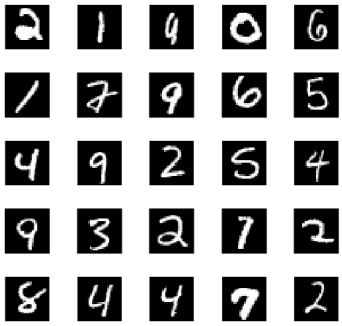
神经网络简介

数据集 MNIST

MNIST 数据集是一个数字的手写体数据集，如下图所示



数据集中的每张图片（单个手写数字）由 28 x 28 个像素点构成, 每个像素点用一个灰度值表示（0.0 表示白色，1.0 表示黑色，从 0 到 1 颜色越来越深）。这样，每个手写数字由 28 x 28 = 784 维的数组确定。

在数字分类中，每个手写数字将用 10 个分量的二进制数组表示，如 [1,0,0,0,0,0,0,0,0,0] 的第 0 位为 1，表示数字 0 (这里为了方便，下标从 0 开始)。

经过一些处理，数据保存在 mnistdata.mat 中 (对 MATLAB 编程)，它分为三部分：

- 训练数据：training_data_x 和 training_data_y，这里的**下标 x 和 y 分别表示输入和输出。**

  training_data_x 是 784 x 50000 的矩阵，**每列对应一个训练数据的输入**;

  training_data_y 是 10 x 50000 的矩阵，每列对应一个训练数据的输出。
- 测试数据: test_data_x 和 test_data_y。它与训练数据结构一样，只不过数据个数为 10000.
- validation_data: 这部分数据与测试数据结构和个数都一样，它暂时用不到。它的作用是为了调整参数，防止过拟合等。

数据导入
MNIST 的原始数据是以一种特殊方式保存的，必须转化为前面说的形式。这里不介绍如何转化，读者可在百度上搜索 MNIST，查阅相关信息。在当前目录下，可直接用`load mnistdata`加载需要的数据。对 Python，后面再说明。

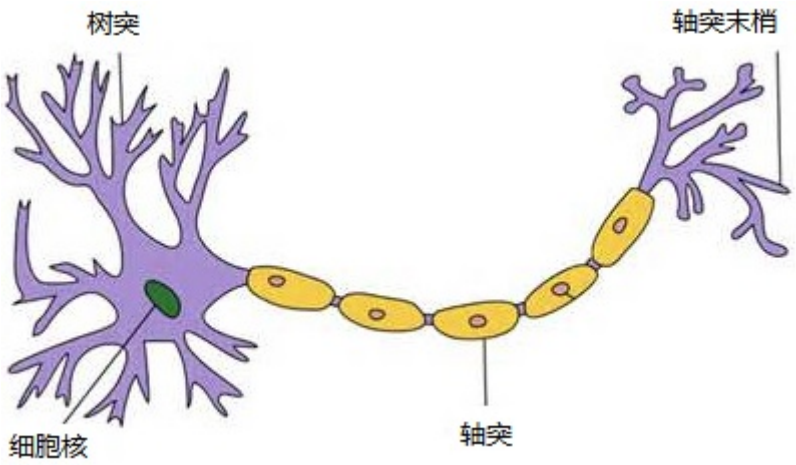
在 MATLAB 中可如下显示手写数字的图片

```
load mnistdata

figure,
set(gcf,'Units','normal');
set(gcf,'Position',[0.0,0.0,0.35,0.55]);

m = 5; n = 5;
for i = 1:m*n
    a = test_data_x(:,i);
    a = reshape(a,28,28)'; % transpose
    subplot(m,n,i), imshow(a);
end
```

神经网络的基本思想

神经网络很早就已提出，但受到当时算力的限制，未受到足够重视。神经网络为大家所熟知，可能得益于AlphaGo 的围棋大赛。在当前的算力下，以及 GPU 的加速支持，神经网络的规模变得非常巨大，称为深度神经网络。也就是说，神经网络的层数 (或深度) 不再是仅仅的几层，而可能是上百层、上千层甚至更多。

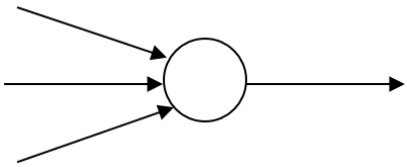
神经网络的设计思路非常朴素，它是对人体神经网络的一种简单模拟。



上图是生物神经元的示意图。神经元也称为神经细胞，是神经系统最基本的结构和功能单位，分为细胞体和突起两部分。

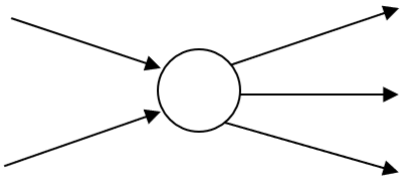
- 细胞体由细胞核、细胞膜、细胞质组成，具有联络和整合输入信息并传出信息的作用。
- 突起有树突和轴突两种。
    - 一个神经元通常具有多个树突，主要用来接受传入信息。
    - 轴突只有一条，其尾端有许多轴突末梢可以给其他多个神经元传递信息。

神经元的特点可抽象为下图

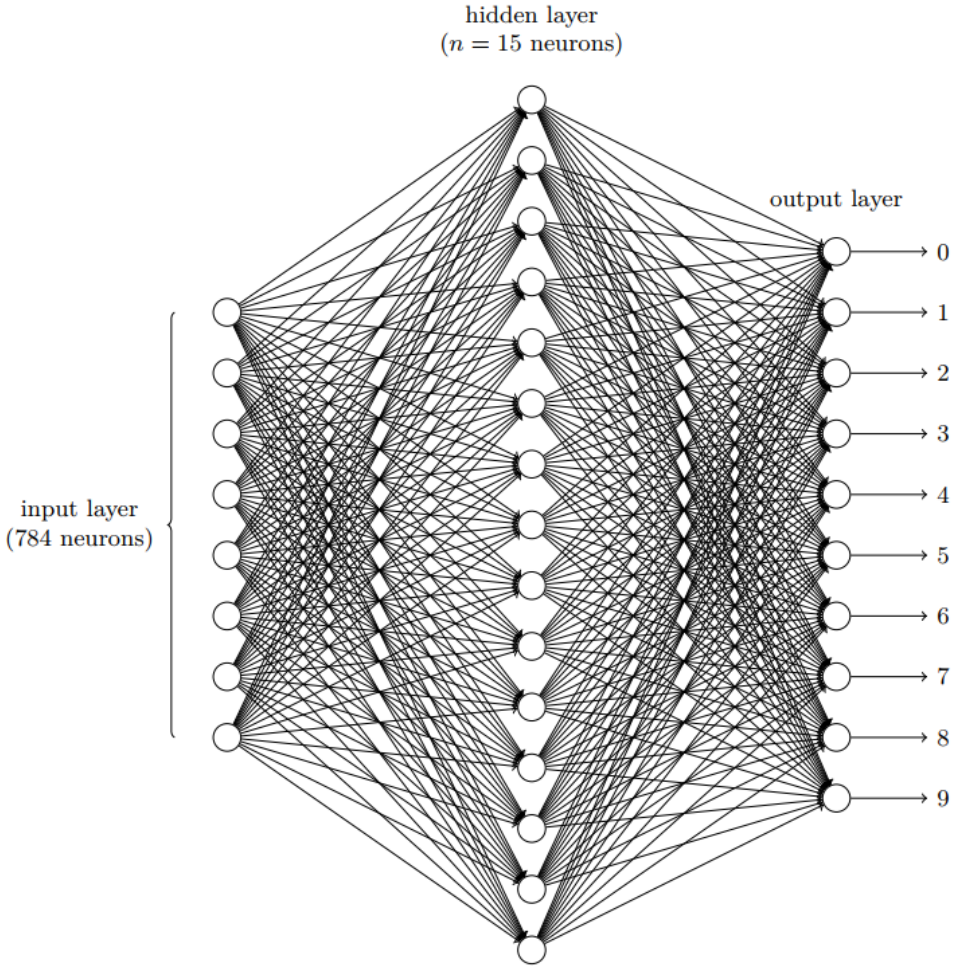


- 在该图中，圆圈相当于神经元的细胞体，左侧的入射线相当于树突，右侧的出射线相当于轴突。

- 由于轴突通过轴突末梢传递信号给其他神经元，因此通常省略轴突，直接给出多个出射线模拟轴突末梢，如下图所示



有了“神经元”，我们就可以将若干个这样的结构搭建起来，以模拟生物神经网络。例如，下图就是手写数字识别的简单神经网络图



人工神经网络的搭建是按层进行的。

- 第一层为输入层，对应输入信号；最后一层是输出层，对应期望输出；中间的所有层都称为隐藏层，用于信号的进一步加工。
- 层与层之间的神经元通过射线连接，这些射线相当于轴突末梢。

搭建好神经网络的结构后，剩下的就是设计神经元之间信号的传递规则。

- 一个简单的规则是线性传递。给定一个神经元信号 x ，线性规则是按 $wx+b$ 的方式传递给下一个神经元。
- 通常还会对这个“线性信号”进行额外处理，记为 $\sigma(wx+b)$，其中的 σ 称为激活函数。

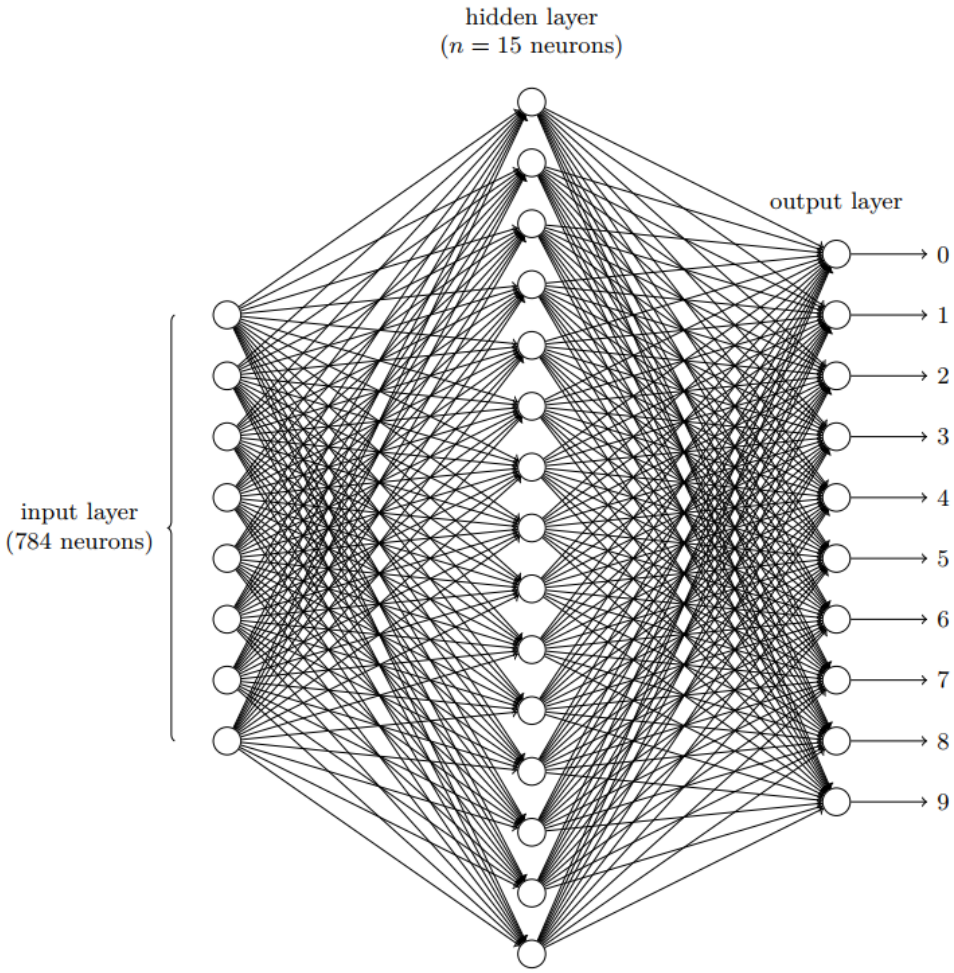
神经网络的搭建
步 1：搭建网络结构，关键是输入和输出层； 步 2：设计传递规则，关键是激活函数。

手写数字分类的简单神经网络

以手写数字识别为例。

> 神经网络的输入和输出视具体问题而定，为了更清楚，这里以手写数字分类为例。

- 该案例的输入是单张手写数字图片。
  - 每张图片由 28 x 28=784 个像素点构成, 每个像素点用一个灰度值表示。
  - 网络设计中，每个像素点用一个输入神经元对应，从而输入层有 784 个神经元。
- 该案例的输出是具体的手写数字。
  - 数字从 0 到 9 共 10 个，我们用 10 个分量的二进制向量表示。若规定从 0 开始编号，则数字 2 就是第 2 个位置为 1，其他位置为零的向量 [0,0,1,0,0,0,0,0,0,0].
  - 网络设计中，二进制向量的每个位置对应一个输出神经元，从而输出层有 10 个神经元。
  - 每个输出神经元输出 0 到 1 的数值，它表示分类为对应数字的概率。



以三层网络为例，即假设隐藏层仅有一层，下面给出数学表达。

- 输入层
  - 对手写数字分类，输入层是手写数字图片的信息。
  - 每张数字图片由 28 x 28 个像素点构成, 每个像素点用一个灰度值表示（0.0 表示白色，1.0 表示黑色，从 0 到 1 颜色越来越深）。
  - 这样，每个手写数字由 28 x 28 = 784 维的向量确定。正因为如此，输入层指定了 784 个神经元。
- 隐藏层
  - 隐藏层可以有多个层，这里仅考虑一个层。深度神经网络就是隐藏层的层数和每层的神经元个数很多，这里的层数可理解为深度，而每层的神经元个数为该层的宽度。
  - 当输入层的神经元给定输入后，每个神经元的值都会加权传递给下一层的所有神经元。

- 记输入层的第 k 个神经元到隐藏层的第 j 个神经元的权重为 w_{jk}. 这种记法的下标有点怪，因为它是从后面层到前面层。这里并没有什么特殊的原因，只是保证矩阵向量乘积的写法 wa 罢了。事实上，设 $a=(a_1,a_2,\cdots,a_n)^T$ 表示输入层神经元对应的数值，则它们对隐藏层的第 j 个神经元的贡献为

  $$\hat{z}_j:=w_{j1}a_1+w_{j2}a_2+\cdots+w_{jn}a_n,\quad j=1,2,\cdots,m. \tag{1}$$

  这样，隐藏层的 m 个神经元获得的贡献可用矩阵向量乘积表示为

  $$\hat{z}=wa \tag{2}$$

  这里的 w 是 m 行 n 列的矩阵，行对应第二层的神经元数，列对应第一层的神经元数。
- 神经元上的输出通常称为**激活值**。当输入层的激活值以加权形式传递给隐藏层后，在隐藏层上还要加上偏置：

  $$z=wa+b, \tag{3}$$

  这里的 $b=(b_1,b_2,\cdots,b_n)^T$，而 b_j 表示隐藏层第 j 个神经元的偏置。z 通常称为**带权输入**。
- 隐藏层的输出，即相应的激活值为 $\sigma(z)=\sigma(wa+b)$，其中的 σ 是激活函数，它的加入导致整个神经网络是高度非线性的。

- 输出层
  - 为了区别，我们记第 l 层的激活值为 a^l，则 a^1 就是神经网络的输入，即手写数字的信息；a^2 是通过加权、偏置和复合得到的；类似 a^3 也是如此。
  - 记第 $l-1$ 层到第 l 层的权重为 w^l，第 l 层上的偏置为 b^l (注意第 1 层没有偏置)，则有第 $l(\geq 2)$ 层的带权输入和激活值分别为

    $$z^l=w^la^{l-1}+b^l,\qquad a^l=\sigma(z^l)=\sigma(w^la^{l-1}+b^l),\quad l=2,3,\cdots,L. \tag{4}$$
  - 最后一层即输出层的激活值为 a^3. **手写数字的输出为 10 维的二进制向量**，例如 $a^3=(0,0,1,0,0,0,0,0,0,0)^T$ 表示数字 2，对应元素 1 的索引位置 (规定从 0 开始)。
  - 注意，神经网络训练后，即确定所有权重和偏置参数后，预测输出向量的每个分量是 0-1 的数值，它表示分类为对应数字的概率。**最大数字的索引就是分类的结果。**

神经网络函数或映射

通过一组已知的输入和输出，确定网络中的参数，即权重和偏置，这个过程称为训练 （实际上就是一种拟合）。训练好的网络就给出了神经网络函数或映射。

激活函数

- 激活函数除了增加网络的非线性外，它的另一个重要作用是实现网络的期望输出。
- 例如，对手写数字识别，我们希望输出 10 维向量，每个位置的值表示对应数字的概率。
- 这样，激活函数要使得函数输出值在 0 到 1 之间。正因为如此，本案例将使用所谓的 sigmoid 函数。
- 要注意，在求解 PDE 的问题中，输出层仅进行线性传递，而不做额外的激活处理。这是因为，PDE 解没有特别的范围限制。若使用 sigmoid 函数激活，则期望输出始终在 0-1之间，不符合要求。