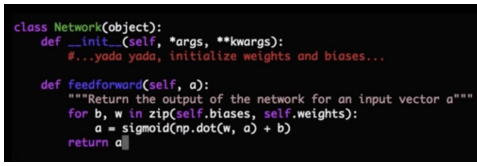
$$\mathrm{p}(\mathrm{k}=1|\mathrm{x},\mathrm{w})=y_k(x,w)$$

。而对于一个有正确类标的训练数据(x,t)，t 会转换成一个 one-hot 编码，编码维度对应分类的类别数，既

$$\mathrm{p}(t_k|\mathrm{x})\in\{0,1\}且\sum_k \mathrm{p}(t_k|\mathrm{x})=1$$

。那么对于一个一共 K 个类别的分类问题，当训练数据总量为 N 时，根据贝叶斯的链式法则，网络模型对于所有的训练数据的正确类标

$$\mathrm{t}=\langle t^1,..,t^N\rangle$$



预测出来的联合条件概率为

$$\mathrm{p}(\mathrm{t}|\mathrm{x},\mathrm{w})=\prod_{n=1}^{N}p(t^n|x_n,w)$$

，因为神经网络对于每个样本的预测也是个联合概率（是个概率分布）

$$p(t^n|x_n,w)=\prod_{k=1}^{K}p(t_k^n=1|x_n,w)^{t_k^n}$$

，所以当给定了训练集$\langle x,t\rangle$和网络模型时，在给定输入 x 的情况下，模型对于正确类标 t 预测出来的概率分布

$$\mathrm{p}(\mathrm{t}|\mathrm{x},\mathrm{w})=\prod_{n=1}^{N}\prod_{k=1}^{K}p(t_k^n=1|x_n,w)^{t_k^n}$$

，因为我们假定训练数据给出的 t 都是绝对准确的，所以理想状态下

$$\mathrm{p}(\mathrm{t}|\mathrm{x})=1$$

，我们希望构建的神经网络模型其预测的分布$\mathrm{p}(\mathrm{t}|\mathrm{x},\mathrm{w})$可以尽量贴近$\mathrm{p}(\mathrm{t}|\mathrm{x})$，又因为$\mathrm{p}(\mathrm{t}|\mathrm{x},\mathrm{w})$是个概率分布，所以它极大值为 1。所以训练时$\mathrm{p}(\mathrm{t}|\mathrm{x},\mathrm{w})=\prod_{n=1}^{N}\prod_{k=1}^{K}p(y_k^n=t_k^n|x_n,w)^{t_k^n}$就可以衡量我们模型的好坏，且这个值越大越好，我们的训练目标就是寻找一组适合的网络连接权值 W 最大化这个值：

$$\max_w\prod_{n=1}^{N}\prod_{k=1}^{K}p(t_k^n=1|x_n,w)^{t_k^n}$$

但是这个式中都是乘积项目，在计算机中连续乘积很容易超过数值精度，因此我们通过 ln 函数，将之转换为求和（ln 是个单调增函数，所以这个转换不会损坏原来的最大化问题）

$$\max_w\sum_{n=1}^{N}\sum_{k=1}^{K}t_k^n\ln p(t_k^n=1|x_n,w)$$

在这里要注意，t^n是转成一个 one-hot 向量，所以每一个$p(t_k^n=1|x_n,w)=y_k(x,w)$都是对$t^n$的一个预测，所以$\prod_{k=1}^{K}t_k^n p(t_k^n=1|x_n,w)=y_k(x,w)\ s.t.\ t_k^n==1\ is\ true$。

通常在处理的时候，喜欢将最大化问题乘个-1 当成最小化问题来求解

$$\min_w\sum_{n=1}^{N}\sum_{k=1}^{K}-t_k^n\ln y_k(x_n,w)$$

。因此我们可以定义训练时候的误差函数 E(w)为：

$$E(w) = \sum_{n=1}^{N}\sum_{k=1}^{K} -t_k^n \ln y_k(x_n, w) \quad (6)$$

$$\min_w E(w)$$

这个函数（6）就是我们训练手写数字识别时的评价指标，在不同的文献中又称为误差函数（error function）或者是损失函数（loss function）。

在信息论中，（6）又被称为交叉熵，当 t 的分布确定时（在这里因为 t 是给定的正确类标，所以它的分布的确是确定的），交叉熵可以用来度量 t 的分布与 y 的分布之间的距离（又称 KL 散度，由 t 的熵、t 与 y 的交叉熵之和构成，细节这里不细述，感兴趣的同学可以自行调研）。

另外，为了防止过拟合，我们在损失函数中加入L_2正则：

$$E(w) = \sum_{n=1}^{N}\sum_{k=1}^{K} -t_k^n \ln y_k(x_n, w) + \frac{\lambda}{2}\sum|w|^2 \quad (7)$$

其中λ是控制正则项粒度的参数。

梯度下降算法

在我们有了损失函数$E(w)$作为训练时的评价指标时，我们接下来如何根据$E(w)$的值来调整模型呢？如果这个$E(w)$和二次凸函数一样有解析解的话，那我们可以很方便的得出结论，但是很不幸的是，这个函数一般情况下没有解析解，所以我们使用迭代式的数值计算方法来求解这个问题[6]，一般来说，这类数值计算方法通过

$$w^{t+1} = w^t + \Delta w^t \quad (8)$$

来更新权值 w，其中 t 是迭代的轮数，Δw^t是第 t 轮更新的权值大小。

梯度下降优化算法（gradient descent optimization）的核心，就是使用损失函数$E(w)$对于权值 w 的梯度信息作为（8）式中的$\Delta w = -\frac{dE(w)}{dw} = -\Delta E(w)$。并且对（8）进行些修改：

$$w^{t+1} = w^t - \eta\Delta E(w^t) \quad (9)$$

式（9）中的$\eta > 0$，是一个被称为学习率的参数，用以控制使用梯度信息的更新粒度。

特别注意，根据公式（7），每次计算损失函数都要在整个数据集上进行计算，并进一步的根据公式（9）去计算新的 w，对于样本量很大的情况，这样做极为耗时，同时需要极大的存储空间。实际使用中我们基于 batch 的随机梯度下降算法（stochastic gradient descent），既每次从整个数据中随机抽取一部分数据（这部分的数据量称为 batch size）,在这部分数据上计算公式（9）和更新梯度。

后向误差传递算法

（该部分请自己完成，这是本次实验的核心）

参考文献

[1] *Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998*

[2] *Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel and Baird H. S.: Constrained Neural Network for Unconstrained Handwritten Digit Recognition, in Suen, C (Eds), Frontiers in Handwriting Recognition, CENPARMI, Concordia University, Montreal, 1990*

[3] *homepage of MNIST : http://yann.lecun.com/exdb/mnist/index.html*

[4] *Wikipedia of Perceptron: https://en.wikipedia.org/wiki/Perceptron*

[5] Chapter 10: Sequence modeling: recurrent and recursive nets. *Ian J. Goodfellow, Yoshua Bengio, Aaron C. Courville: Deep Learning. Adaptive computation and machine learning, MIT Press 2016, ISBN 978-0-262-03561-3, pp. 1-775*

[6] Chapter 5: Neural Network. *Christopher M. Bishop: Pattern recognition and machine learning, 5th Edition. Information science and statistics, Springer 2007, ISBN 9780387310732, pp. I-XX, 1-738*