



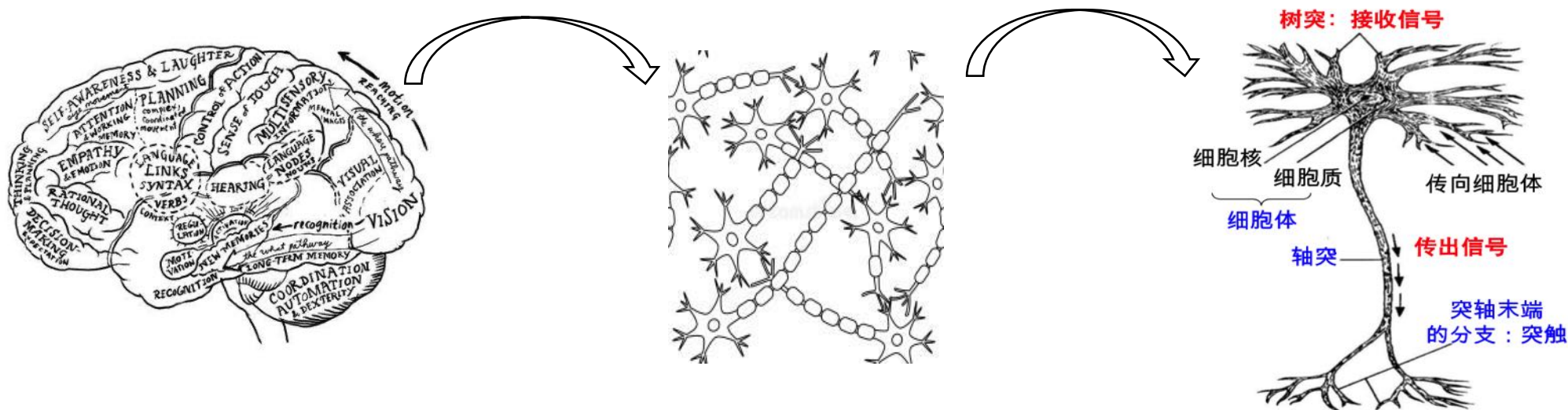
人工智能与信息社会

基于仿生算法的智能系统：神经网络

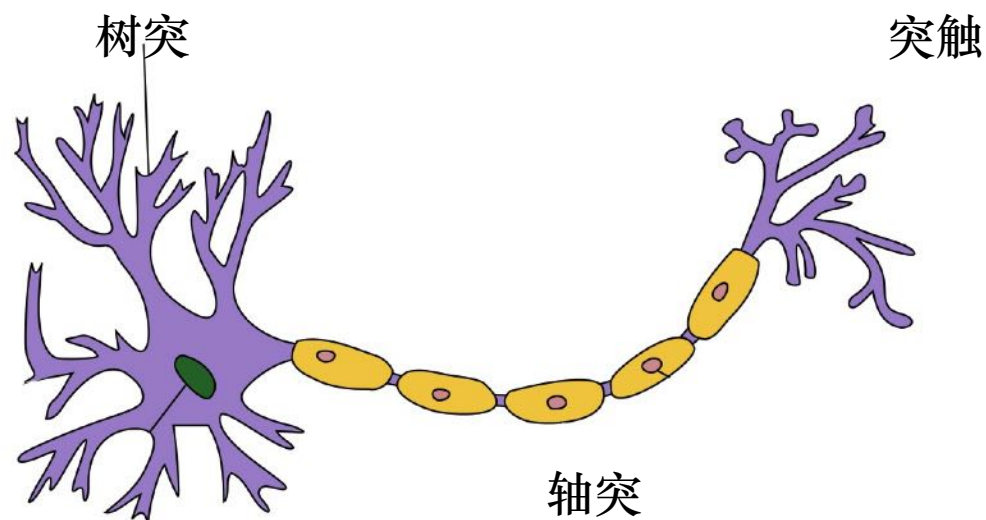
2019.10.21

概述

- 生物原型：人脑以及神经元
- 人脑包含800亿个神经元，这些神经元类似一个小的处理单元，它们按照某种方式连接，接受外部刺激，做出响应处理的过程也就是大脑的对信息处理的过程。

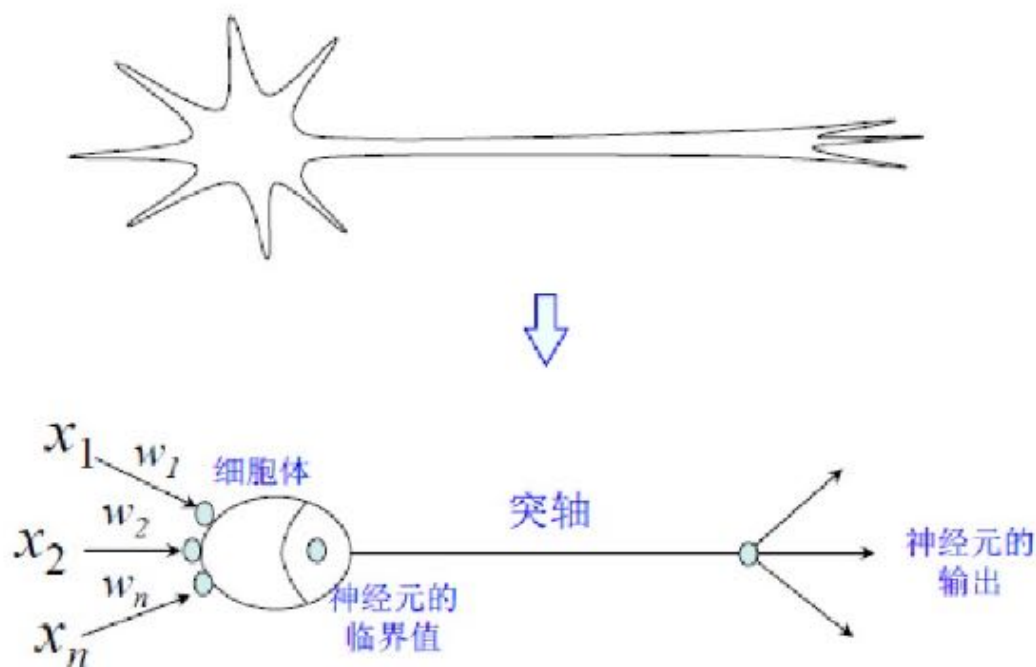


神经元：生物机制



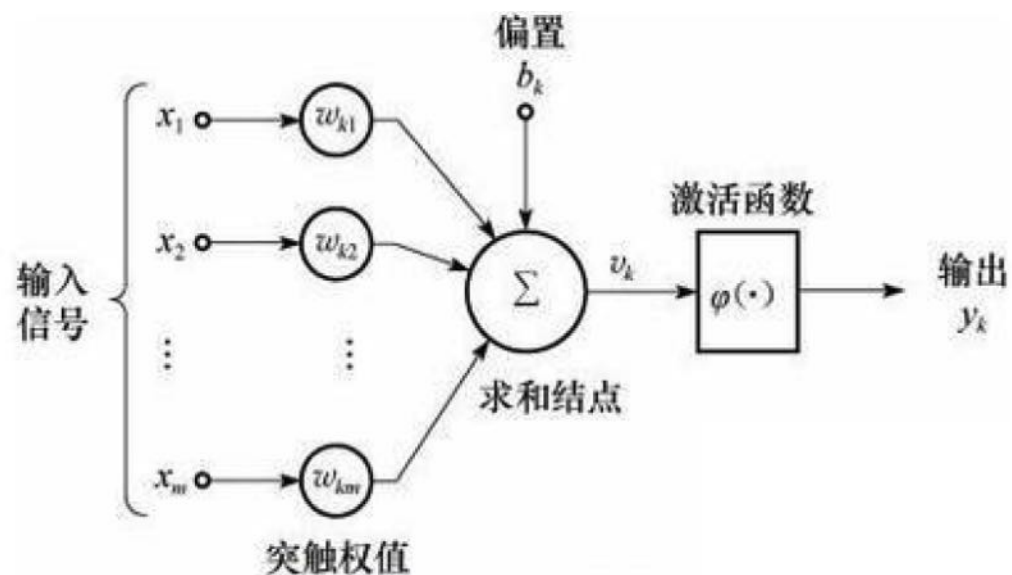
- 树突：为神经元的输入通道，将自其他神经元所接收的电信号传送至细胞本体。
- 轴突和突触：将处理过的信号传递到下一个神经元。

人工神经元：从生物模型到数学模型

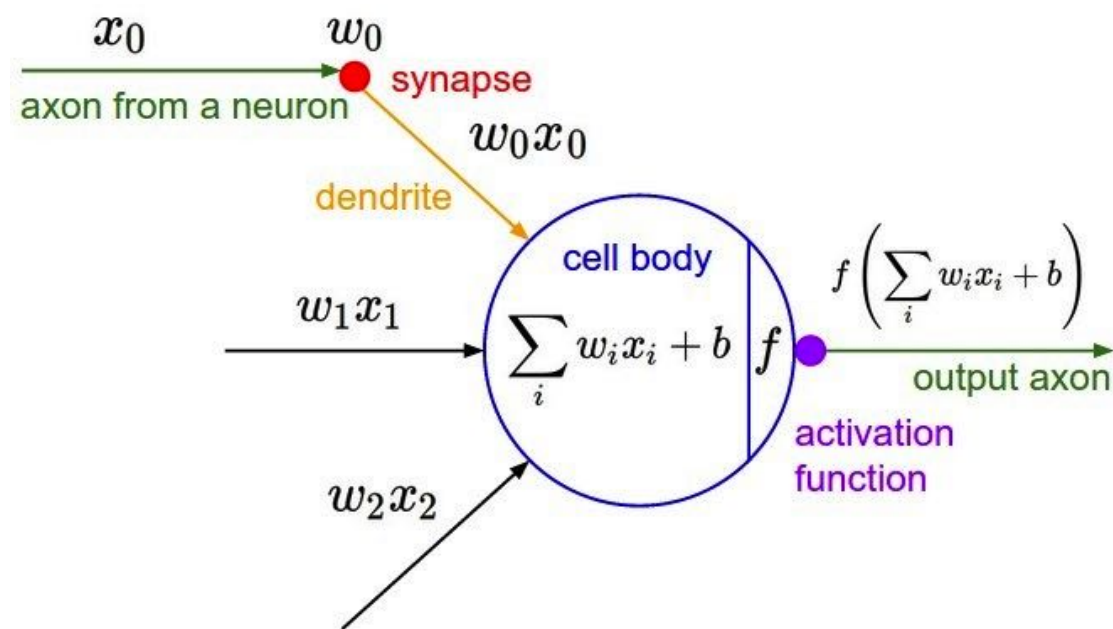


- 输入信号： $x_1, x_2, \dots, x_n$
- 处理单元：
 - 对每个输入信号进行加权处理确定其强度；
 - 对所有输入信号求和确定组合效果；
 - 通过激励函数确定其输出。
- 输出信号： y

人工神经元：从生物模型到数学模型

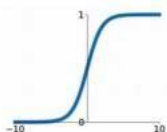


$$y_k = \varphi(\sum_{i=1}^m x_i w_{ki} + b_k)$$



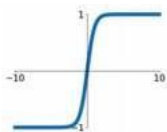
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



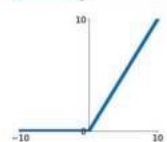
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

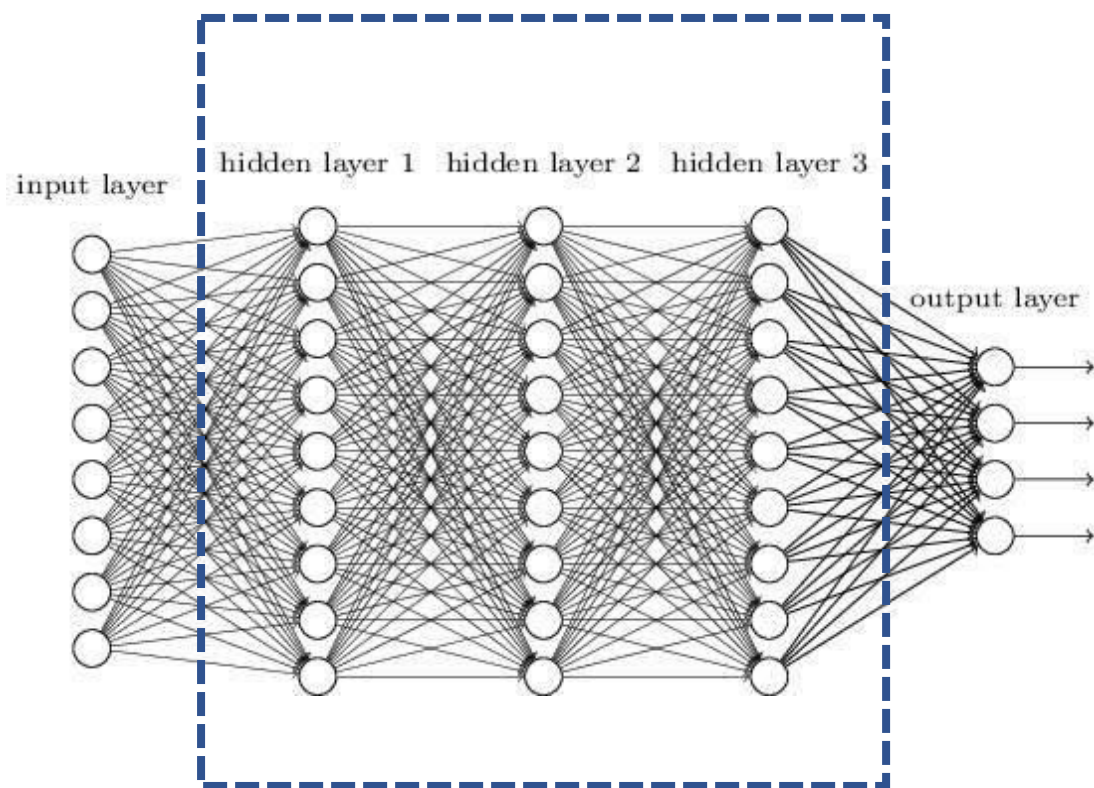
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



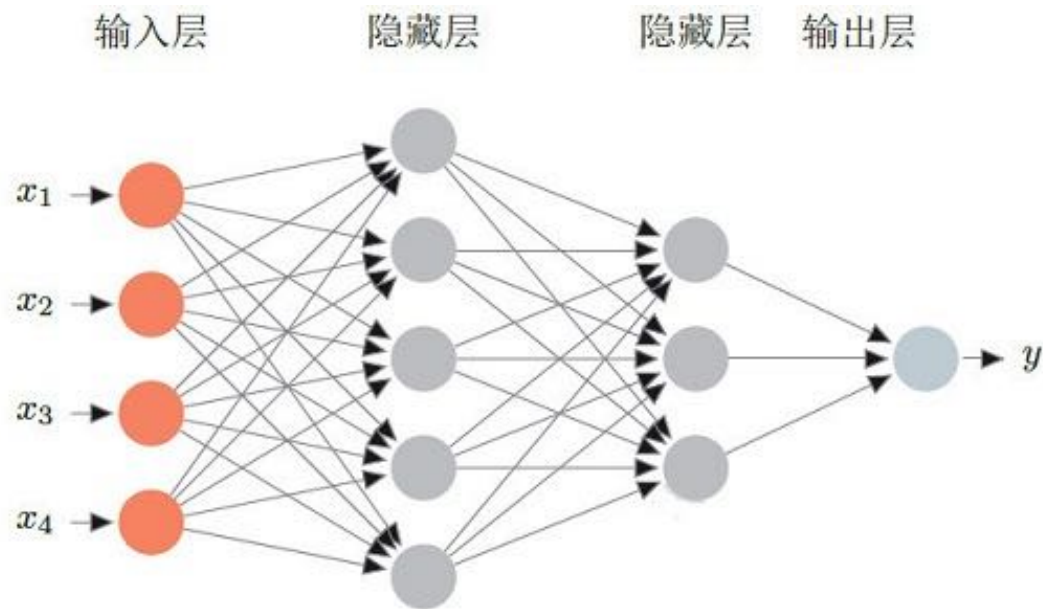
人工神经网络：仿生大脑



- 人工神经网络：模仿神经元在人脑中的结构连接。
- 一个网络可以由几个甚至几百万个人工神经元构成，这些神经元排列在一系列的层中，每个层之间彼此相连。
- 一个完整的神经网络由一层**输入层**、多层**隐藏层**、一层**输出层**构成。

人工神经网络：前馈型神经网络

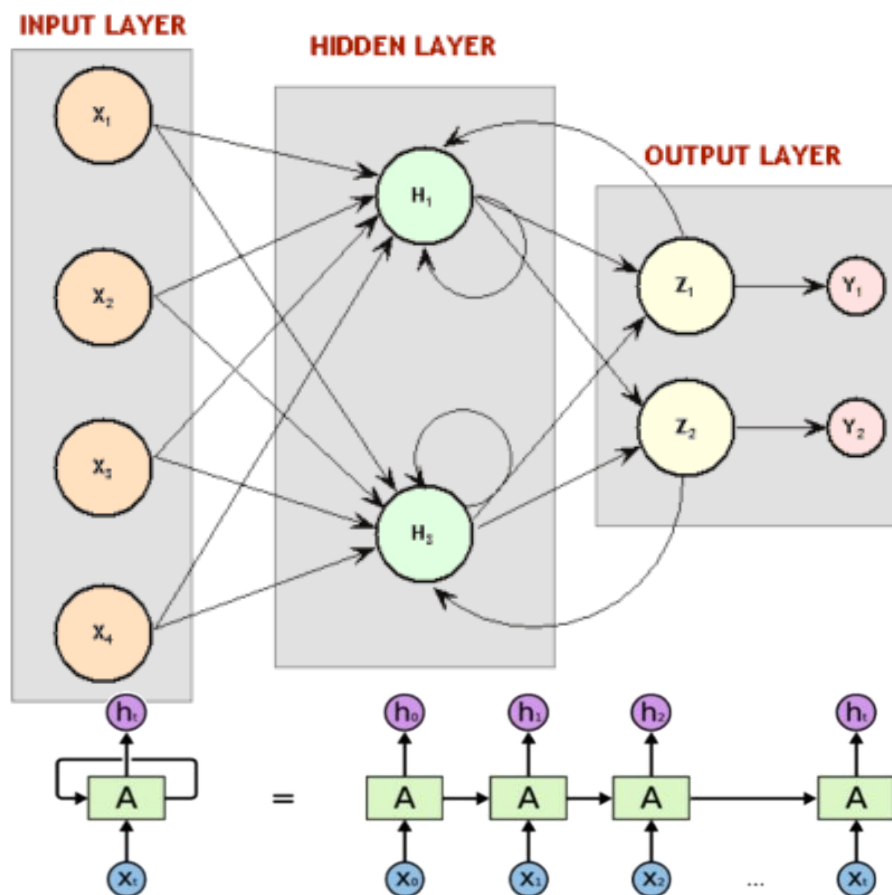
Feed-forward Neural Network



- 单向多层结构
- 即各神经元从**输入层**开始
- 只接收上一层的输出并输出到下一层直至输出层
- 整个网络中**无反馈**。
- 常用于图像识别、检测、分割

人工神经网络： 反馈型神经网络

Feedback Neural Network



- 是一种从输出到输入具有反馈连接的神经网络
- 当前的结果受到先前所有的结果的影响。
- 常用于语音、文本处理、问答系统等



深度学习的工作过程

用训练数据来训练神经网络



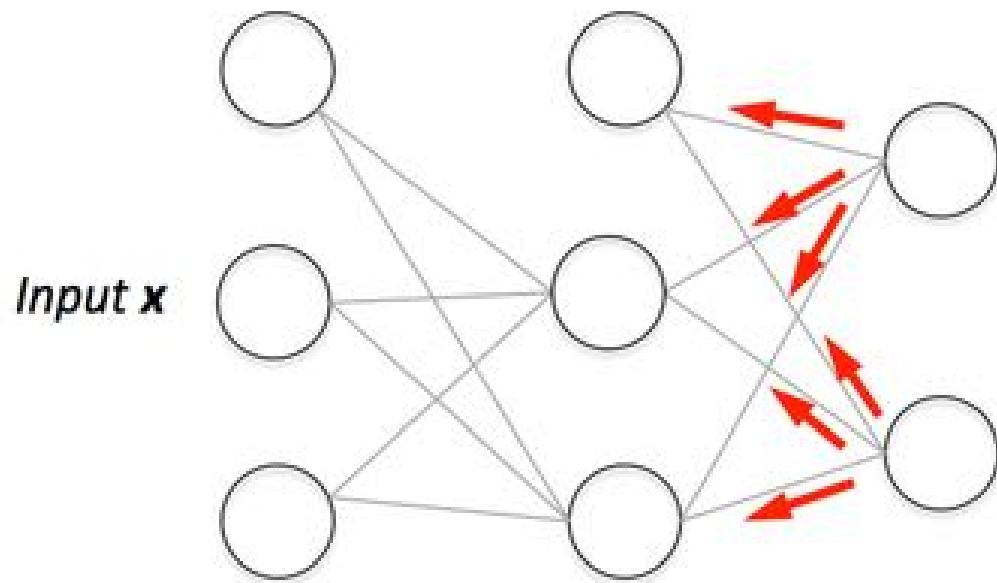
用测试数据来验证神经网络



将神经网络投入实际应用



神经网络训练算法



- 训练目的就是确定所有的参数
 - 每个神经元输入的权重 w 偏置 b
 - 数量巨大
- 以误差为主导的**反向传播**算法
- 其本质是通过前向传递输入信号直至**输出**产生**误差**,
 - 误差的确定有一个“损失函数”
 - 输出值和目标值的差异
- 再将误差信息反向传播去更新网络**权重**矩阵, 减少误差

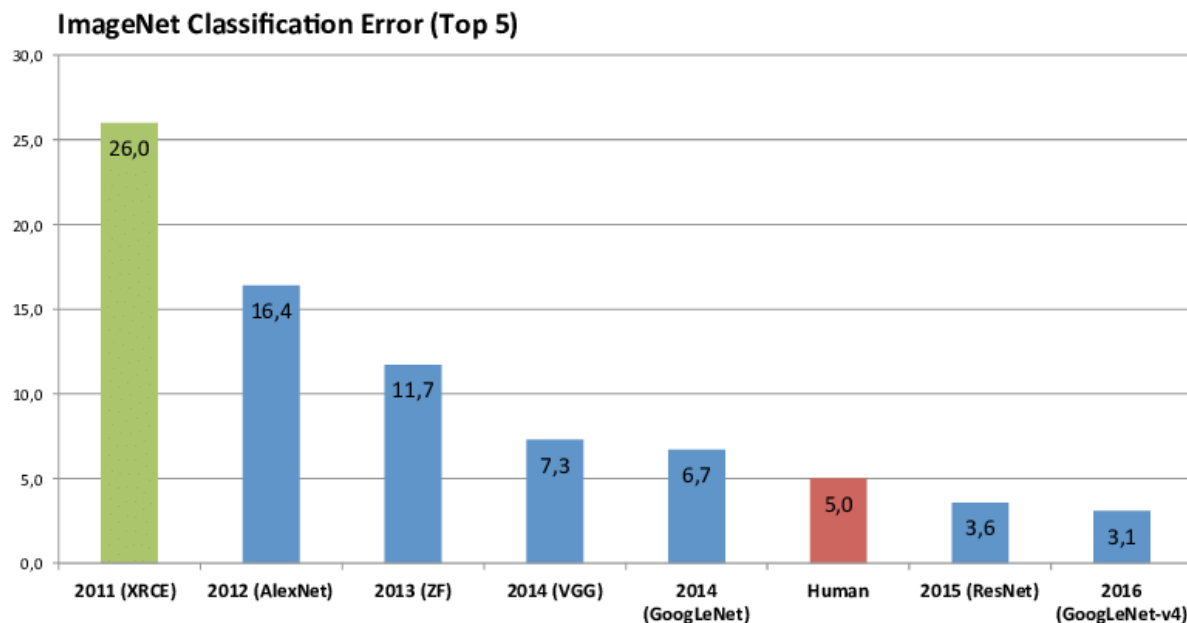
神经网络训练算法



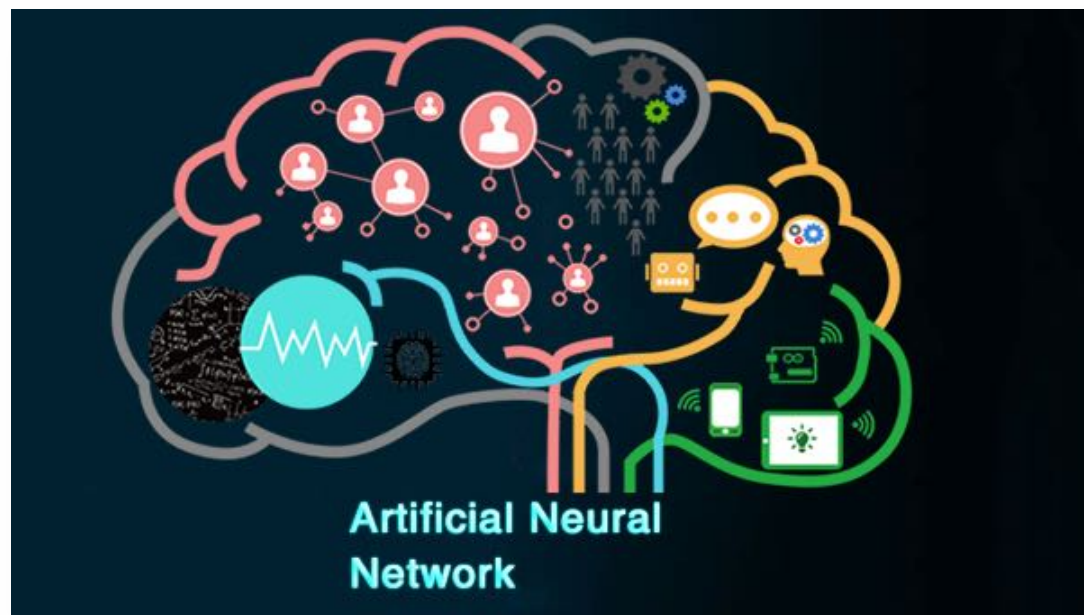
- 通过这种反馈训练机制，反馈越多，神经网络学习的结果越准确。
- 目前基于神经网络对手写体数字识别正确率可达99.8%

神经网络训练算法

- ImageNet图像分类比赛
- 神经网络算法的错误率低至3.1%，人类分类错误率为5%，神经网络分类结果已经优于人类。

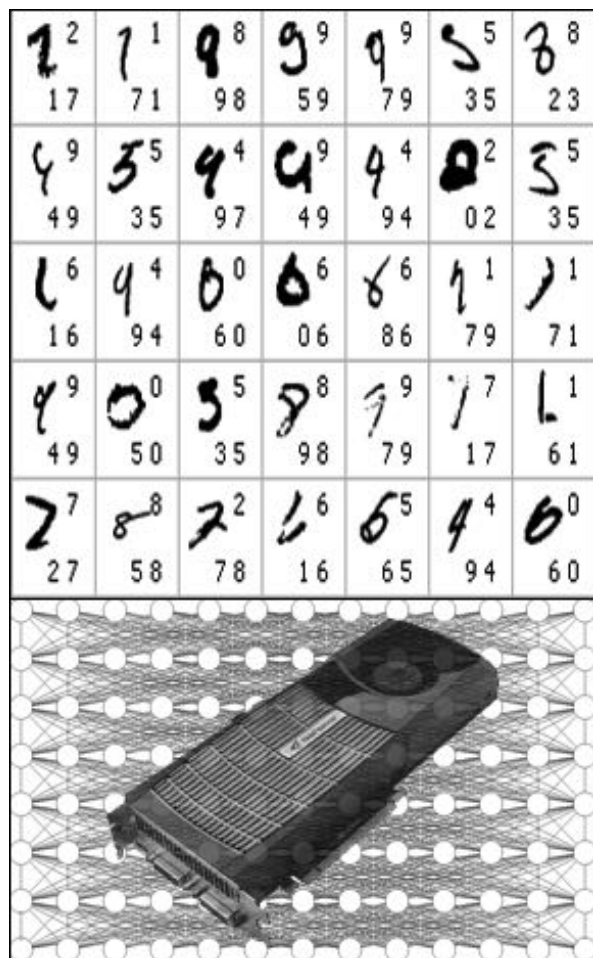


总结



- 人工神经元与人工神经网络模仿人脑的神经结构
- 旨在让计算机拥有更强大的学习能力
- 神经网络的主要学习机制为误差反向传播算法
- 不断修改网络参数使得分类效果越来越好

实例：手写数字识别



- 通过手写数字识别的实例
- 我们可以看到神经网络的强大
- 也可以更好地理解它是如何运行的

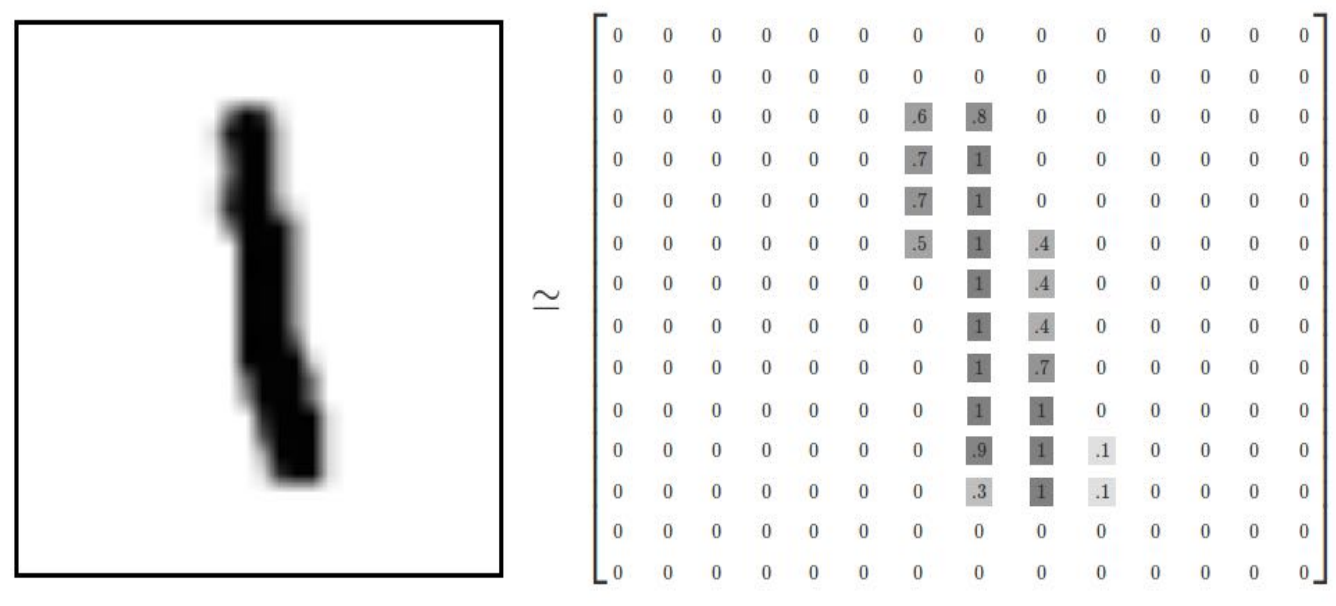
MNIST数据集准备



- MNIST是一个著名的手写体数字识别数据集
 - 训练集包含60,000 个样本和标签，
 - 测试集包含10,000 个样本和标签。
- 训练集类似于我们在学习中使用到的各种学习资料帮助我们提升学习能力；
- 测试集类似于考试试卷，检测我们的学习成绩；
- 标签可以认为是习题和考试的标准答案。

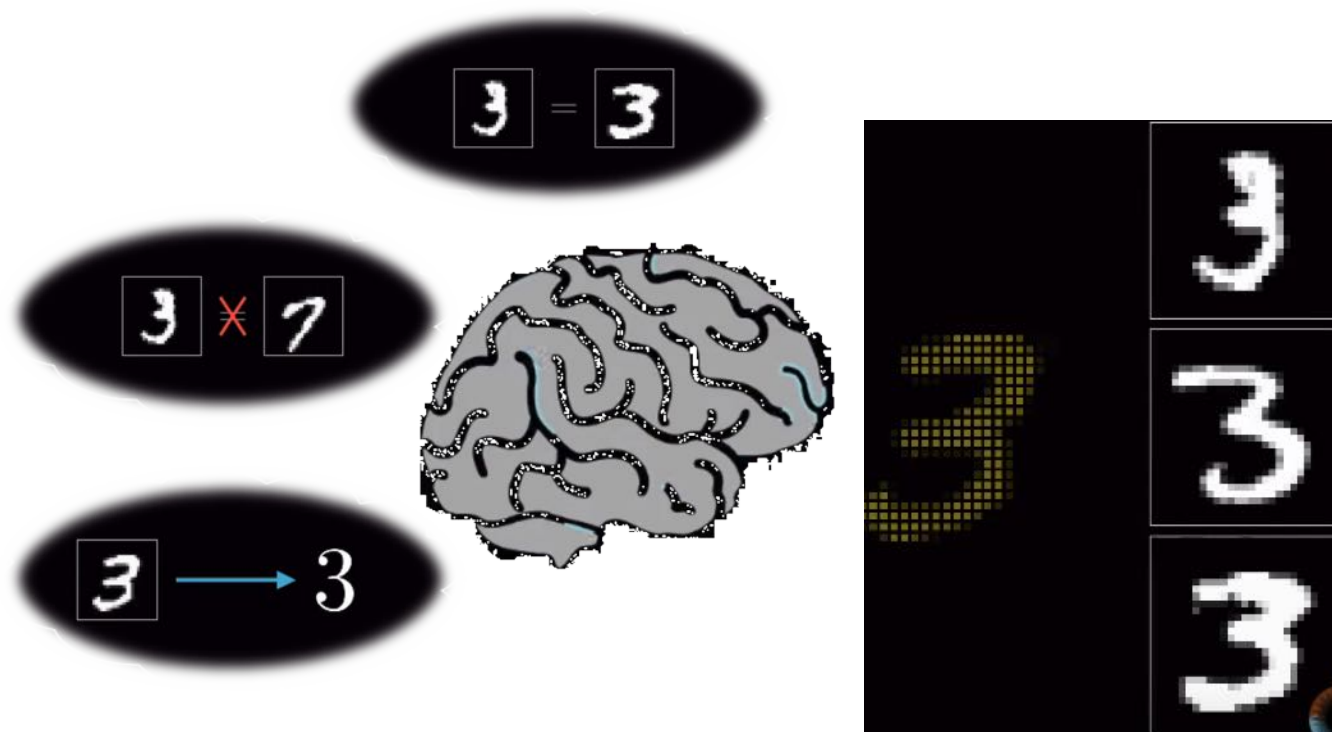
MNIST数据集准备

- 其中样本为代表0~9中的一个数字的灰度图片，对应一个所代表数字的标签，图片大小28*28，且数字出现在图片正中间。



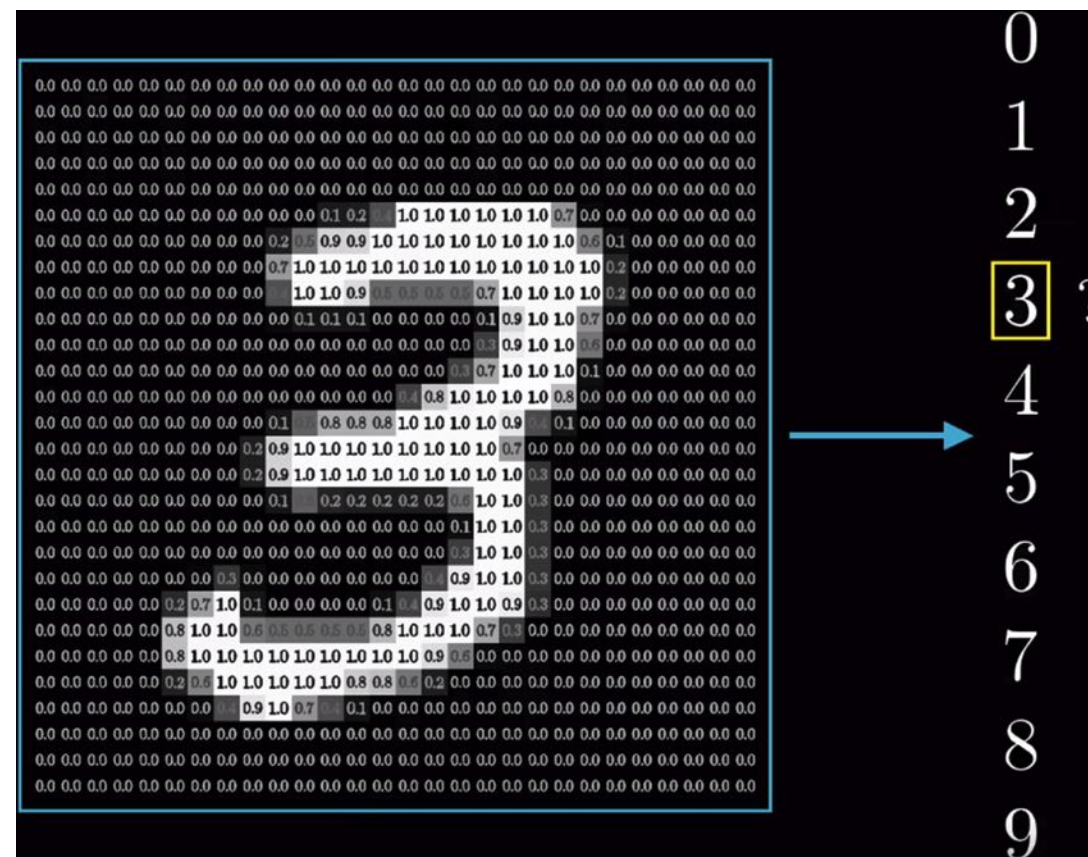
识别数字

- 对于一幅样本图像，人类会根据画面的黑白结构很轻松的判断数字是几。



识别数字

- 电脑“看到”样本图像时其实是得到了一系列的像素点的灰度值数据
- 白的地方是1
- 黑的地方是0。

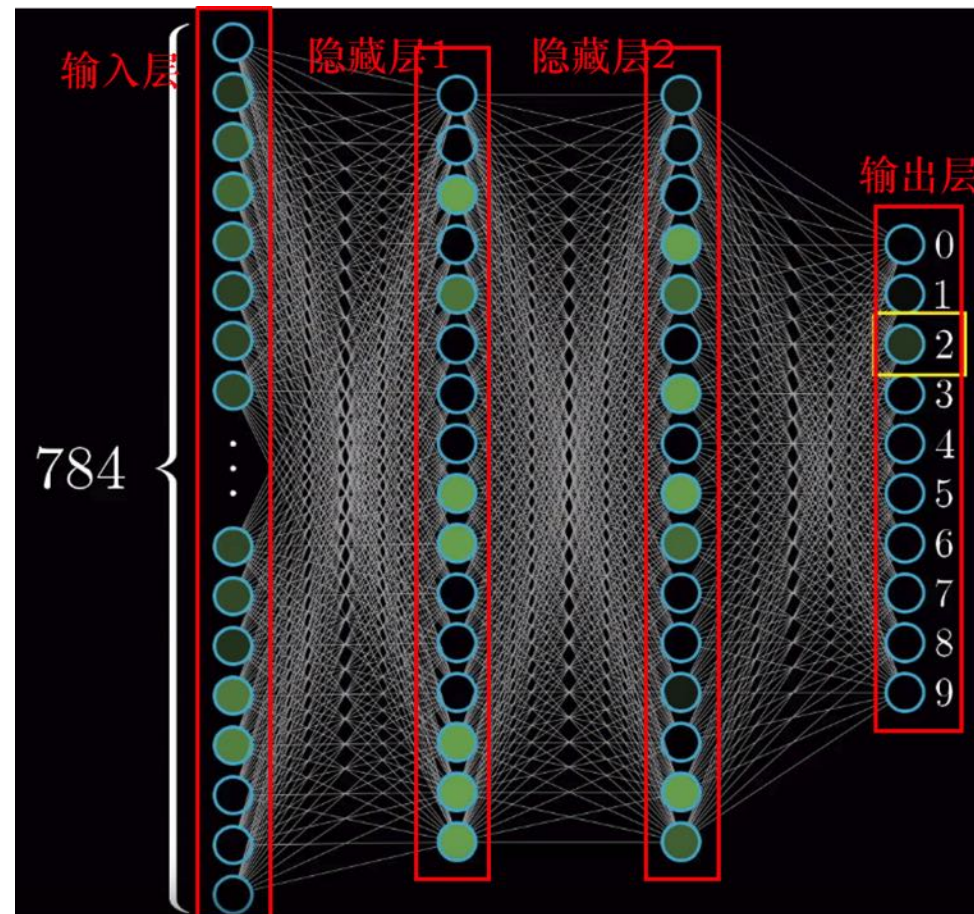


输入输出

- 向量就是多个数字按顺序排成一组，其中数字的个数称为向量的维数。
- 每个样本图像的输入都是一组784个数值
 - 称为一个784维向量， $28*28=784$
- 每个标签数据中，把数字n将表示成一个只有在第n维度数字为1的10维向量。
 - 比如，数字2的标签将表示成：[0,0,1,0,0,0,0,0,0,0]
 - 数字5的标签将表示成：[0,0,0,0,0,1,0,0,0,0]

输入输出

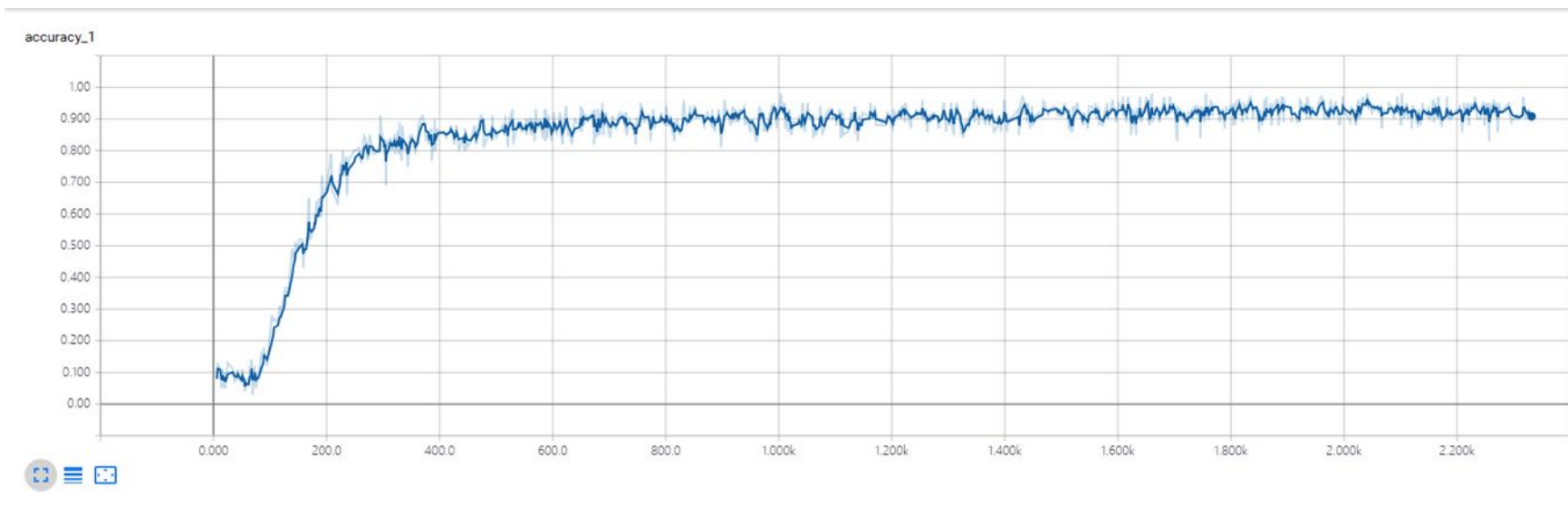
- 输入的784维向量经过两层中间计算后
- 输出为一个10维的向量
- 每个维的值代表是该数字的概率。



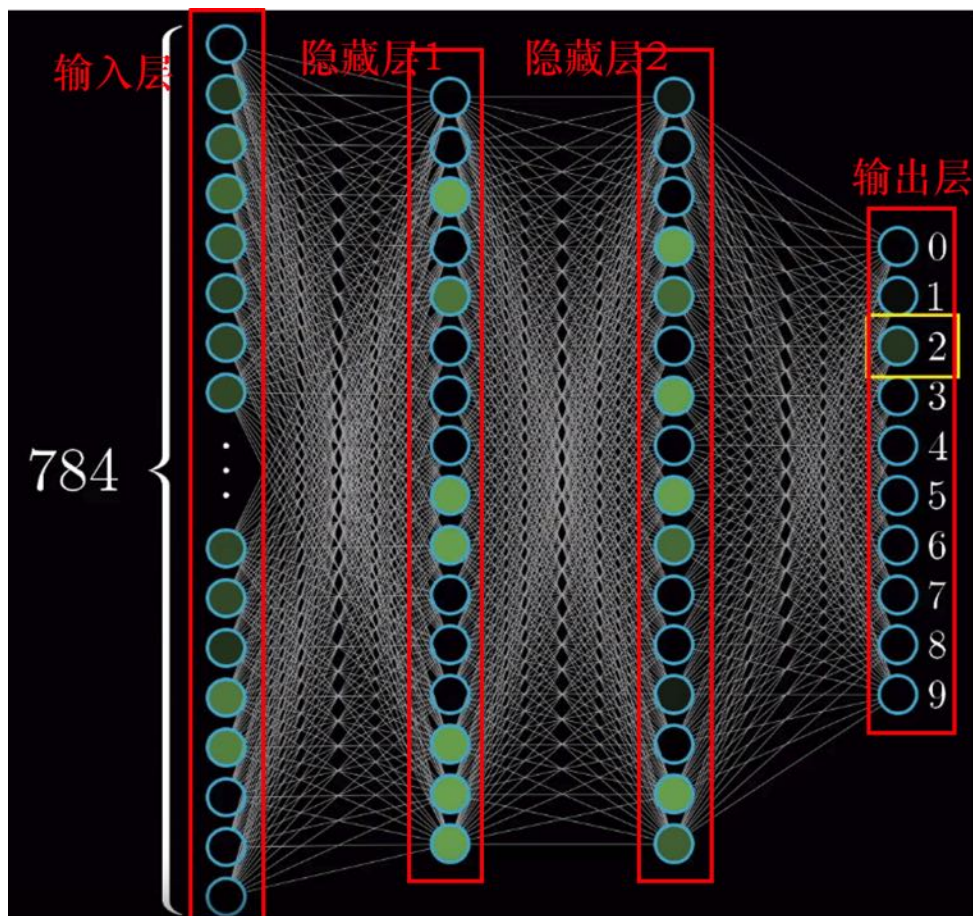
识别结果

- 经过一小段时间的训练，AI识别手写数字的准确率已经达到了93.13%

```
Accuracy at step 0: 0.0819
Accuracy at step 128: 0.3761
Accuracy at step 256: 0.7901
Accuracy at step 384: 0.8468
Accuracy at step 512: 0.8702
Accuracy at step 640: 0.886
Accuracy at step 768: 0.8964
Accuracy at step 896: 0.9032
Accuracy at step 1024: 0.9078
Accuracy at step 1280: 0.9111
Accuracy at step 1408: 0.9116
Accuracy at step 1536: 0.9141
Accuracy at step 1664: 0.9194
Accuracy at step 1792: 0.9227
Accuracy at step 1920: 0.9265
Accuracy at step 2048: 0.9275
Accuracy at step 2176: 0.9284
Accuracy at step 2304: 0.9313
```



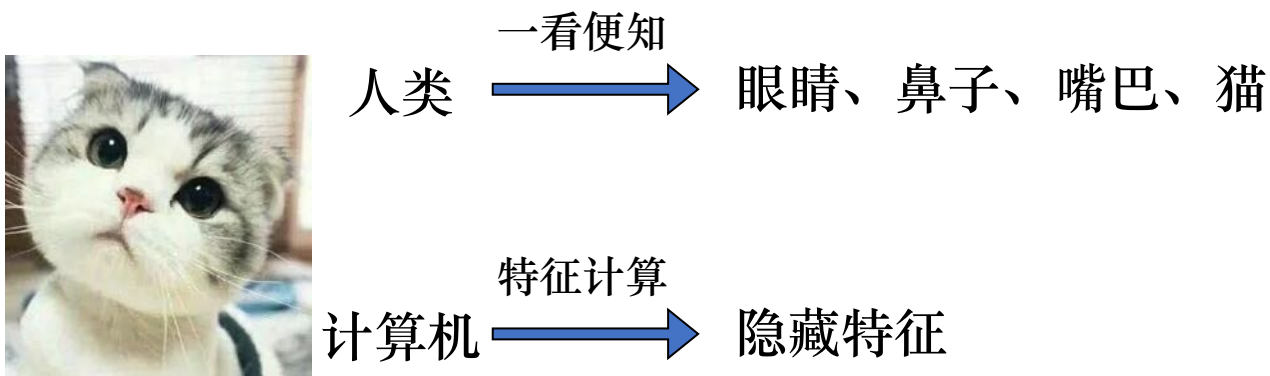
神经网络构成



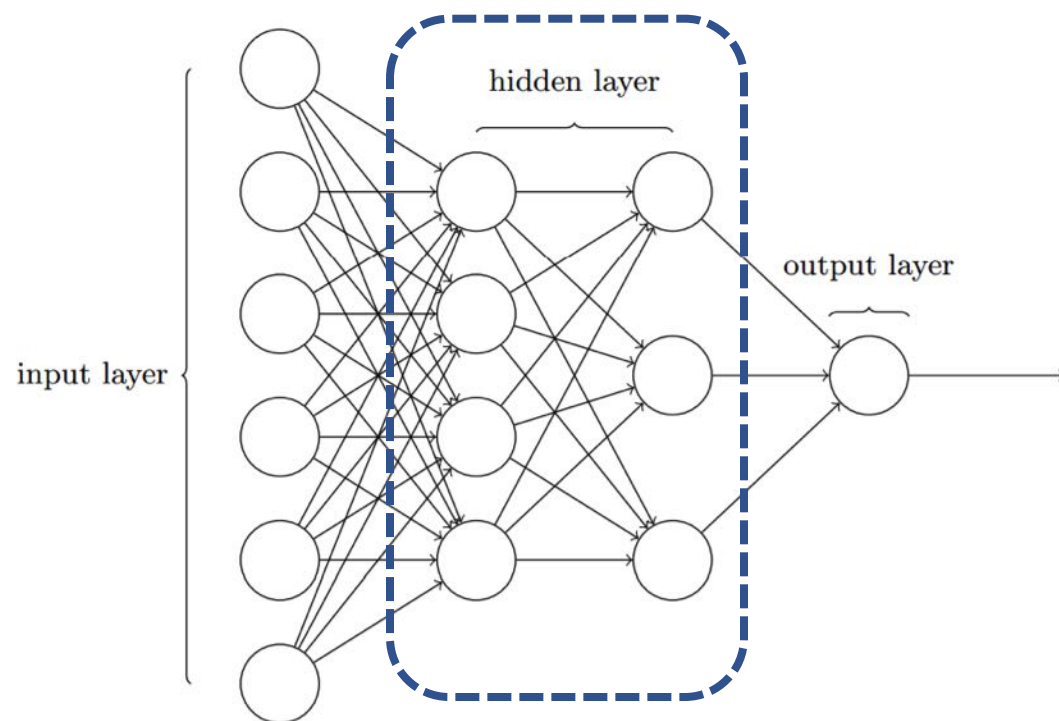
- 1个输入层、1个输出层
- 若干个隐藏层
- 隐藏层和外部世界没有直接联系
- 相当于人类大脑中提取特征各个神经元层次。

隐藏层

- 对于人类而言，我们看一眼图片就能够在大脑中形成图片的各种特征，从而对图片做出正确的判断。
- 对于计算机而言，要从一串数据中提取特征，就要依靠不同的隐藏层来实现。



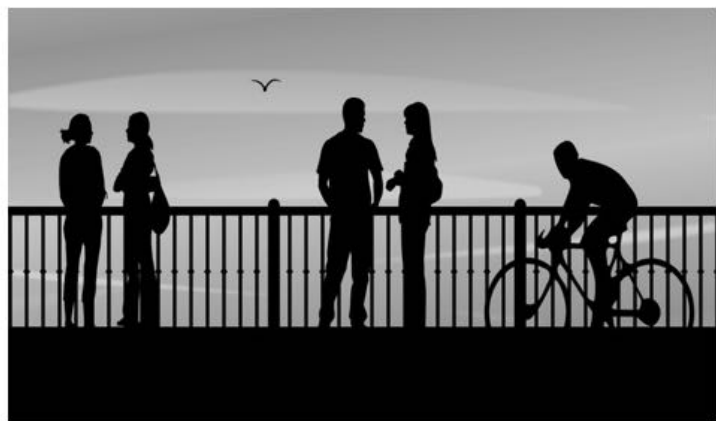
隐藏层



- 隐藏层有各种类型：
 - 卷积层、全连接层、池化层、归一化指数层、激活层等。
- 通过多个顺序链接的隐藏层的组合
- 神经网络就可以将原始图像变换为高层次抽象的图像特征，
- 从而能够“认识”图像中的东西。

卷积层

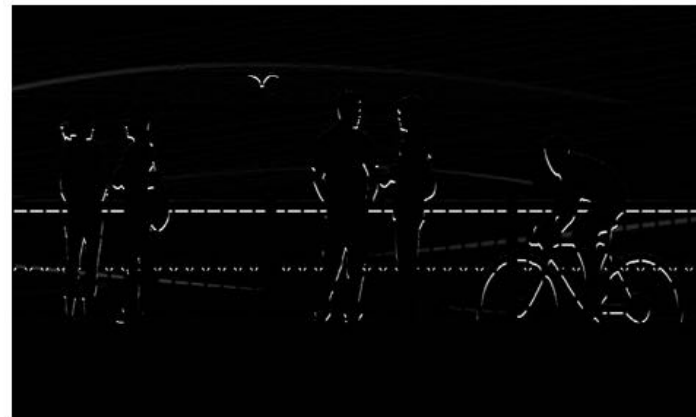
- 卷积层的作用是提取图像的二维特征，通过不同的算子可以检测图像不同边缘。



原始图像

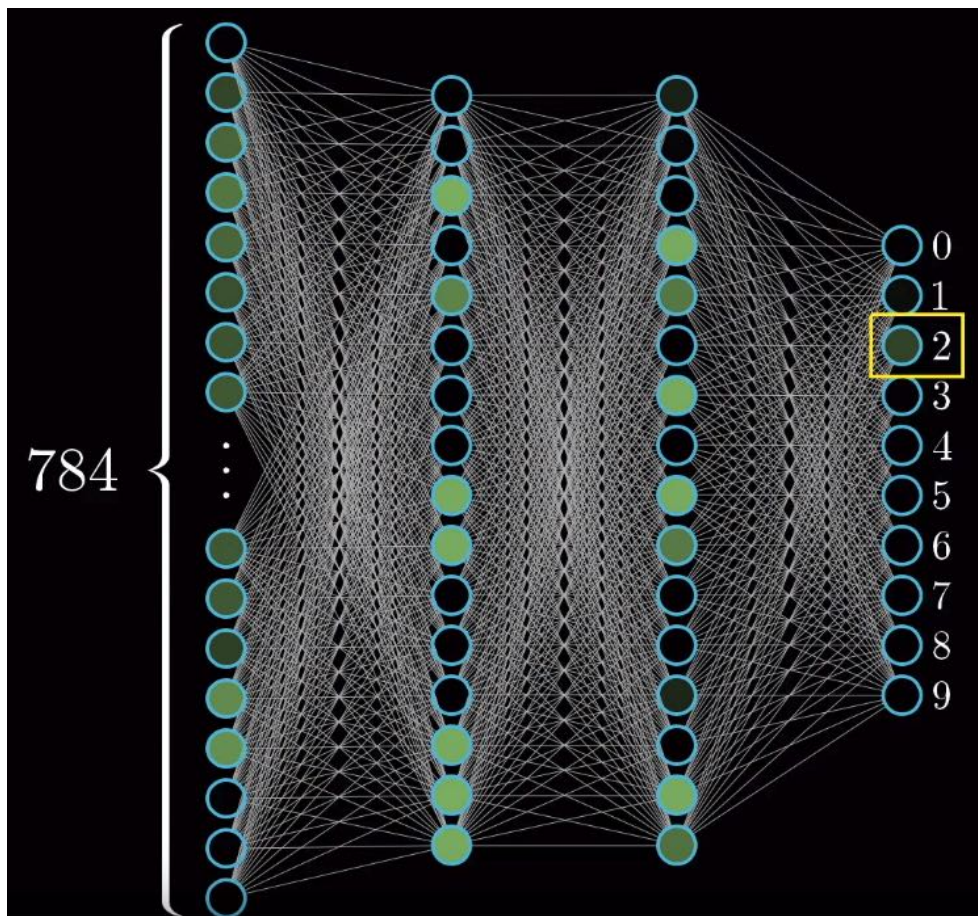


垂直边缘



水平边缘

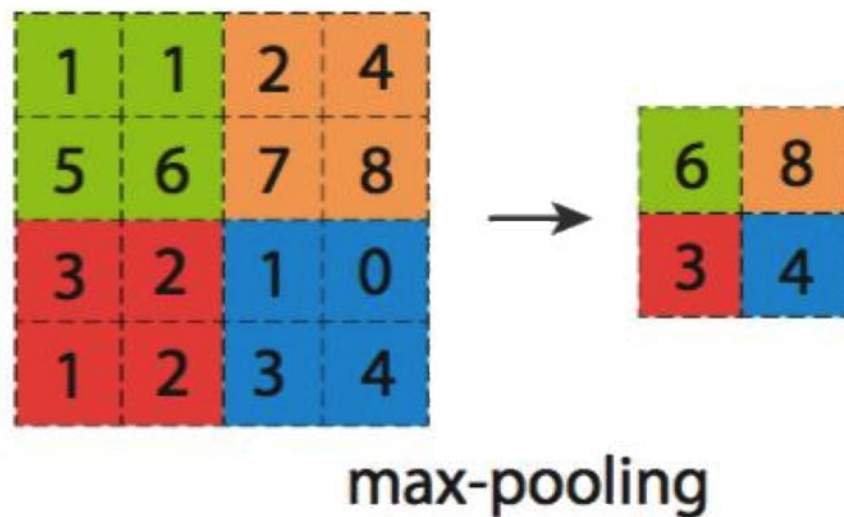
全连接层



- 全连接层的作用是将所有特征融合到一起。
 - 通常用来把计算得到的蕴含特征的大维度向量
 - 转换成与输出层向量的维度相同。
- 顾名思义，全连接就是每个神经元节点都与下一层连接。

池化层

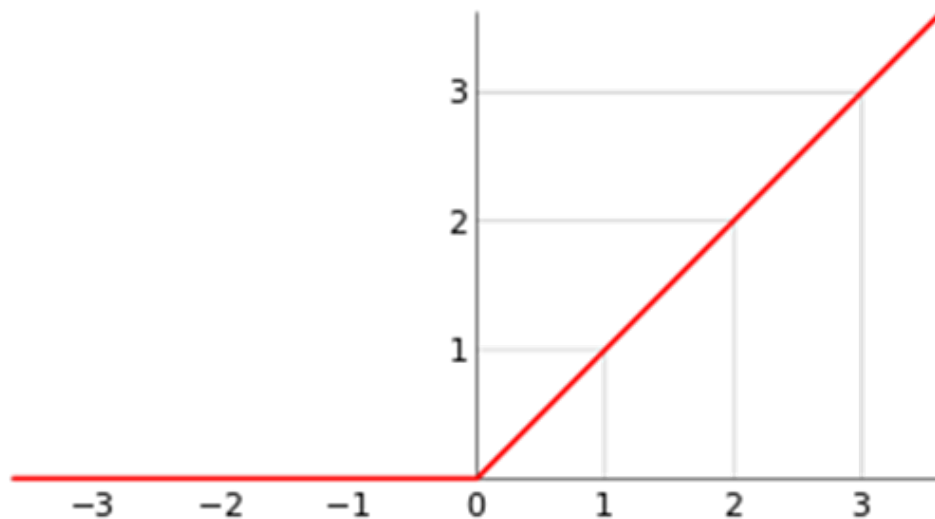
- 池化层作用是**减少**训练参数，是对原始特征信号进行采样。
- 当输入数据过大时，卷积层的计算量就会很大，这时需要减少参数，因此池化层常出现在卷积层之后。



激活层

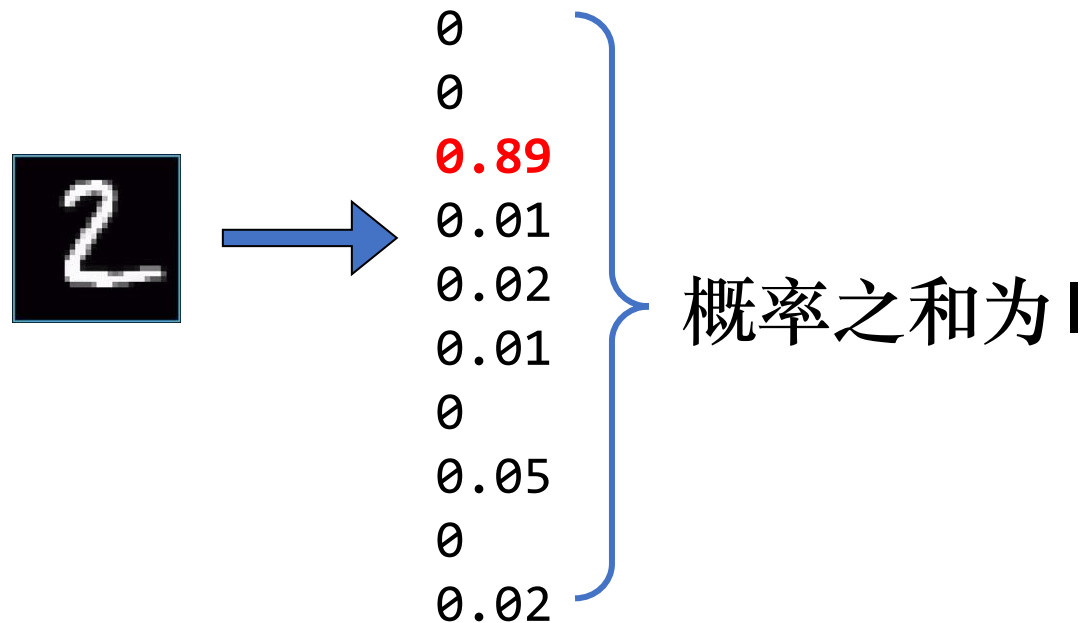
- 神经元有两种状态：**兴奋**和**抑制**。事实上处于不同程度“兴奋”的神经元传播化学物质也不尽相同。
- 非线性激活层就是决定哪些神经元的**活跃程度高**，哪些神经元的**活跃程度低**。

ReLU线性整流函数



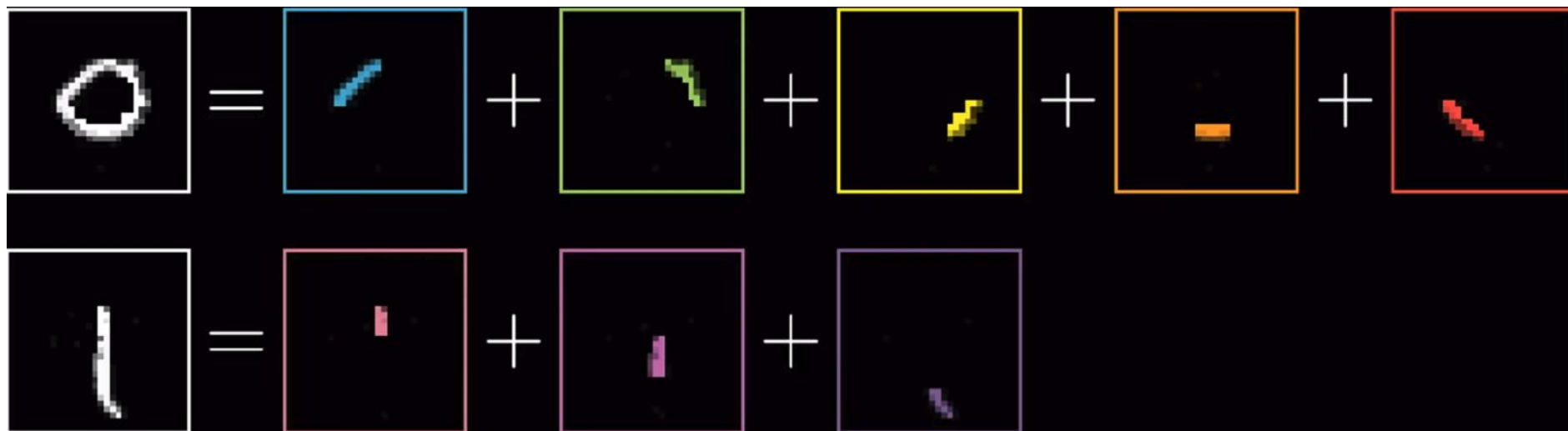
归一化指数层

- 归一化指数层的作用就是完成最后输出分类时每个类别概率的计算。
- 通过“**归一化**”，使得最后输出的10维向量的每个值加起来总和为1。
- 每个单独的数值就可以理解为AI认为图像是该分类的概率。



特征提取

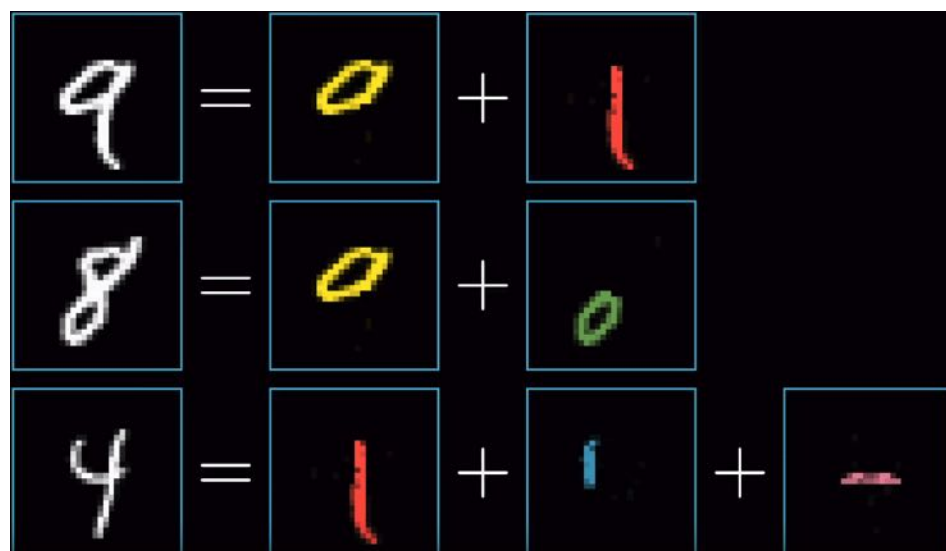
- 本次实例中采用2层隐藏层来对图像特征进行提取。
- 每一层以**前一**层提取的特征作为输入，可以得到更加复杂的特征。



隐藏层I，小结构特征

特征提取

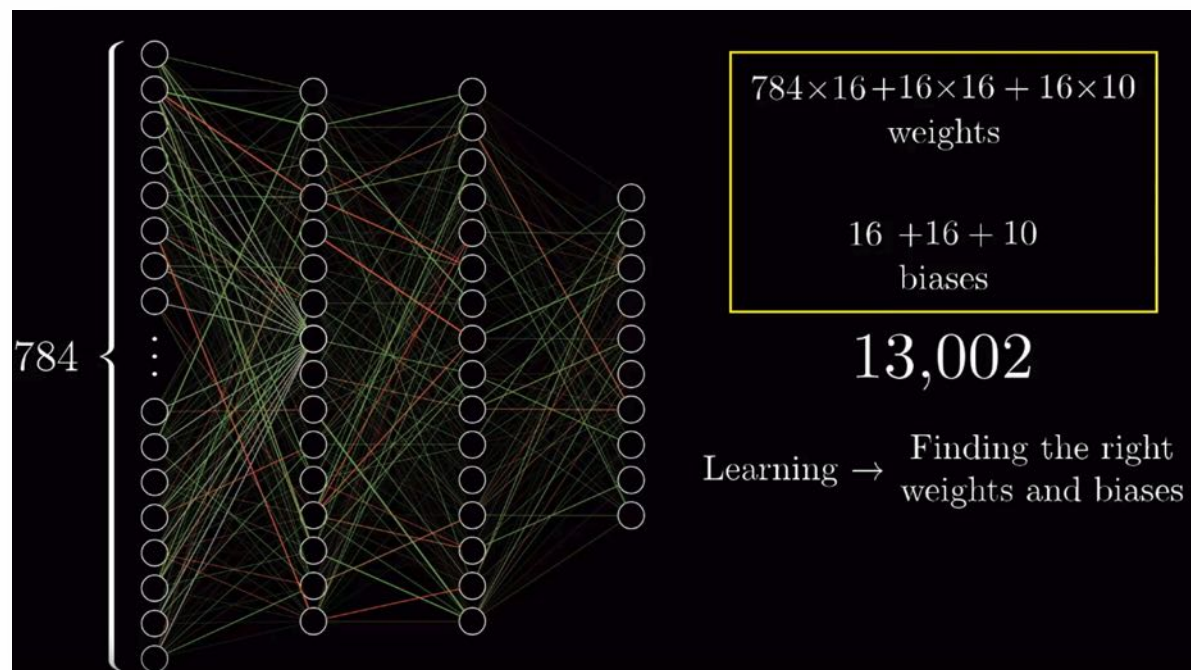
- 本次实例中采用2层隐藏层来对图像特征进行提取。
- 每一层以**前一层**提取的特征作为输入，可以得到更加复杂的特征。



隐藏层2，大结构特征

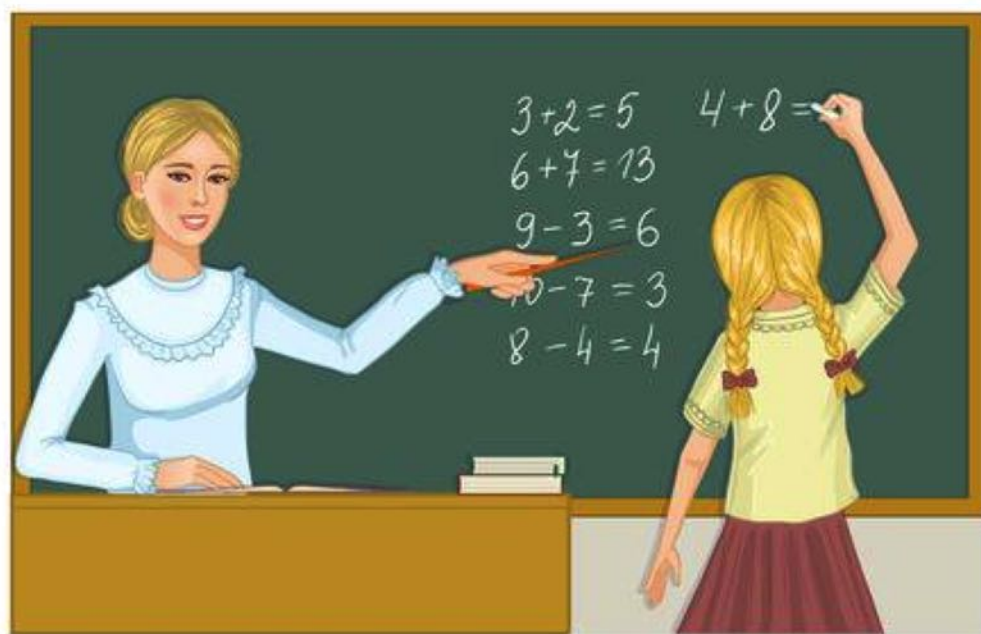
人工神经网络的训练

- 在介绍了神经网络的主要模型并搭建完一个人工神经网络之后，我们就要对它进行**训练**。
- 对于一个2层的小网络，就可能有13,000多个参数需要我们通过训练来求解。

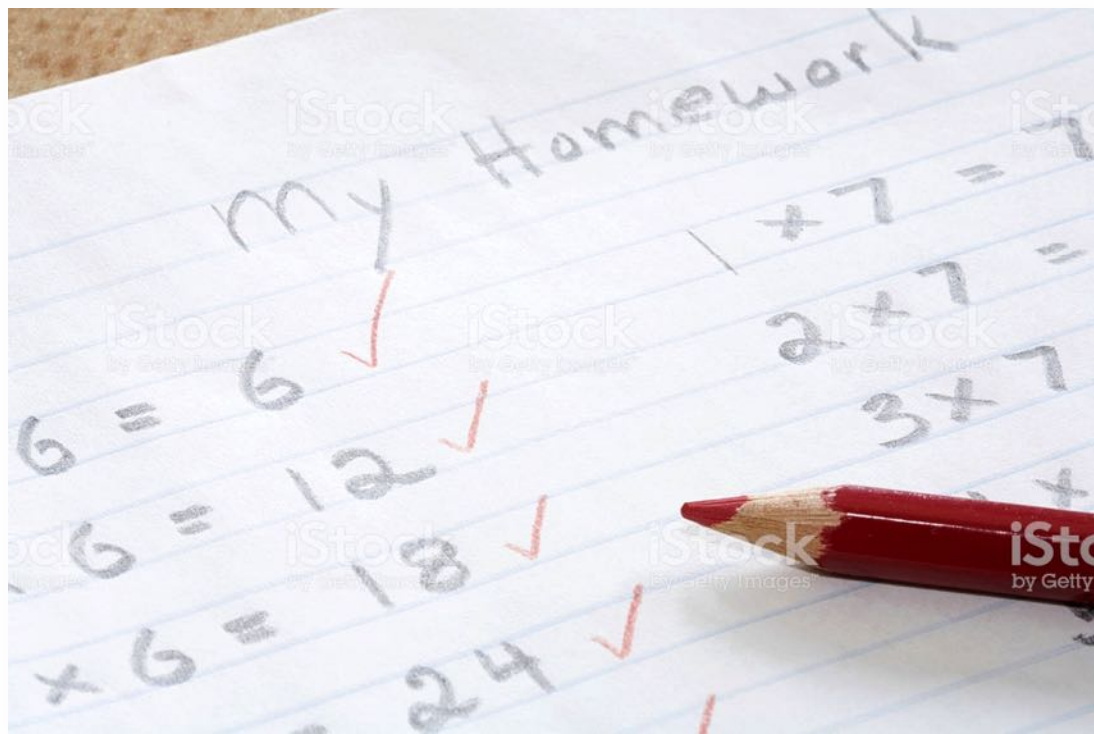


人工神经网络的训练

- 我们对于网络的训练是通过一系列带有**标记**的**样本数据**来进行的，这种学习方式称为监督学习。



人工神经网络的训练



- 既然样本数据带有标记，我们就要把神经网络预测得到的结果与标记进行**对比**，从而判断预测的准确性。
- 如果发现预测结果不够好，那么我们将**调整**网络的参数，使得网络能够做出更好的预测。
- 如何评价预测结果好坏？通过**损失函数**来定量计算。

损失函数

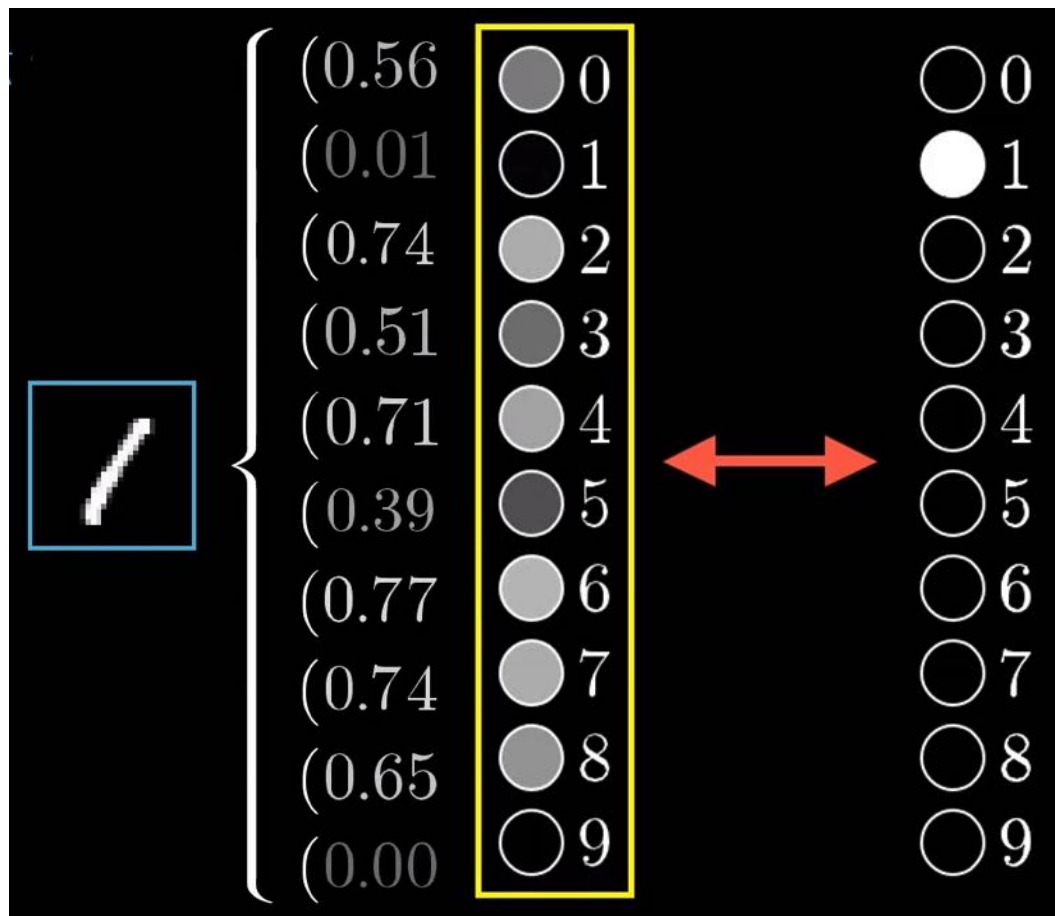
- 损失函数 (loss function) 又叫做代价函数 (cost function) , 是用来评估模型的预测值与真实值的误差大小。
- 神经网络训练的过程就是**最小化**损失函数的过程,
- 通过不断优化网络参数, 使模型的预测值**更加接近**真实值。

损失函数：0-1 损失函数

- 0-1 损失函数：一种常用于分类任务的损失函数。
 - 预测值与真实值不同，就是预测错误，则损失是1；
 - 预测值与真实值相等，就是预测正确，损失是0，就是没有损失。

$$l_i = \begin{cases} 0 & \text{如果}\vec{x}\text{不属于第}i\text{类} \\ 1 & \text{如果}\vec{x}\text{属于第}i\text{类} \end{cases}$$

损失函数



- 在本实例中，不同的数字有不同的**概率**，不是非0即1
- 不适用于0-1损失函数

损失函数：平方损失函数

- 平方损失函数：预测值与真实值的差的平方和。
- 平方损失函数比0-1损失函数能够反映多的信息，在手写数字识别实例中更有利于参数调整。

$$L = (Y - f(x))^2$$

Y : 真实值（标签）

$f(x)$: 预测值

损失函数

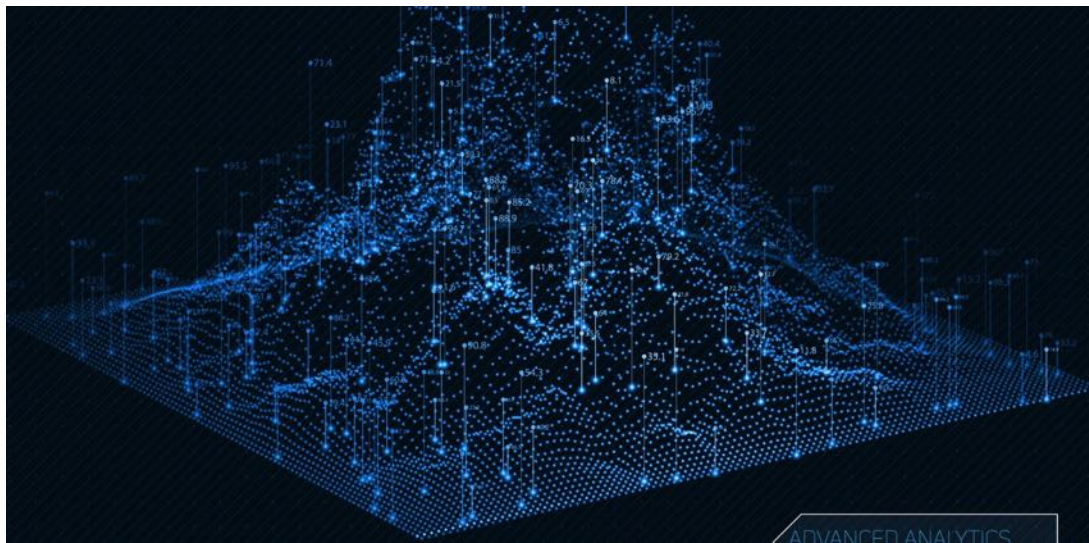
- 通过不断减小平方损失函数来提高分类的准确率。


$$\left\{ \begin{array}{l} (0.56 - 0.00)^2 + \\ (0.01 - 1.00)^2 + \\ (0.74 - 0.00)^2 + \\ (0.51 - 0.00)^2 + \\ (0.71 - 0.00)^2 + \\ (0.39 - 0.00)^2 + \\ (0.77 - 0.00)^2 + \\ (0.74 - 0.00)^2 + \\ (0.65 - 0.00)^2 + \\ (0.00 - 0.00)^2 \end{array} \right.$$

总结

- 通过**损失函数**来定量计算神经网络**预测结果的好坏**。
- 神经网络训练的过程就是**最小化**损失函数的过程
- 损失函数**越小**说明预测结果**越接近**真实结果（标记）

如何优化参数



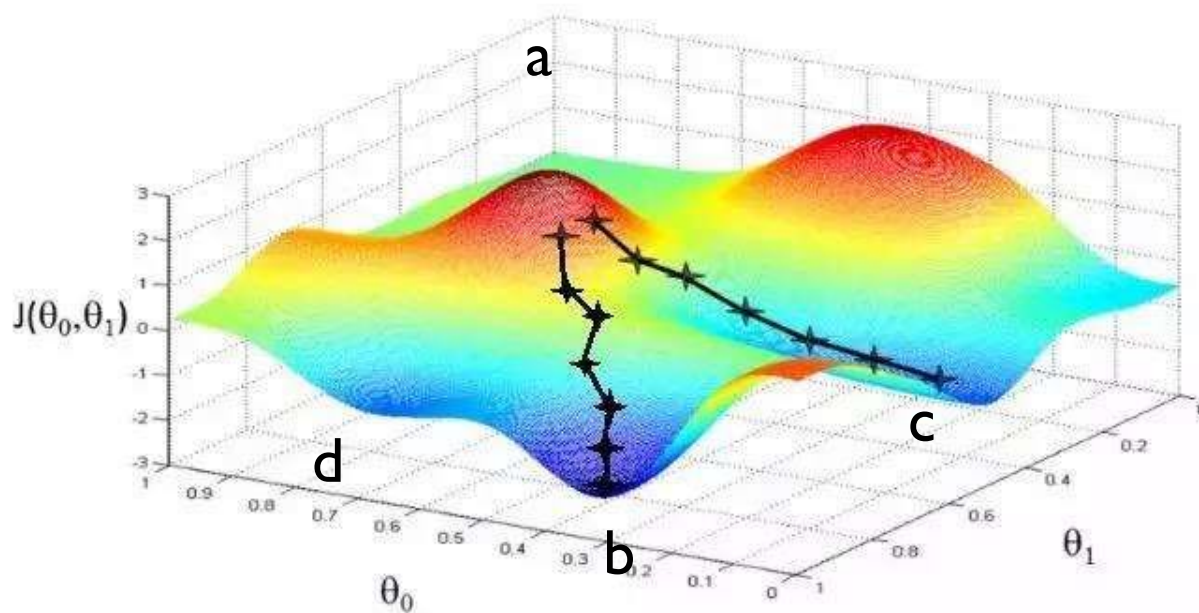
- 我们已经知道通过损失函数能够定量神经网络预测结果的好坏
- 那我们如何根据这个评价结果来**更改优化**我们的参数？
- 神经网络中的参数是海量的，如果修改的方法不当，就会耗费大量的计算时间。

优化器

- 在损失函数确定情况下，优化网络的任务就交给了优化器。
- 优化器代表了调整网络参数，使损失函数达到最小的过程。
- 不同的优化器有不同的算法，改变损失函数的过程和效率也各不相同。

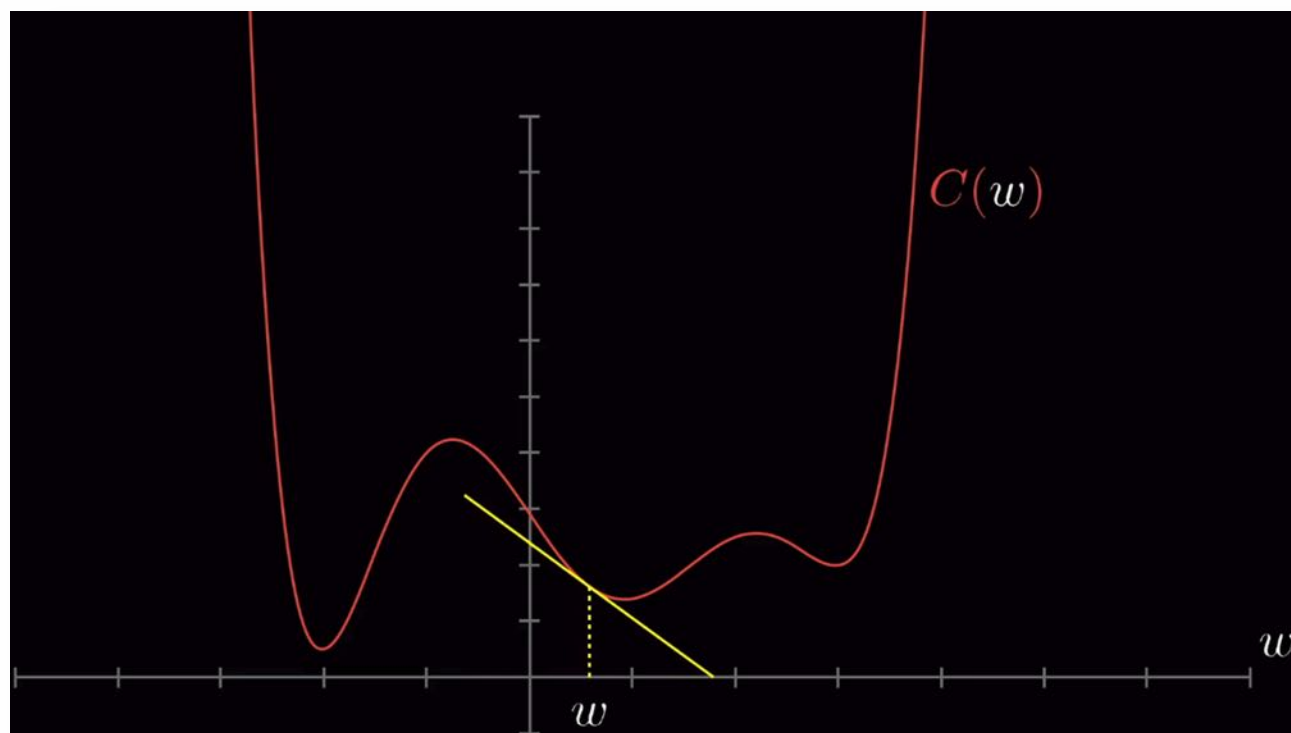
函数优化例子

- 下图曲面表示具有两个参数的损失函数，可以把它想像成一个起伏的山谷。
- 假设小明现在处于图中最高点a，要去到最低点b，那该如何选择最优路径？



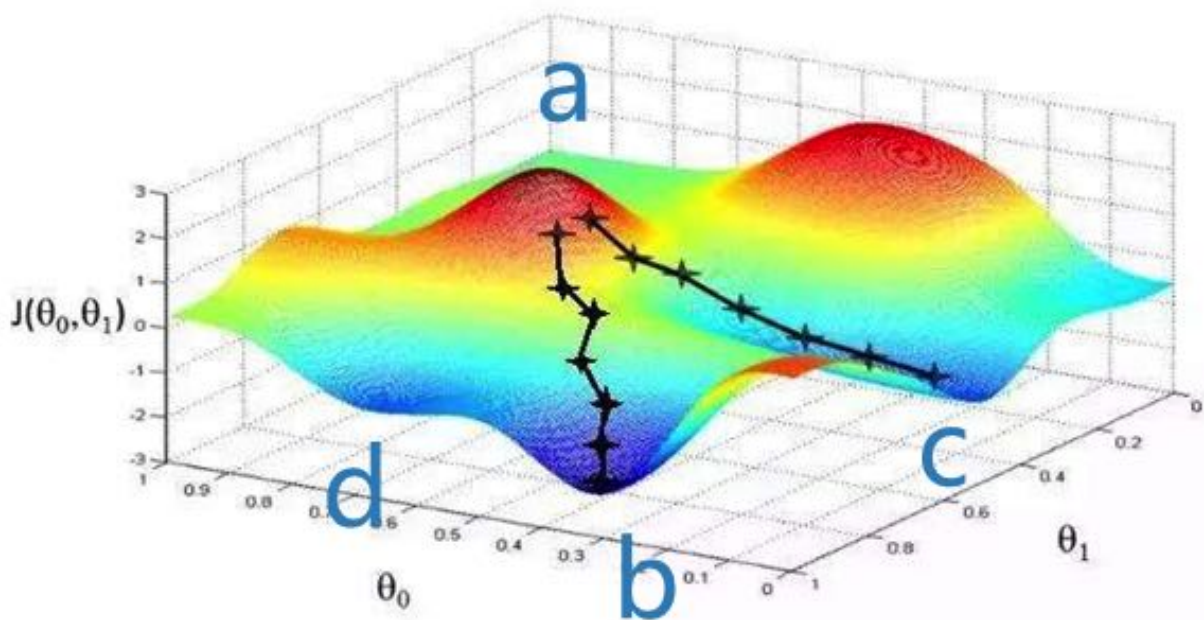
梯度

- 表示函数在某一点变化最快的方向
- 图中 w 点的梯度方向就是黄色的切线



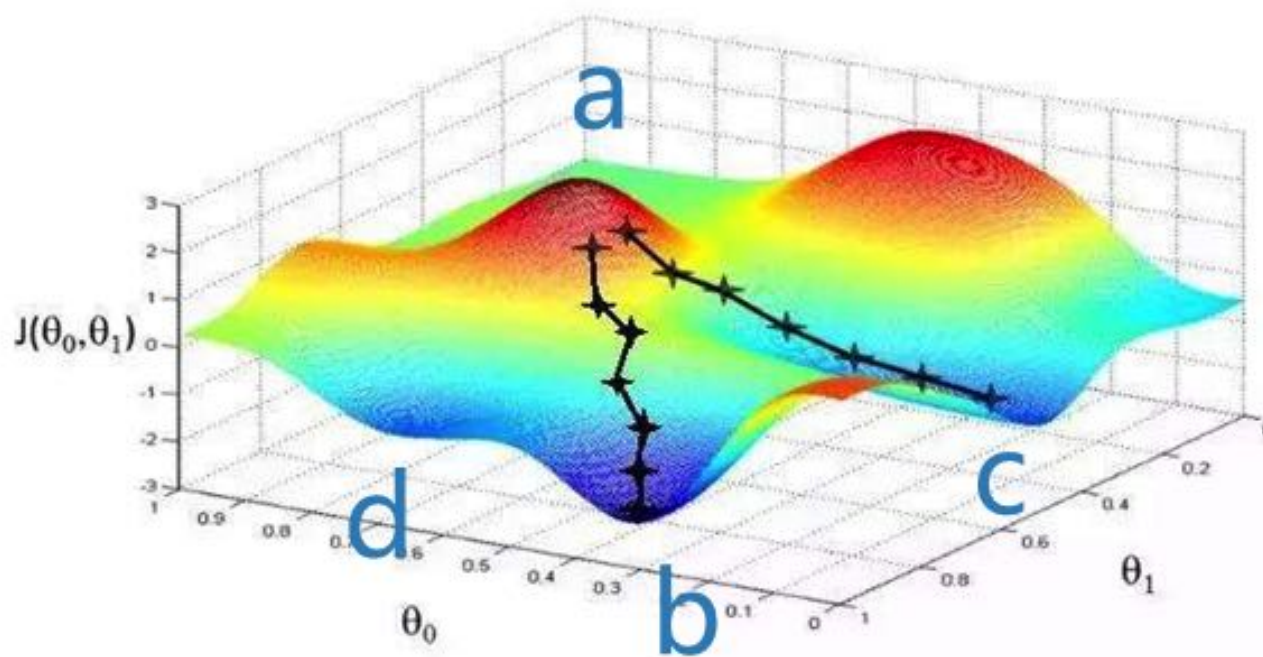
梯度下降算法

- 梯度下降法的基本思想可以类比为一个下山的过程。
- 假设小明被困在山上，浓雾很大**可见度低**，下山的路径就无法确定，他必须利用自己**周围**的信息去找到下山的路径。



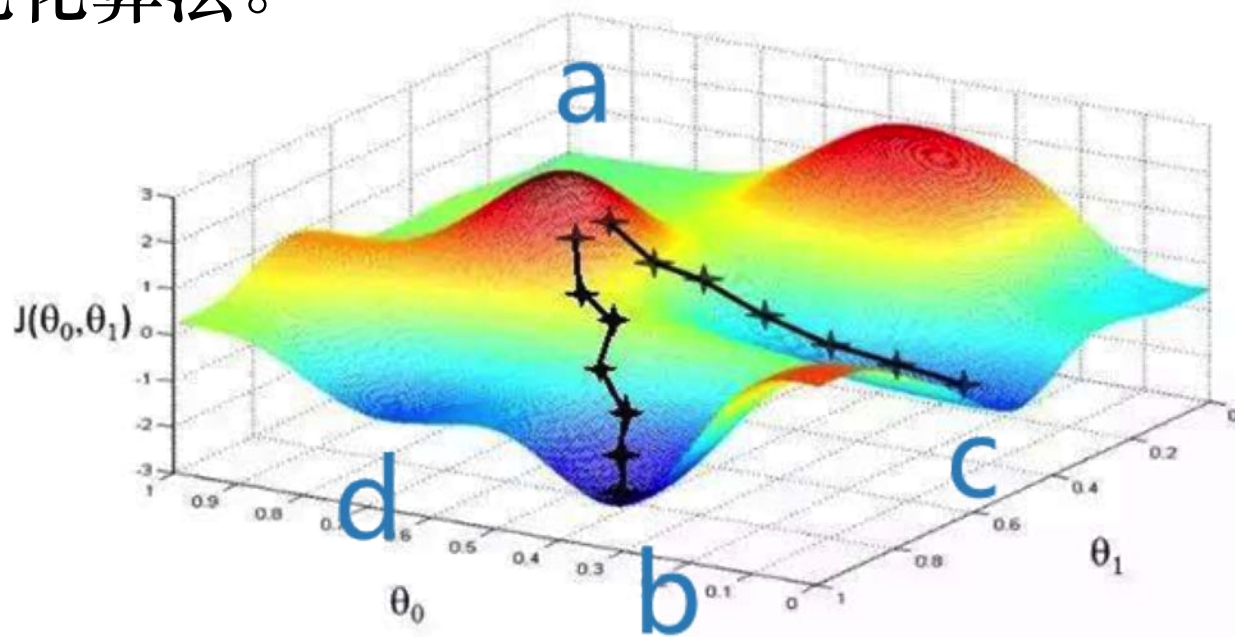
梯度下降算法

- 这个时候，他就可以利用梯度下降算法来帮助自己下山。
- 具体来说就是，以他当前的所处的位置为基准，寻找这个位置**最陡峭**的地方，然后朝着山的高度**下降**的地方走。



梯度下降算法

- 小明**每次**沿着当前位置下山**一小步**，这样就能保证越来越接近山谷，即得到**更小**的损失函数，直到到达**b**，得到损失函数最小值。
- 但如果不小心进入**c**、**d**点，那么梯度下降的算法就无法再回到**b**，这时就需要其他优化算法。

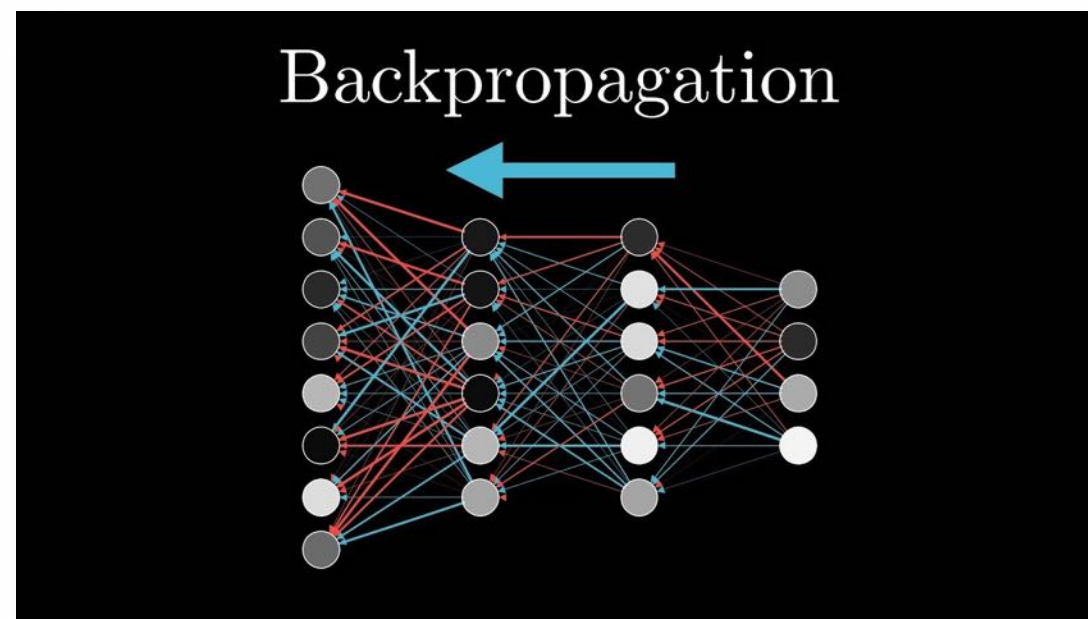


总结

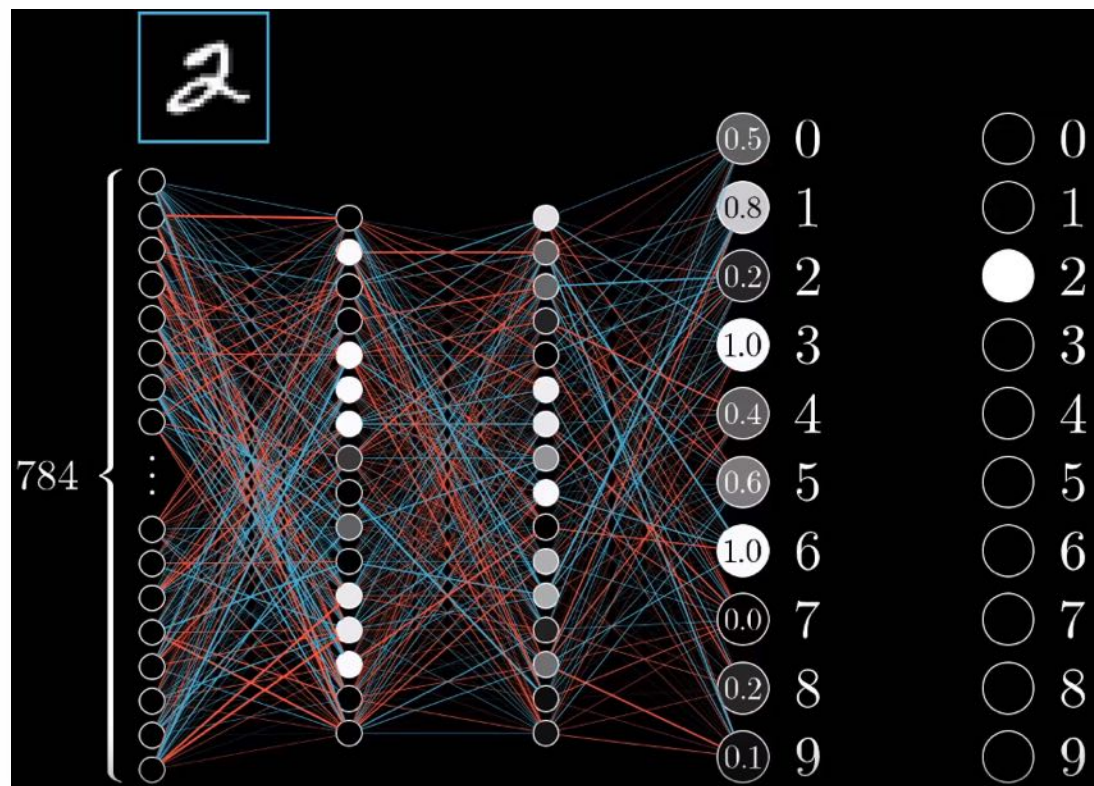
- 优化器代表了调整网络参数，使损失函数达到最小的过程。
- 不同的优化算法的计算过程和效率也各不相同。
- **梯度下降算法**是最常用也是最有效的优化。

反向传播

- 通过多层神经网络得出的结果与实际标签的结果作比较，得到了损失函数。
- 根据梯度下降的方法，将**当前**的损失函数反馈给**之前**各层的神经网络
- 并**调整**各层网络参数的**权值**，这个过程称为**反向传播**。



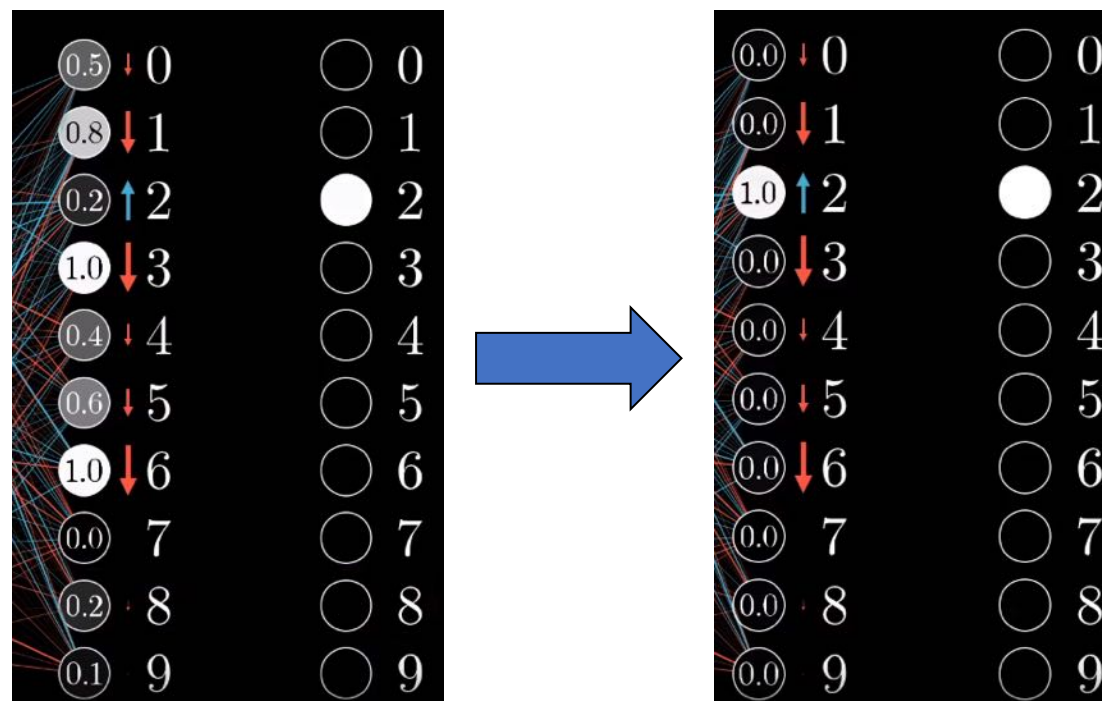
结果比较



- 对于数字2来说，未训练好的网络可能得到如下的一个结果
 - 784个像素值通过神经网络计算得到每个数字的概率
 - 与真实值相距甚远
- 我们不能直接调整输出结果值
- 我们能改的只有网络中的各个参数值
 - 使得同样的784个像素输入要得到理想的真实值

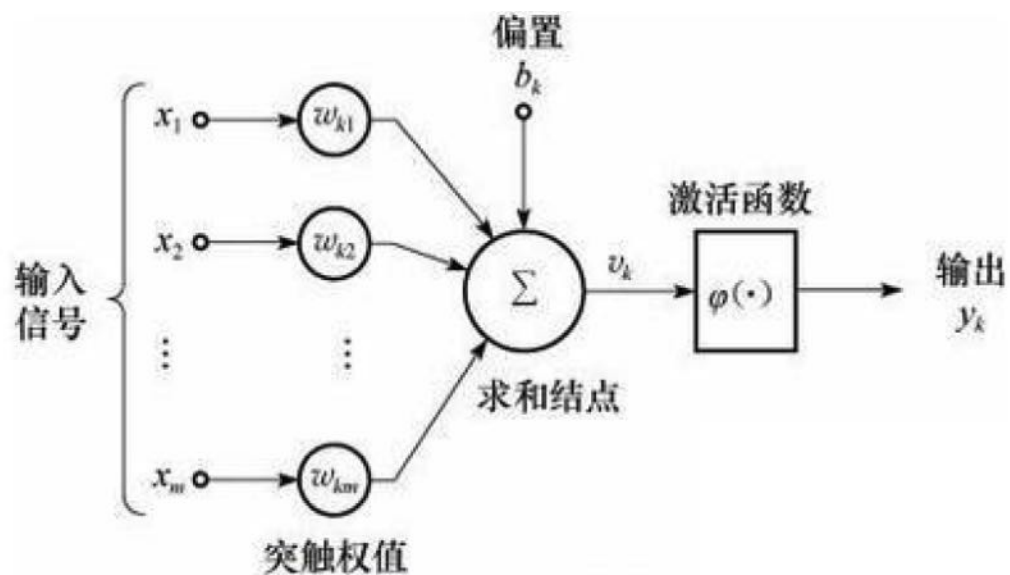
结果比较

- 对于这个结果，我们**希望**代表2的神经元的值提高，而把其他的神经元值降低。
- 提高和降低的程度和它现有的值成比例，提高2的值就比降低8的值来的重要。



神经元的参数

$$y = w_1x_1 + w_2x_2 + b$$



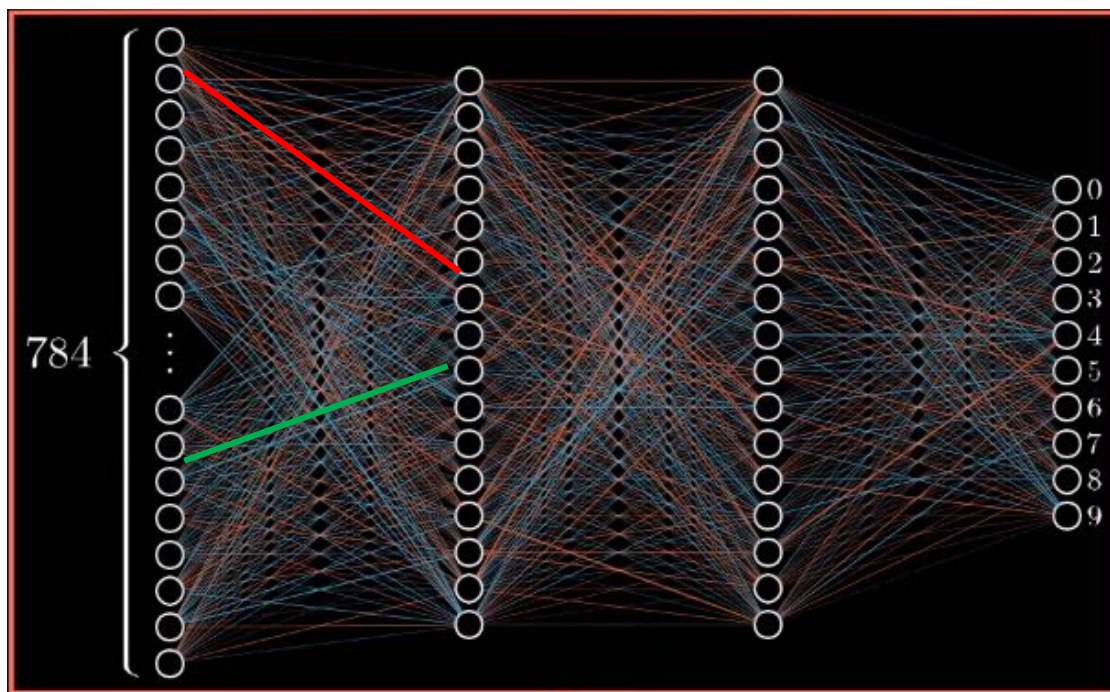
- 一个神经元的输入输出可以类比为 一个多元一次的线性方程。
- 例如：二元一次：
- 可以通过调整 w 、 x 、 b 的值来调整 y 的值。

调整参数

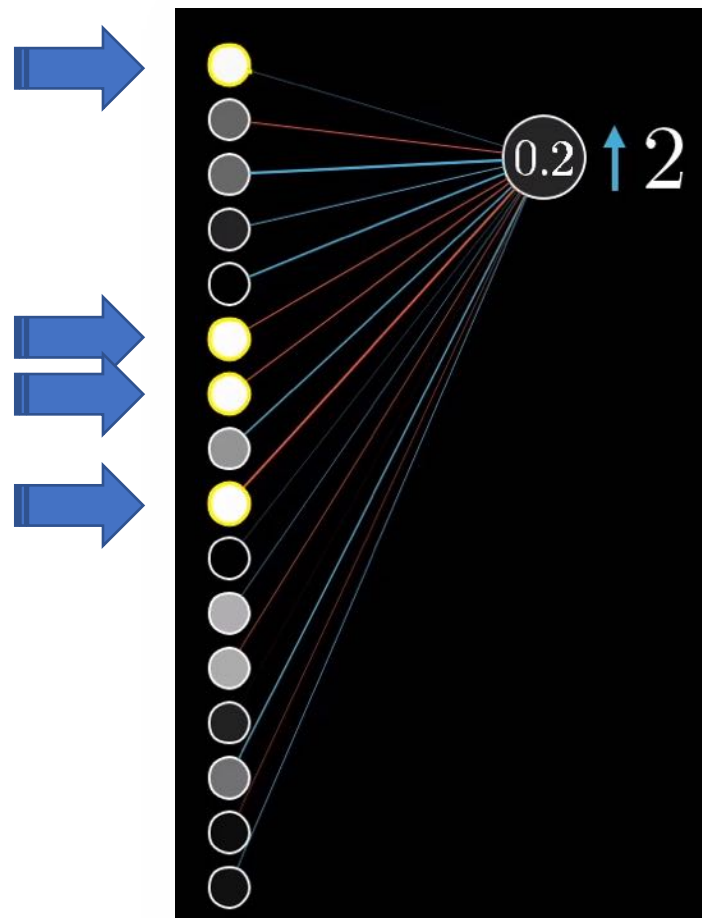
- 根据梯度大小不同，调整不同的参数对损失函数影响不同。

梯度值：3.6 调整它的影响大

梯度值：0.1 调整它的影响小

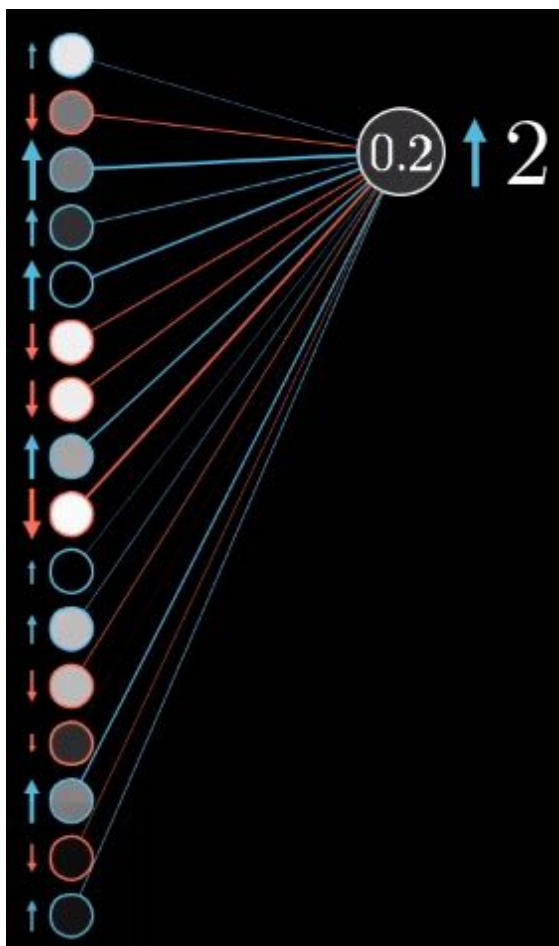


调整系数 w



- 调整输出 y 的一个方法是调整系数 w 。
- 前一层的神元激活值越大，表示越有可能“看到”某些与2相关的特征。
- 调整**激活值大**的神元的系数 w_i ，
- 影响比调整激活值小的参数要大。

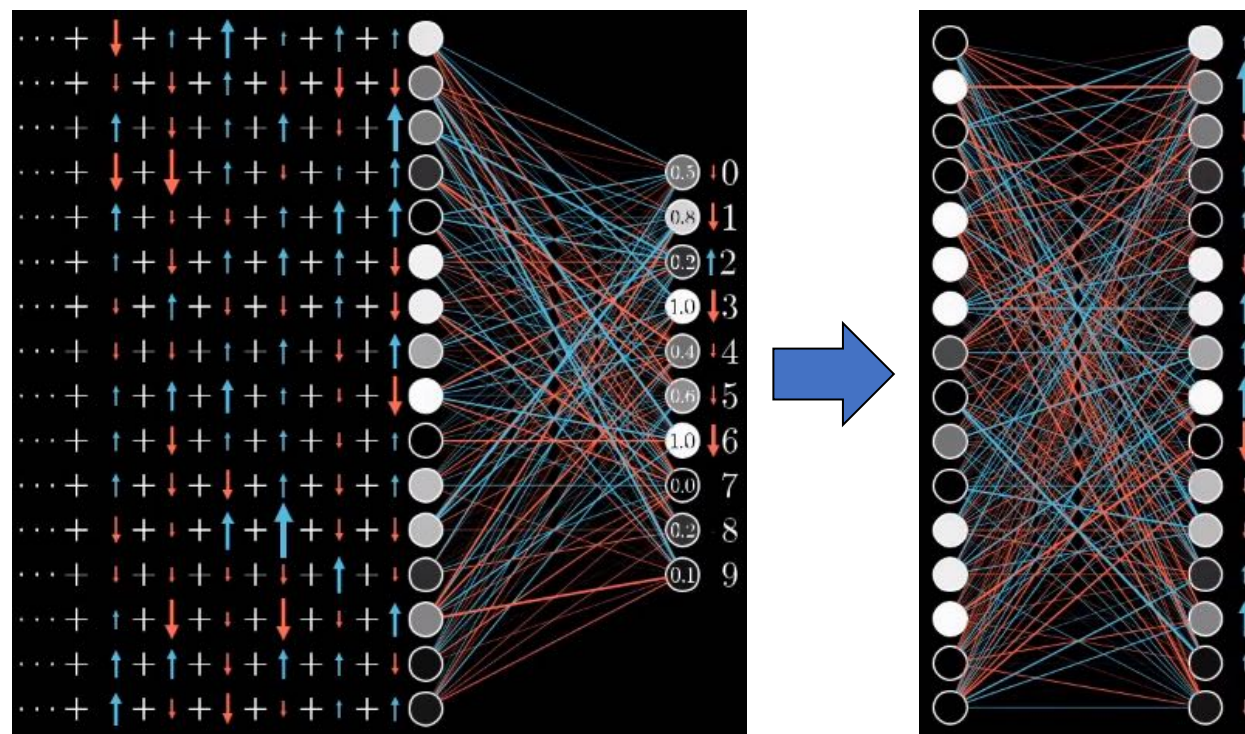
调整输入x



- 调整输出 y 的另一个方法是调整输入 x 。
- 增加原始参数 w 是正值的神经元的激活值，
- 减少原始参数 w 是负值的神经元的激活值，
- 增加和减少的程度与参数 w 大小相关。

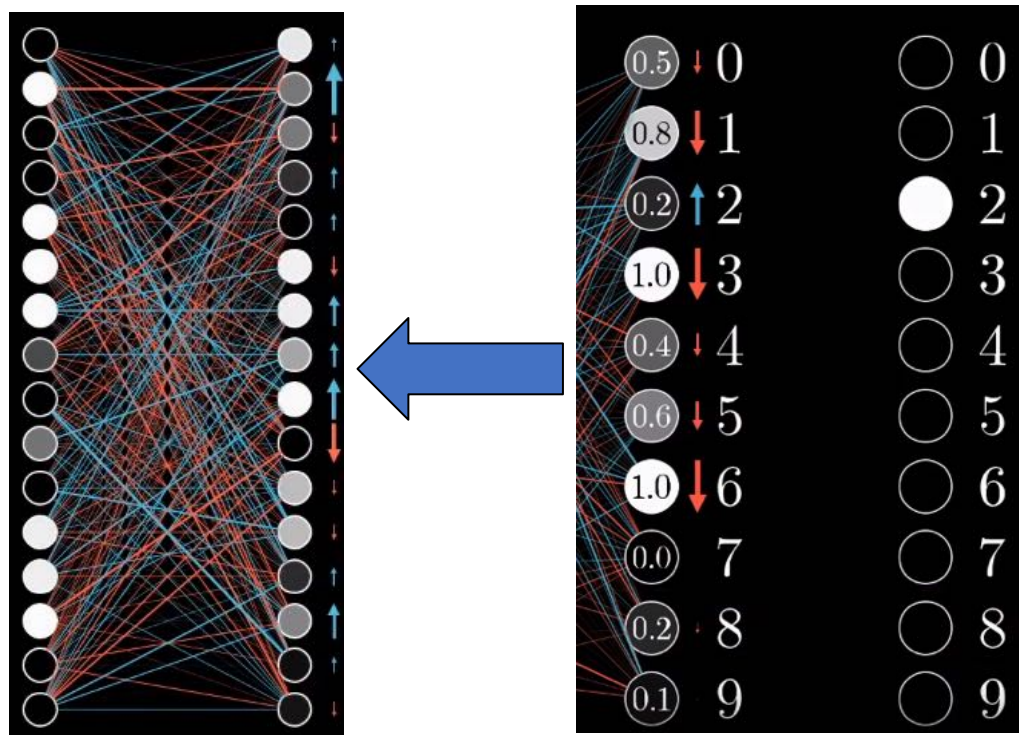
调整输入x

- 但我们无法直接改变每个神经元的激活值。
- 将每个输出需要改变的大小叠加之后，能够得到总改变量。



反向传播

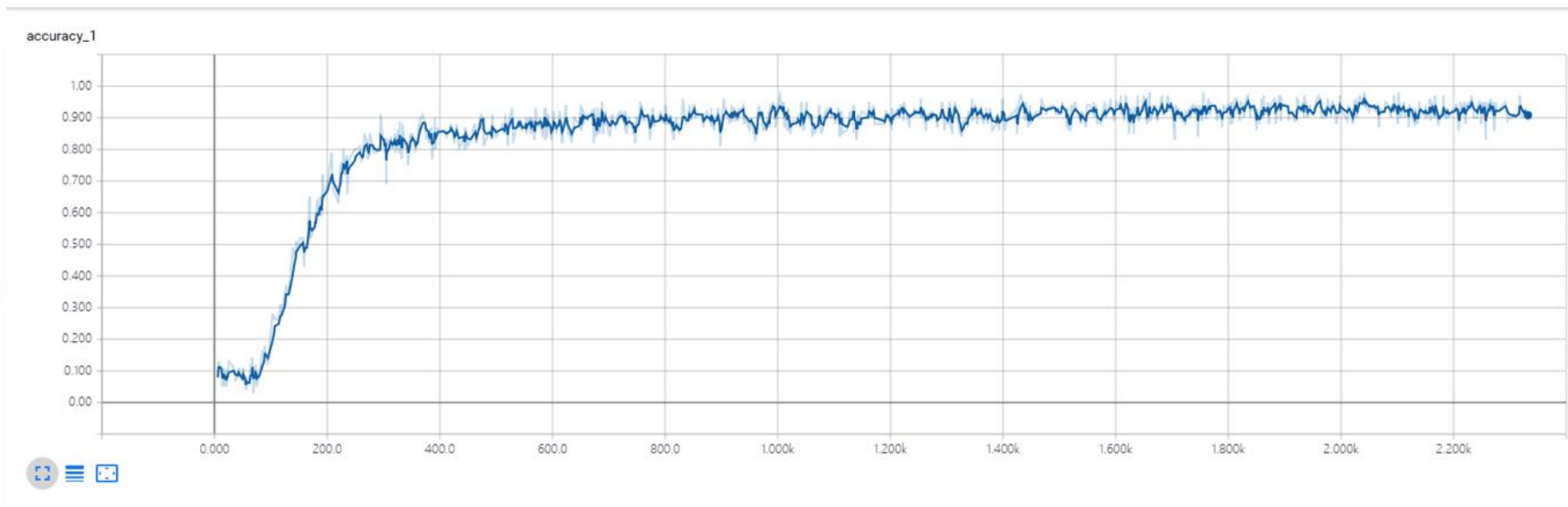
- 通过刚才的过程，我们实现了从最后一层到倒数第二层的传播。
- 这样不断层层传播回去，最后我们就能知道该如何修改整个网络的参数。



训练结果

- 从随机参数开始，经过梯度下降法和反向传播训练
- 手写数字识别准确率从不到10%提高到了93%

```
Accuracy at step 0: 0.0819
Accuracy at step 128: 0.3761
Accuracy at step 256: 0.7901
Accuracy at step 384: 0.8468
Accuracy at step 512: 0.8702
Accuracy at step 640: 0.886
Accuracy at step 768: 0.8964
Accuracy at step 896: 0.9032
Accuracy at step 1024: 0.9078
Accuracy at step 1280: 0.9111
Accuracy at step 1408: 0.9116
Accuracy at step 1536: 0.9141
Accuracy at step 1664: 0.9194
Accuracy at step 1792: 0.9227
Accuracy at step 1920: 0.9265
Accuracy at step 2048: 0.9275
Accuracy at step 2176: 0.9284
Accuracy at step 2304: 0.9313
```



深度学习图像分类算法

- 请按照深度学习的几个要素，描述**手写数字识别**的算法
 - 输入是什么、输出是什么
 - 训练数据是什么？
 - 神经网络有哪些层？如何连接？
 - 神经网络有哪些参数需要通过训练得到？
 - 如何训练神经网络？
 - 损失函数是什么？
 - 梯度下降和反向传播是怎么帮助训练的？
- 请模仿手写数字识别的算法，描述一个对照片进行分类（猫、狗、汽车、自行车、人、会议、饮料、房屋、大树等类别）的深度学习算法
 - 输入输出是什么？
 - 你需要什么样的训练数据？
 - 了解ImageNet
 - 训练的过程跟手写数字识别相比，有哪些共同点和不同之处？
- 可以设想识别音乐的流派吗？