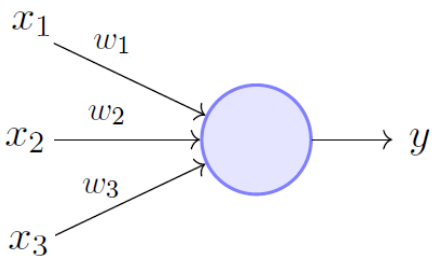
人工神经网络

董峦 新疆农业大学

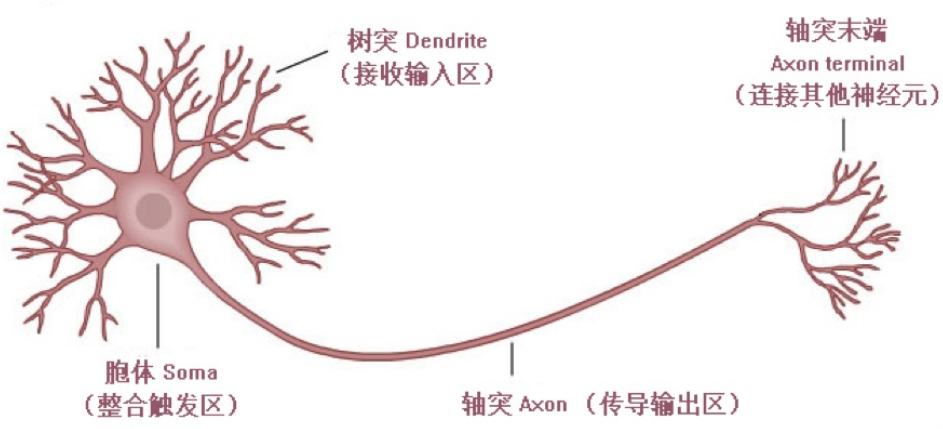
引言

在图像、语音和自然语言中广泛存在模式（pattern），在分辨模式方面人类的能力远远超越计算机。由于计算机需要明确的指令才能工作，而人类还无法将自己的大脑活动用精确的指令描述出来，从而通过编程把该能力赋予计算机，所以计算机始终是僵化的机械。那么能不能让计算机模拟人脑的活动过程，从而获得人类的能力呢？这方面研究可以追溯到19世纪，当时人们已经知道人类意识的物质层面表现为一组神经元的活动，到上世纪中页研究人员模拟神经元的结构提出了下图所示的人工神经元——感知机（perceptron）模型，提出了 Delta 学习规则用于感知机的训练，并制造出硬件版的感知机 `Mark I Perceptron`，向人们展示了一种机器自我进化的方式。下图的感知机模型中输入 x 乘以一组权重 w 再经过某种运算（图中用浅紫色单元表示）得到输出 y，其中的运算在感知机中是单位阶跃函数，模拟神经元的激活或静默。



Perceptron Model (Minsky-Papert in 1969)

观察实际神经元的结构，如下图所示，可以发现人工神经元对该生物过程做了高度抽象和概括。生物世界的神经元利用树突从其它神经元获得信号，信号在神经元内部整合后经过轴突传递到其它神经元，大量的神经元按这种模式协同工作，人类便有了我们所说的智能。不过人类的意识活动到底该怎么解释目前还没有定论，从神经元的工作方式到思想感情、逻辑推理这些意识层面的现象经历了哪些过程我们仍然知之甚少。人工智能领域也并不是所有人都认可上述模型，那些认为智能存在于神经元的连接和作用方式的人属于连接主义（connectionism）学派，其它还有符号主义和行为主义学派，目前看来连接主义处于上风，近几年深度学习（deep learning）的流行让该学派处在聚光灯下。



本文将利用人工神经网络识别手写数字，描述模型的建立和优化过程，观察计算机究竟学到了什么。

数据集

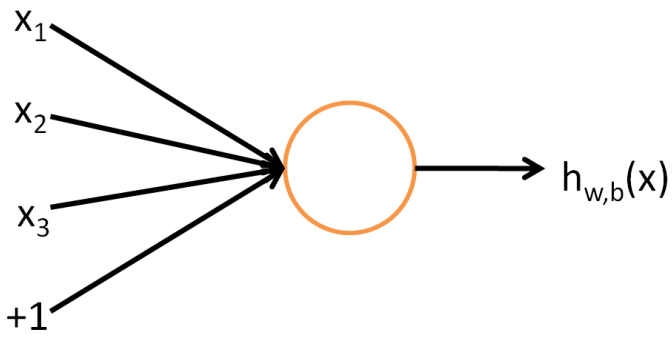
当我们开展一项研究时最好从简约而不简单的事物着手，比如生命科学研究一般采用模式生物（小鼠、果蝇、酵母等），那么机器学习研究有没有这种“模式数据集”呢？下面的手写数字数据集MNIST（http://yann.lecun.com/exdb/mnist/）就是这种数据集。该数据集包含0~9的手写数字图像，训练和测试集里分别有60000和10000幅图像，图像是28×28的灰度图，数据集里的一些样本如下图所示。

在该数据集的官网上可以看到人们目前在该数据集上的测试错误率是0.23%，可以说是非常高了，继续刷新该指标可能是一种过拟合的情况。2019年研究人员扩充了该数据集的测试集，又增加了50000个测试样本（https://github.com/facebookresearch/qmnist），为该数据集注入了新的生命力。Fashion-MNIST（https://github.com/zalandoresearch/fashion-mnist）数据集在数据规模和图像尺寸上与 MNIST 相同，是后者的另一个替代。

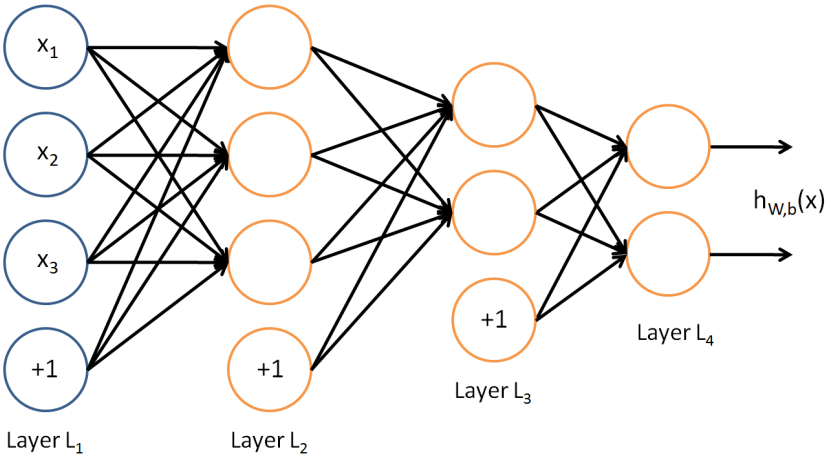
对该数据集用 LDA（或PCA） 降维后可视化，如下所示，可见数据成簇的态势比较明显。

模型

人工神经网络中单个神经元如下图所示，可以看到它的结构与感知机相同，输出 $y = h_{w,b}(x) = h(\sum_{i=1}^{3} x_i w_i + b)$。不同的是非线性函数 h 不再局限于阶跃函数。然而如果仅仅停留在该结构上则解决不了什么复杂的问题，比如马文·明斯基曾经指出感知机无法解决异或问题（即线性不可分问题），他的评论直接造成人工智能研究进入“寒冬期”。



如果在水平和纵深层面堆叠该结构则形成了人工神经网络，如下图所示，该网络理论上可以拟合任何函数（Universal approximation theorem， 通用近似定理）。对于手写数字识别来说，把图像看做输入，“数字”看做输出，联系这个输入和输出的未知函数可以用人工神经网络来拟合。训练好的神经网络是这个未知函数的良好替代，利用神经网络将能够根据输入预测（识别）图像中的数字。



上图是一个3层神经网络（桔色神经元组成的部分），输入层不计入层数。输入层的神经元用蓝色圆圈表示，表示原始数据，图中有三个，标记“+1”的神经元代表偏置项。最右侧是输出层，有两个神经元。中间的两层是隐含层，分别有三个和两个神经元。图中，神经网络的计算过程从左向右进行，每一层的计算过程类似，下一层可以看做再下一层的输入。用数学表示是：

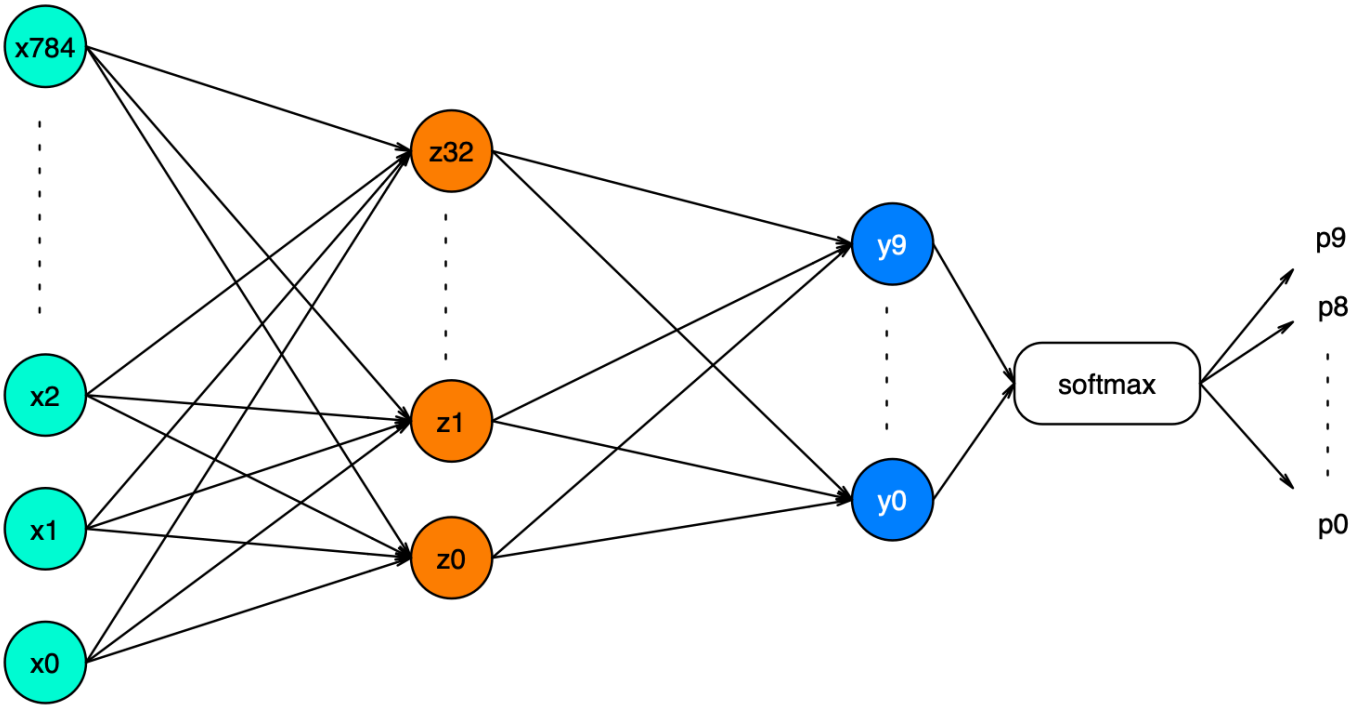
$$y = h(W_3^T g(W_2^T f(W_1^T X + b_1) + b_2) + b_3)$$

其中W_1、W_2、W_3是每两层神经元之间连接的权重，b_1、b_2、b_3是偏置项与神经元之间连接的权重，f, g, h是隐含层和输出层上的非线性函数。

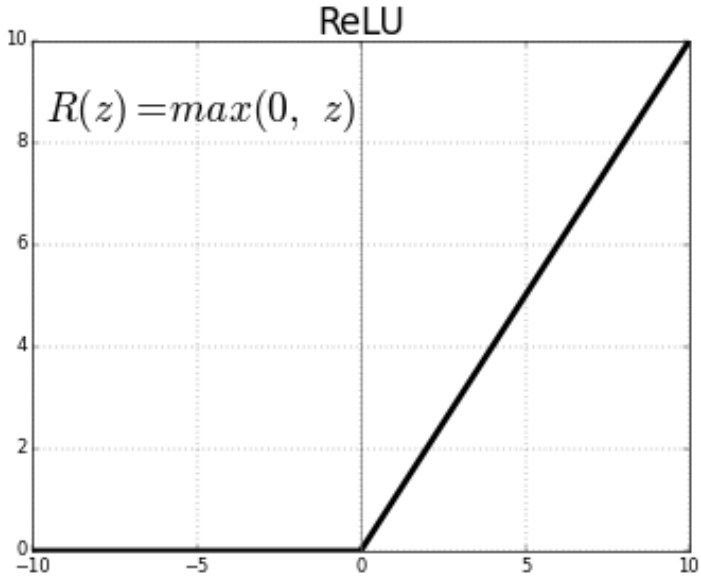
本文采用单隐层神经网络，如下图所示。输入层有784+1个神经元（28×28的图像有784个像素），是一幅手写数字图像像素的个数，即本文以原始像素为输入。隐含层有32个神经元（不算偏置项的话），该数值可以更改，理论上数值越大模型能力越强，但训练难度也随之增大。输出层有10个神经元，对应0~9十个数字。为了将模型输出转化为概率分布，一般在输出层神经元上施加 softmax 激活函数，本文中 softmax 输出的计算过程是

$$p_i = \frac{e^{y_i}}{\sum_{j=0}^{9} e^{y_j}}$$

给定输入，模型的预测值是 $\arg\max_i(p_i)$ 。



隐含层的非线性激活函数采用 ReLU，函数如下图所示



该函数的特点是，自变量大于零时导数为1，自变量小于等于零时导数为零。

损失函数

上一节提到，softmax 函数的输出可以看做分布，因此我们用交叉熵（cross entropy）衡量这个分布与目标分布（即标签）的偏差。交叉熵作为损失函数，单个样本的损失定义为

$$L_i = -\sum_j \hat{p_{ij}} log(p_{ij})$$

其中 $\hat{p}_i$ 是真实分布，真实分布根据数据的标签用 one-hot 形式表示，比如某个图像的标签是 3，则 $\hat{p}_i = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ 。i 是样本下标，j 是输出（或标签）的元素下标。交叉熵衡量两个分布的差异程度，本文是个多分类问题，输出可看做一个多项式分布。由于 $\hat{p}_i$ 中只有一项为 1，所以 L_i 简化为：

$$L_i = -log(p_{ij}) \mathbb{1}\{\hat{p_{ij}} = 1\}$$

其中 $\mathbb{1}\{\}$ 是示性函数。对全部样本来说损失函数为 $L = \frac{1}{m} \sum_i L_i$，m 是样本总量。

如果采用 PyTorch 框架的话，只要调用该框架已实现的交叉熵函数即可，并且既不用计算 softmax 输出也不用将标签转换成 one-hot 形式。

评价指标

与上一讲的二分类问题不同，本文涉及的问题是多分类问题。本文用 Macro Average F1 score 和 Weighted Average F1 score 评价多分类模型的性能。

- Macro Average 指某个二分类指标在多个类别上的**算数平均**。以 F1 值为例，把模型在某个数字上的分类看成二分类问题，计算 F1 值。把每个类别上的 F1 值取平均就是 Macro Average F1 值。
- Weighted Average 是某个二分类指标在多个类别上的**加权平均**，权重是不同类别在样本总体中所占的比重。

优化

上述人工神经网络的代价函数在参数空间是非凸的，这意味着用简单的梯度下降优化方法会受困于鞍点和局部极小值点。动量法（Gradient Descent With Momentum）是克服该问题的有效方法， 动量法的迭代式是

$$v^{k+1} = \beta v^k + \nabla L_{\theta^k}$$
$$\theta^{k+1} = \theta^k - \alpha v^{k+1}$$

其中 v 是动量，θ 是神经网络的参数，包括每一层的权重和偏置项。v 的维度与 θ 相同， v^0 （初始值）所有元素都是零向值。β 是一个标量，影响动量的大小，α 是学习率。

将上面迭代式的第一项展开为

$$\begin{aligned} v^{k+1} &= \beta(\beta v^{k-1} + \nabla L_{\theta^{k-1}}) + \nabla L_{\theta^k} \\ &= \beta^2 v^{k-1} + \beta \nabla L_{\theta^{k-1}} + \nabla L_{\theta^k} \\ &= \beta^3 v^{k-2} + \beta^2 \nabla L_{\theta^{k-2}} + + \beta \nabla L_{\theta^{k-1}} + \nabla L_{\theta^k} \\ &= \beta^n v^{k-n+1} + \beta^{n-1} \nabla L_{\theta^{k-n+1}} + \beta^{n-2} \nabla L_{\theta^{k-n+2}} + \ldots + \nabla L_{\theta^k} \end{aligned}$$

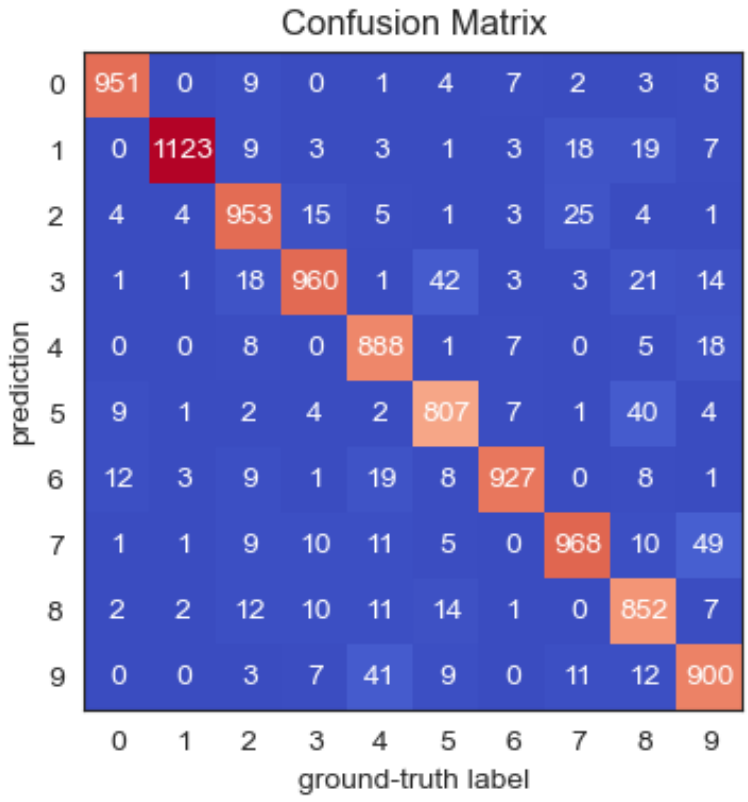
可见 v^{k+1} 是所有之前梯度的**指数加权平均**。通常 $0 < \beta < 1$ ，因此较新近的梯度对动量的贡献较大，但过去的梯度会在一定程度上遗留影响，即 v 具有一定“惯性”，β 越大“惯性”越大。

形象地说，因为动量具有“惯性”所以可以“冲出” 鞍点或局部极小值点，避免陷入常规梯度下降法的窘境。采用 PyTorch 的话只要配置优化器的 momentum 参数即可，如下

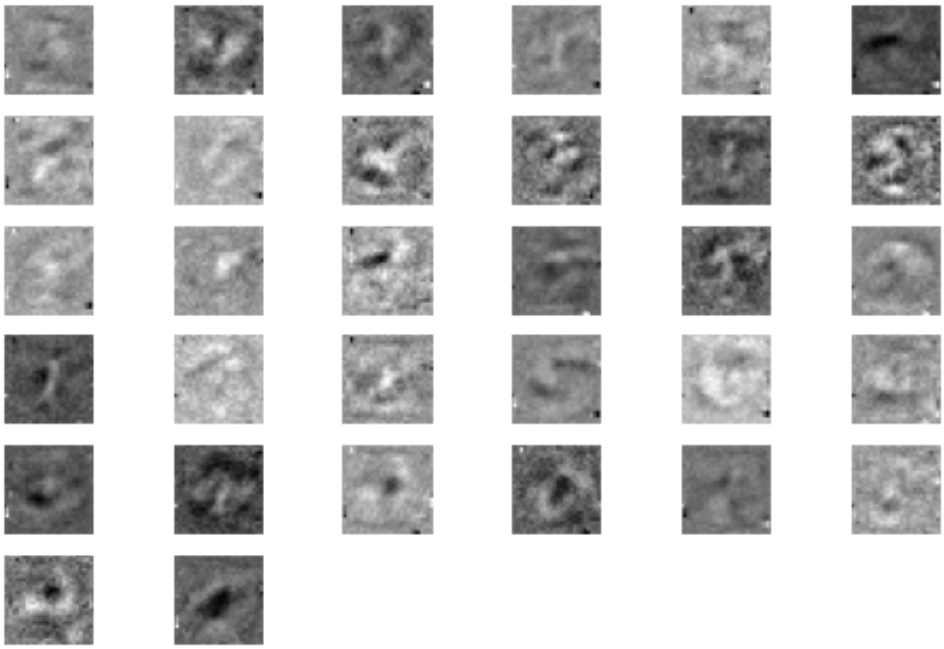
```
from torch import optim

optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.95)
```

测试

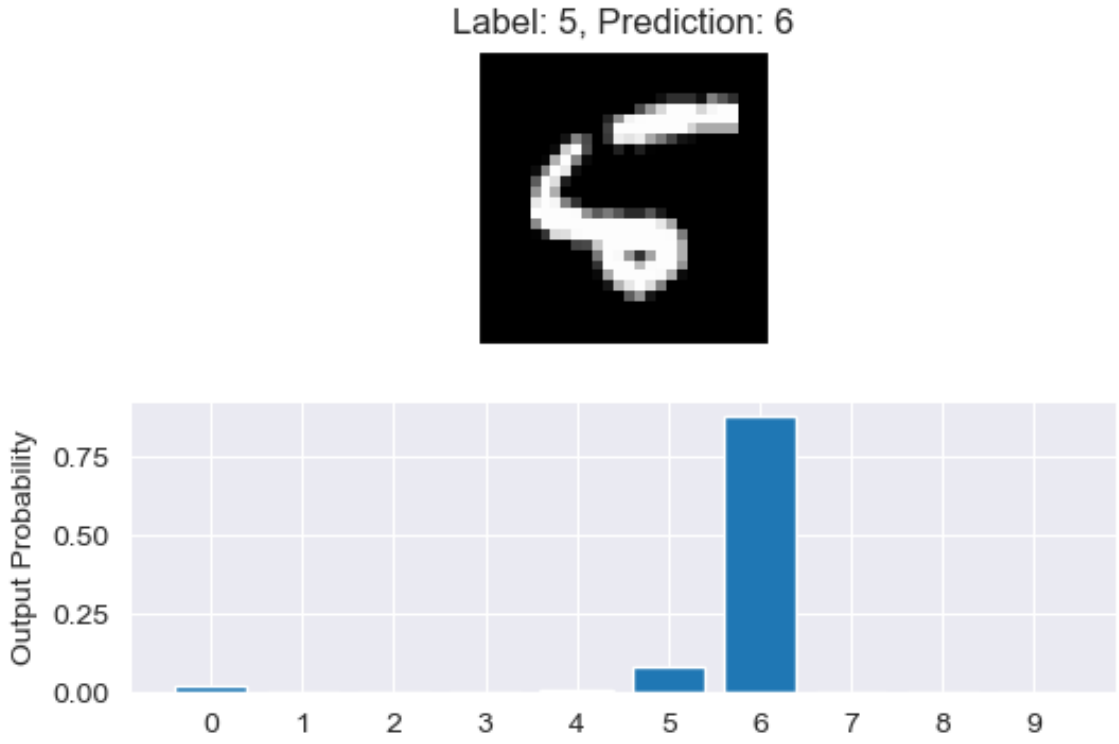
神经网络模型在测试集上的性能是：average precision 为 0.933，average recall 为 0.932，macro average F1值为 0.932，weighted average F1值为 0.933。混淆矩阵的热图如下所示，其中对角线上值表示正确预测的样本数量，其它位置上的值是假阳性和假阴性的数量。



可视化第一层神经网络的权重可以观察模型对哪些模式敏感，如下所示



在调试机器学习模型的时候观察错误预测的样本非常重要，它有助于开发人员检查错误，找出下一步调整的方向。以一个误判样本对应的模型输出为例，如下图所示，样本标签是5，但预测是6。从输出的分布情况看，模型判断样本是6的概率非常高，从样本的图像发现它跟数字6确实非常相似。由于手写的随意性和个性化，MNIST数据集里类似的情况还有很多。



小结

由于上文提到的通用近似定理（或万能逼近定理），神经网络模型是一种建模能力强大的模型。本文搭建单隐层结构的神经网络解决手写数字分类问题，F1值的加权平均达到0.93。2006年以来随着数据量、算力、优化技巧等方面的进步，深层神经网络席卷了各类机器学习方面的竞赛。但神经网络也有缺点，主要是可解释性差，因此常常被诟病为“黑盒”。

神经网络的优化是一个较难的课题，本文的模型结构简单优化难题不突出，但对于深层神经网络就不得不关注模型优化问题了。近些年研究人员做的大量工作都可归结为解决大型网络难以优化的问题。本文介绍的动量法就是一种克服非凸函数优化难题的工具。

在多分类问题中，当样本数量存在不平衡情况时，F1值的加权平均是客观的评价指标。为了克服样本不平衡问题，可在交叉熵函数中增加权重系数。PyTorch 已实现了该机制，具体可参考 `torch.nn.CrossEntropyLoss` 的文档。

思考

- 对于某一类样本来说，混淆矩阵中哪些数值代表假阳性哪些代表假阴性?
- 神经网络隐含层的激活函数采用 Sigmoid 函数对优化过程会有什么影响?
- 可以从哪些方面改进神经网络模型的性能?