九曲阑干

AI深度学习

神 经 网 络 概 述

九曲阑干

1 认识神经网络

- 神经网络的分类与基本结构
- 输出层的设计与softmax
- 全连接神经网络结构

2 神经网络训练

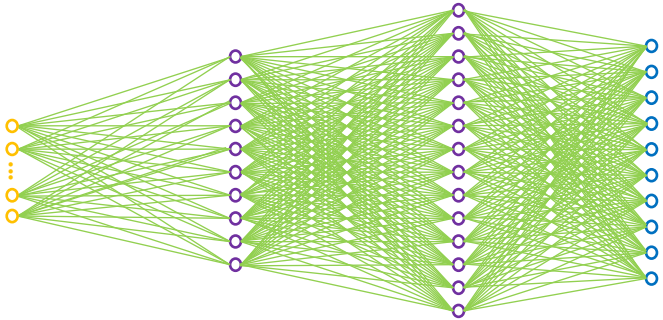
- 使用数据集
- Batch、epoch、迭代
- 损失函数
- 前向传播
- 反向传播
- 超参数

1.认识神经网络

神经网络-定义

神经网络：一种模仿生物神经网络（尤其是人脑）的计算模型

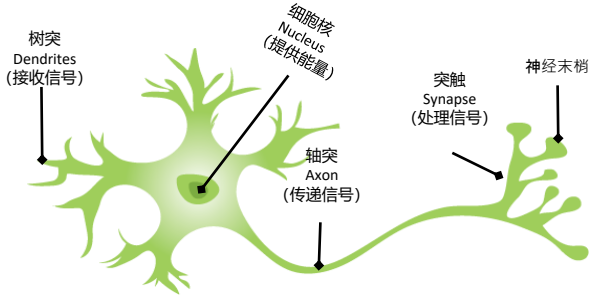
生物神经网络
Biological Neural Networks



人工神经网络
Artificial Neural Network

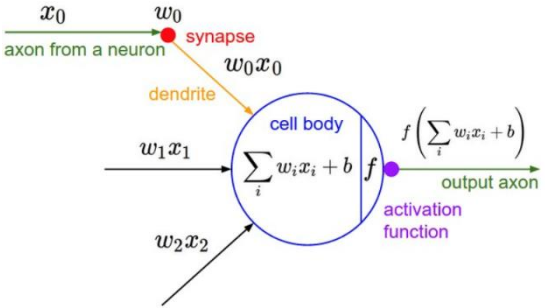
神经网络-神经元

树突：接受刺激，传向胞体
胞体：处理刺激，产生冲动
轴突：接受冲动，向外传导
多输入单输出
输入分兴奋和抑制
具有整合和阈值特性



神经元（生物神经网络）

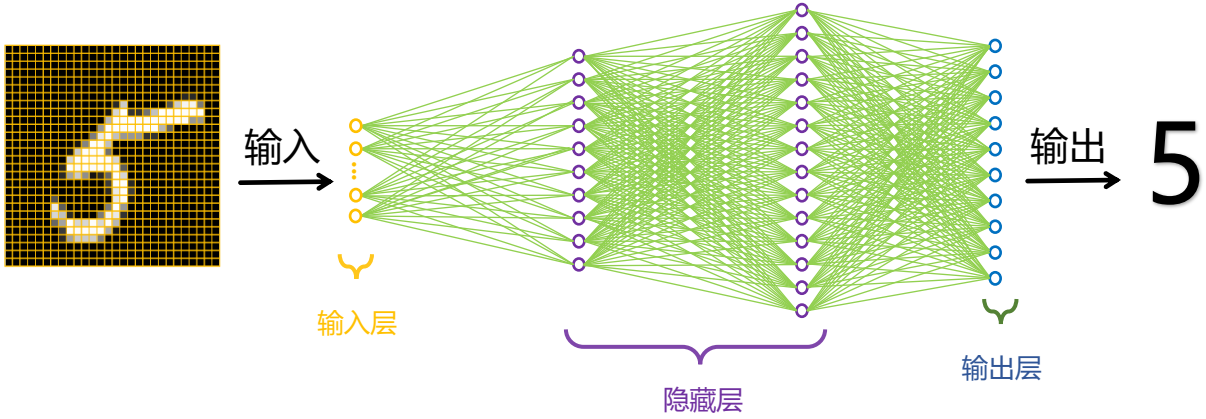
输入：多个输入x0,x1,x2
输出：单个输出
权重：兴奋抑制w0,w1,w2
偏置：阈值特性b
求和：整合特性 Sum
激活：产生冲动 f



神经元（人工神经网络）

神经网络-基本结构

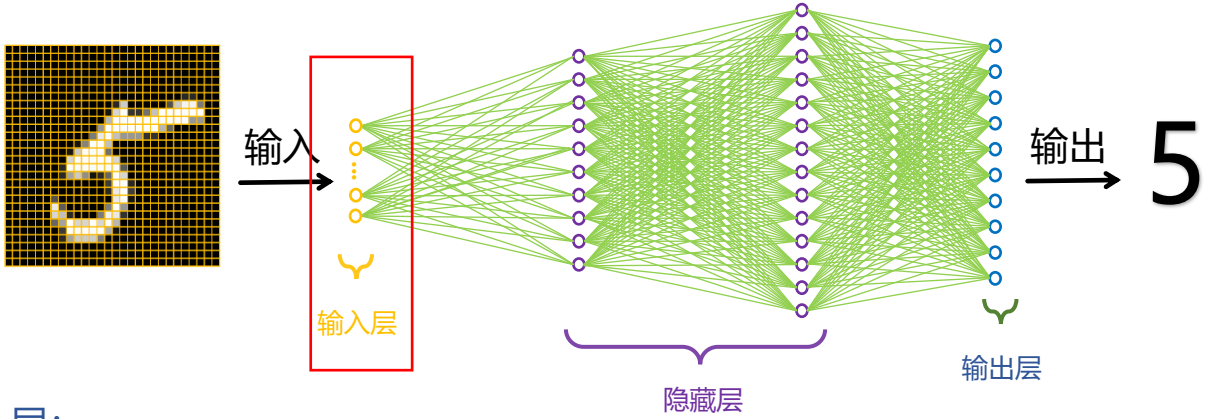
神经网络由多个层组成，每层包含多个神经元



全连接神经网络（多层感知机MLP）：
网络中的每个神经元都和下一层的全部神经元相连

神经网络-基本结构

神经网络由多个层组成，每层包含多个神经元

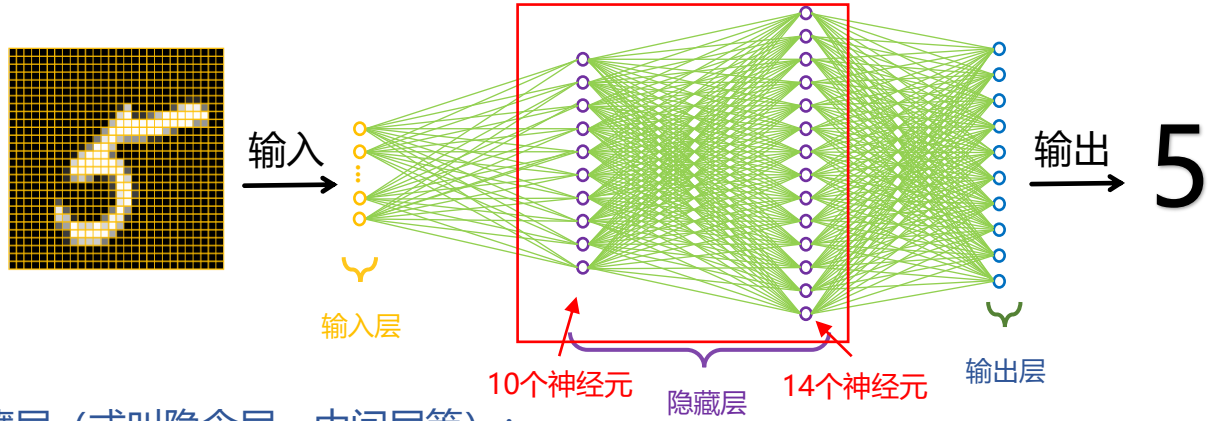


输入层：
表示原始输入数据（如归一化和flatten后的手写数字图像）
输入层的神经元数量为输入数据维度，是确定的（784）
一般只有一层输入层
计为神经网络的第0层（输入层不算在神经网络的总层数里）

面试问题1：输入层的功能是什么？一般数量有多少？

神经网络-基本结构

神经网络由多个层组成，每层包含多个神经元

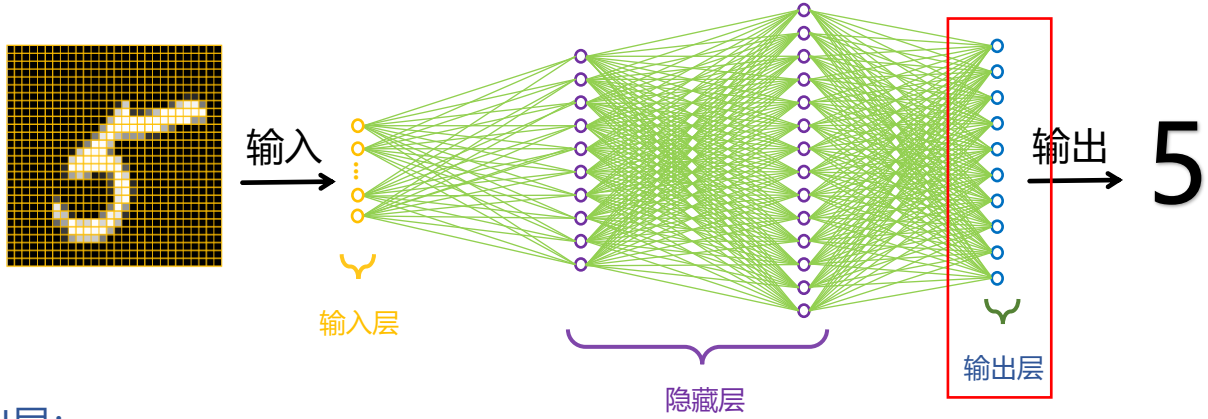


隐藏层（或叫隐含层，中间层等）：
对输入数据非线性变换，以进行特征提取和加工
一般有多个隐藏层
隐藏层的数量，每个隐藏层的神经元数量都可以自己调整
第一个隐藏层计为神经网络的第1层

面试问题2：隐藏层的功能是什么？一般数量有多少？

神经网络-基本结构

神经网络由多个层组成，每层包含多个神经元

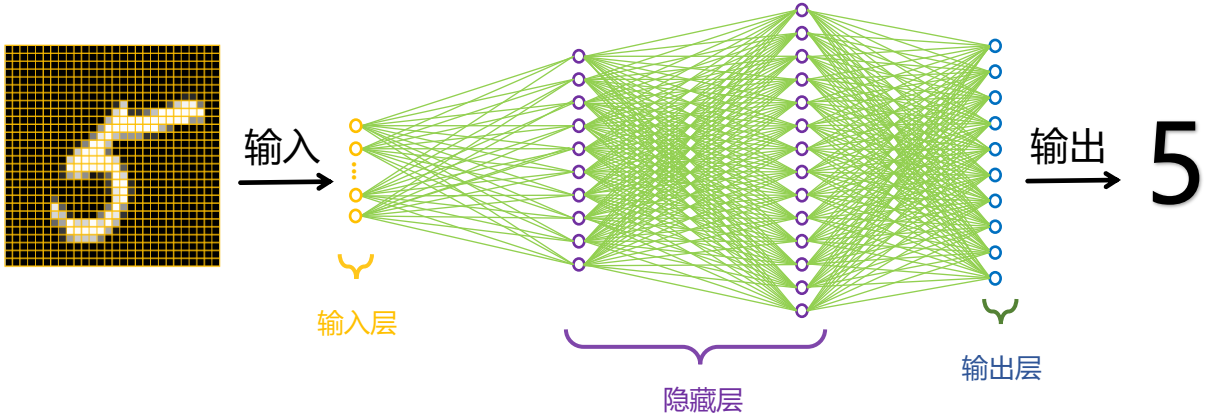


输出层：
输出最后的分类结果（每个类的概率大小）
输出层的神经元数量为分类类别数，是确定的（10）
一般只有一层输出层
神经网络层数一般为隐藏层数量+输出层数量

面试问题3：输出层的功能是什么？一般数量有多少？

神经网络-基本结构

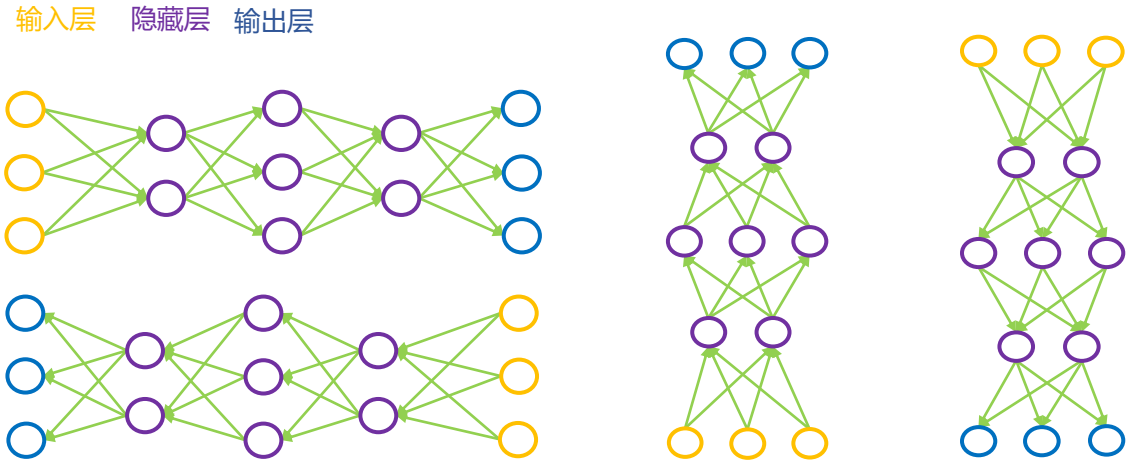
神经网络由多个层组成，每层包含多个神经元



输入层：输入数据，一层
隐藏层：非线性变换（特征提取），多层
输出层：输出结果，一层

面试问题：全连接神经网络中的层有哪几类？其功能分别是什么？数量分别有多少？

神经网络-方向

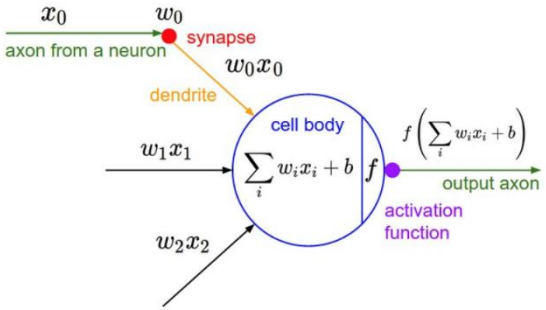


神经网络的方向：**从输入到输出**

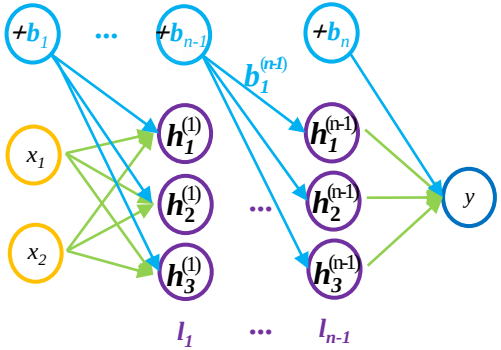
圆圈代表**神经元**，各个神经元之间的**连接线**代表该神经元对应的**权重**（**箭头**代表数据的流向）

神经网络的图示是一种表现形式，由左至右、由下至上表达都可以

神经网络中的重要概念



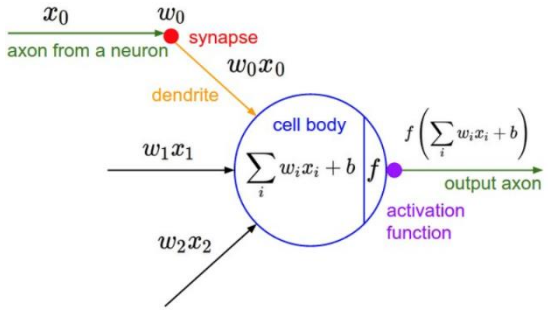
神经元



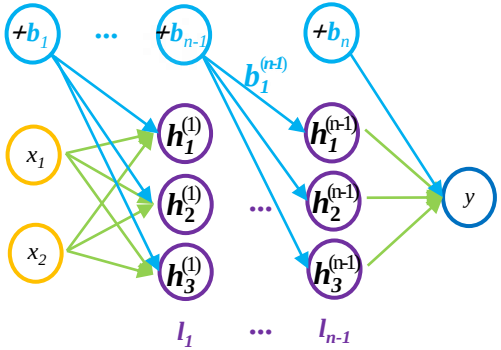
多层神经网络

重要概念：
输入、输出、**权重、偏置，激活**

神经网络-输入输出



神经元



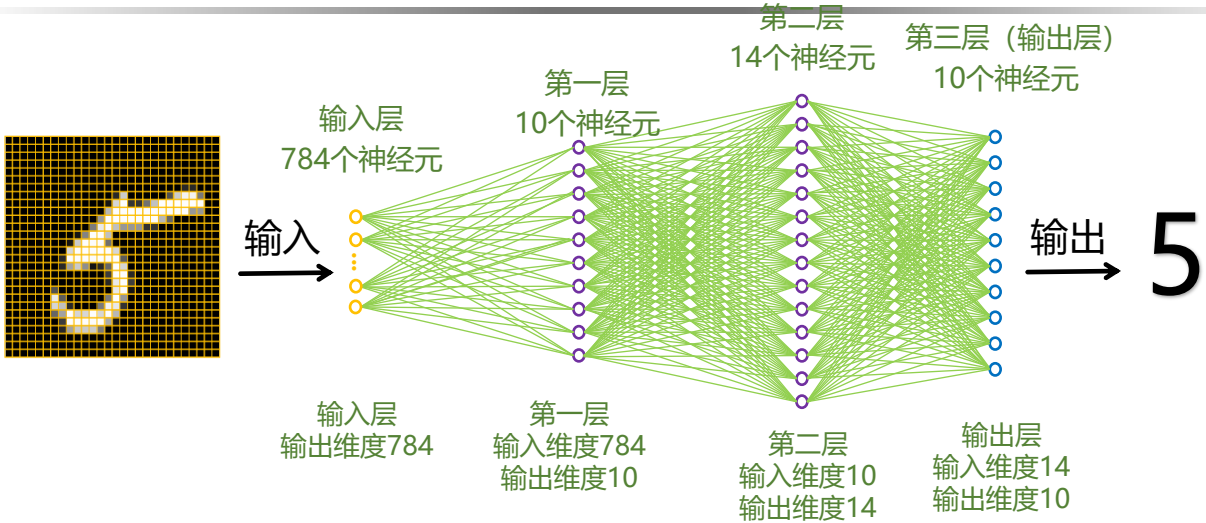
多层神经网络

神经元：
多个输入
一个输出

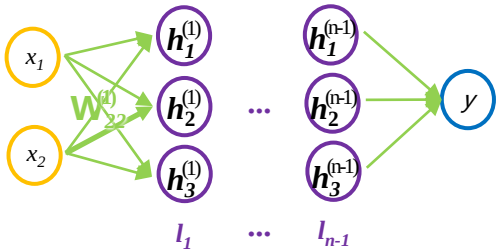
面试问题：全连接神经网络中某一层的输入输出维度如何确定？

神经网络：
输入输出都用一维向量表示
本层的输入是上一层的输出
多个输入，输入维度为上一层的神经元个数
多个输出，输出维度为本层的神经元个数

神经网络-输入输出


输入层
784个神经元
第一层
10个神经元
第二层
14个神经元
第三层（输出层）
10个神经元
输入
输出
5
输入层
输出维度784
第一层
输入维度784
输出维度10
第二层
输入维度10
输出维度14
输出层
输入维度14
输出维度10

神经网络-输入输出



多层神经网络

圆圈中的字母表示的是该神经元的输出结果

一层的输出是一个一维向量

$$h^{(i)} = (h_1^{(i)}, h_2^{(i)}, \ldots, h_n^{(i)})$$

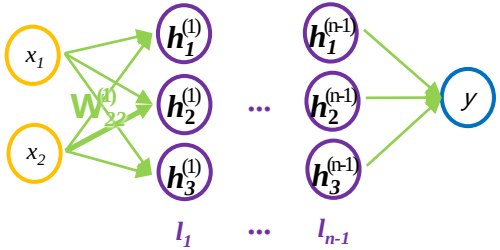
i表示第i层，n是第i层的神经元个数

表示层数：第1层的输出

$h_1^{(1)}$

表示对应的神经元在本层的位置：第1个神经元的输出

神经网络-权重weight



多层神经网络

两层之间的连线表示权重weight
权重是一个二维数组

$$W^{(i)} = \begin{bmatrix} w_{11}^{(i)} & w_{12}^{(i)} \cdots & w_{1n}^{(i)} \\ w_{21}^{(i)} & w_{22}^{(i)} \cdots & w_{2n}^{(i)} \\ \vdots & \vdots & \vdots \\ w_{m1}^{(i)} & w_{m2}^{(i)} \cdots & w_{mn}^{(i)} \end{bmatrix}$$

i表示第i层，m是第i-1层的神经元个数，n是本层的神经元个数

权重的参数个数
=前一层神经元个数*本层神经元个数

权重存在于神经网络的每一层的节点之间。

$W_{2\ 2}^{(1)}$

- (1)：表示层数：第1层的权重
- 2：表示本层神经元的位置：本层的第2个节点
- 2：表示前一层神经元的位置：前一层的第2个神经元

神经网络-权重weight

只有输入和权重怎么计算?

输入为行向量，维度1xm $h^{(i-1)} = (h_1^{(i-1)}, h_2^{(i-1)}, \dots, h_m^{(i-1)})$

输出为行向量，维度1xn $h^{(i)} = (h_1^{(i)}, h_2^{(i)}, \dots, h_n^{(i)})$

第一步：输入*权重

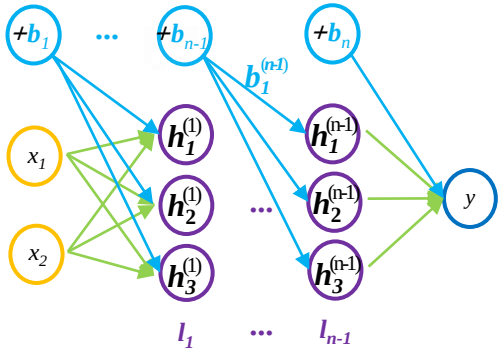
写成求和形式 $h_j^{(i)} = \sum_{k=1}^{m} h_k^{(i-1)} * w_{kj}^{(i)}$

写成矩阵形式 $h^{(i)} = h^{(i-1)} * W^{(i)}$

权重W为矩阵，维度mxn

$$(h_1^{(i)}, h_2^{(i)}, \dots, h_n^{(i)}) = (h_1^{(i-1)}, h_2^{(i-1)}, \dots, h_m^{(i-1)}) \begin{bmatrix} w_{11}^{(i)} & w_{12}^{(i)} & \dots & w_{1n}^{(i)} \\ w_{21}^{(i)} & w_{22}^{(i)} & \dots & w_{2n}^{(i)} \\ \vdots & \vdots & & \vdots \\ w_{m1}^{(i)} & w_{m2}^{(i)} & \dots & w_{mn}^{(i)} \end{bmatrix}$$

神经网络-偏置bias



多层神经网络

偏置的参数个数=本层神经元个数

一层的偏置bias是一个一维向量，
偏置的维度和本层输出的维度相同

$$b^{(i)} = (b_1^{(i)}, b_2^{(i)}, \ldots, b_n^{(i)})$$

i表示第i层，n是第i层的神经元个数

$b^{(1)}_1$

- (1)：表示层数：第1层的偏置
- 1：表示对应神经元的位置：第1个神经元

除了输入层，神经网络的其它所有层都有偏置。

神经网络-偏置bias

加上偏置怎么计算?

输入为行向量，维度1xm $h^{(i-1)} = (h_1^{(i-1)}, h_2^{(i-1)}, \dots, h_m^{(i-1)})$

输出为行向量，维度1xn $h^{(i)} = (h_1^{(i)}, h_2^{(i)}, \dots, h_n^{(i)})$

第二步：输入*权重+偏置

写成求和形式

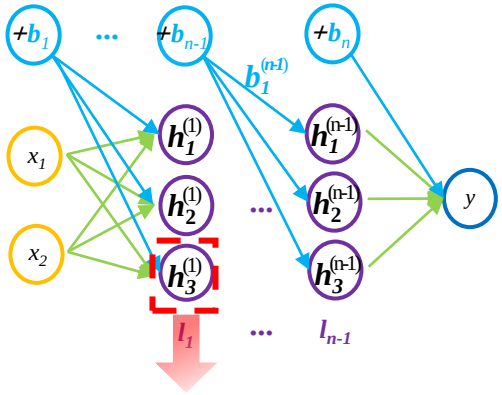
$$h_j^{(i)} = \sum_{k=1}^{m} h_k^{(i-1)} * w_{kj}^{(i)} + b_j^{(i)}$$

偏置b为行向量，维度1xn

写成矩阵形式 $h^{(i)} = h^{(i-1)} * W^{(i)} + b^{(i)}$

$$(h_1^{(i)}, h_2^{(i)}, \dots, h_n^{(i)}) = (h_1^{(i-1)}, h_2^{(i-1)}, \dots, h_m^{(i-1)}) \begin{bmatrix} w_{11}^{(i)} & w_{12}^{(i)} & \dots & w_{1n}^{(i)} \\ w_{21}^{(i)} & w_{22}^{(i)} & \dots & w_{2n}^{(i)} \\ \vdots & \vdots & & \vdots \\ w_{m1}^{(i)} & w_{m2}^{(i)} & \dots & w_{mn}^{(i)} \end{bmatrix} + (b_1^{(i)}, b_2^{(i)}, \dots, b_n^{(i)})$$

神经网络-激活函数



$$h_j^{(i)} = f\left(\sum_{k=1}^{m} h_k^{(i-1)} * w_{kj}^{(i)} + b_j^{(i)}\right)$$

隐藏层的每一个神经元都要经过激活函数激活才继续向前传

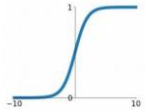
激活函数是非线性的，是逐元素操作

隐藏层常用的激活函数有：

Sigmoid，Tanh，ReLU等

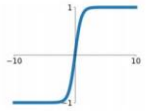
Sigmoid

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



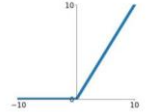
tanh

$\tanh(x)$



ReLU

$\max(0, x)$



神经网络-激活函数

加上激活函数怎么计算?

输入为行向量，维度1xm $h^{(i-1)} = (h_1^{(i-1)}, h_2^{(i-1)}, \dots, h_m^{(i-1)})$

输出为行向量，维度1xn $h^{(i)} = (h_1^{(i)}, h_2^{(i)}, \dots, h_n^{(i)})$

面试问题：全连接神经网络中，隐藏层的计算公式是什么？公式中每个变量是什么含义?

第三步：输入*权重+偏置后，激活函数对每个神经元进行计算

写成求和形式 $$h_j^{(i)} = f(\sum_{k=1}^{m} h_k^{(i-1)} * w_{kj}^{(i)} + b_j^{(i)})$$

写成矩阵形式 $$h^{(i)} = f(h^{(i-1)} * W^{(i)} + b^{(i)})$$

面试问题：神经网络中激活函数的要求是什么?

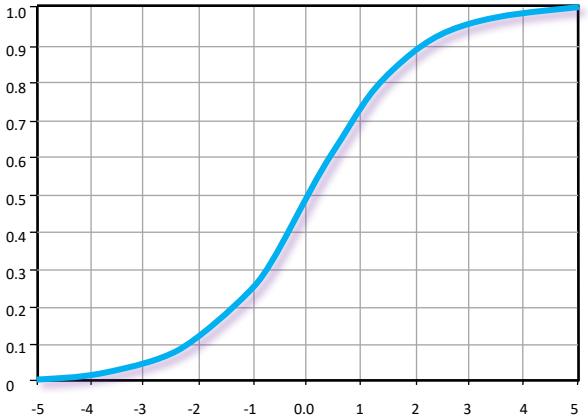
面试问题：神经网络中激活函数的作用是什么？如果把激活函数全部去掉会怎么样?

激活函数的作用：对神经网络加入非线性操作，提高神经网络的拟合能力

如果不使用激活函数：会变成输入x和一堆w连乘，最后退化成一层

输出层-softmax函数

用于**分类**的神经网络，**输出层**的每一个神经元都要经过softmax激活函数



面试问题：分类网络的输出层使用softmax的目的是什么？

函数表达式:

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

exp(x)：是表示 e 的指数函数

n：输出层神经元的个数（即分类的所有类别数）

k：第k个神经元；

分子：输入信号 a_k 的指数函数；

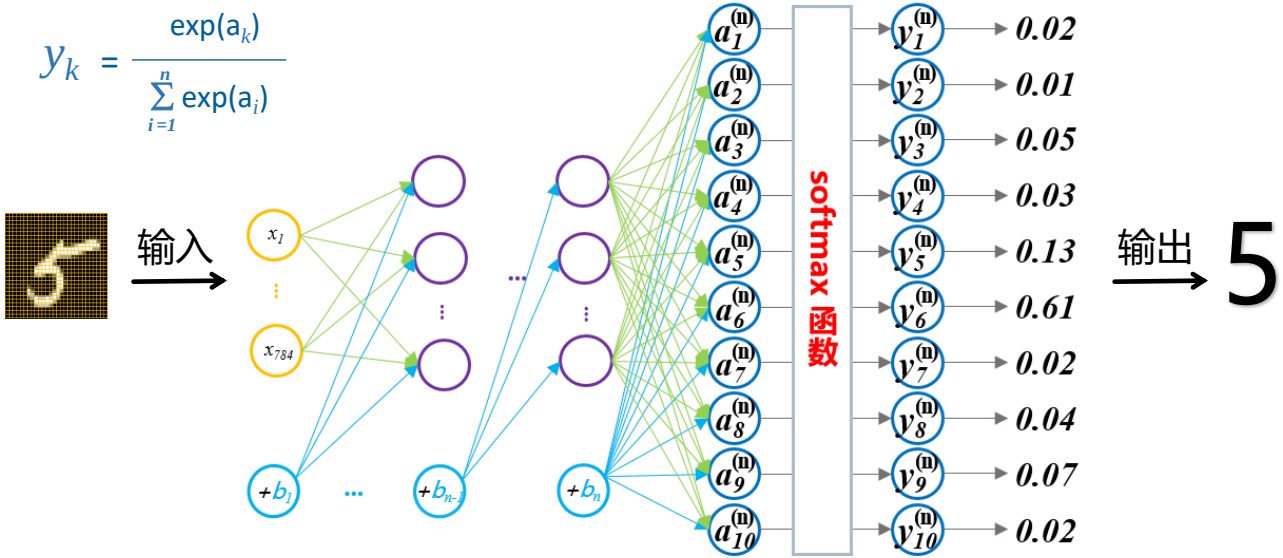
分母：所有输入信号的指数函数的和。

softmax将输出映射到0.0 到1.0之间的实数，输出和总是1。可以理解为“分类概率”

经过softmax计算后，概率值不会为负数。

输出层-softmax函数

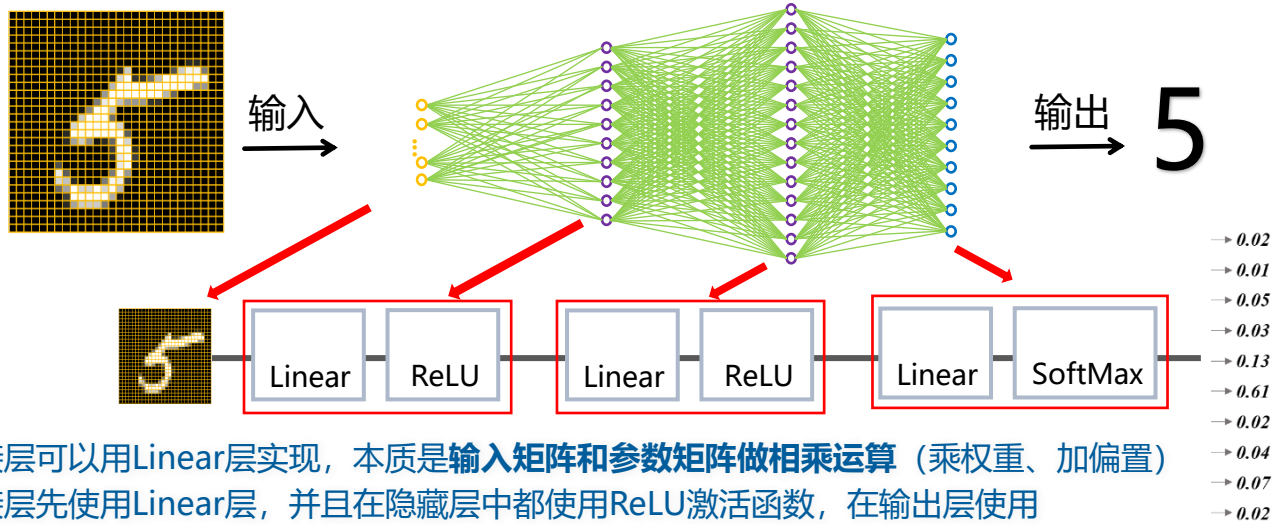
$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$



对于训练集，$y^{(i)} \in \{0,1,2,\ldots,k\}$，共k个类。以MNIST为例：此时k=10，$y \in \{0,1,2,3,4,5,6,7,8,9\}$

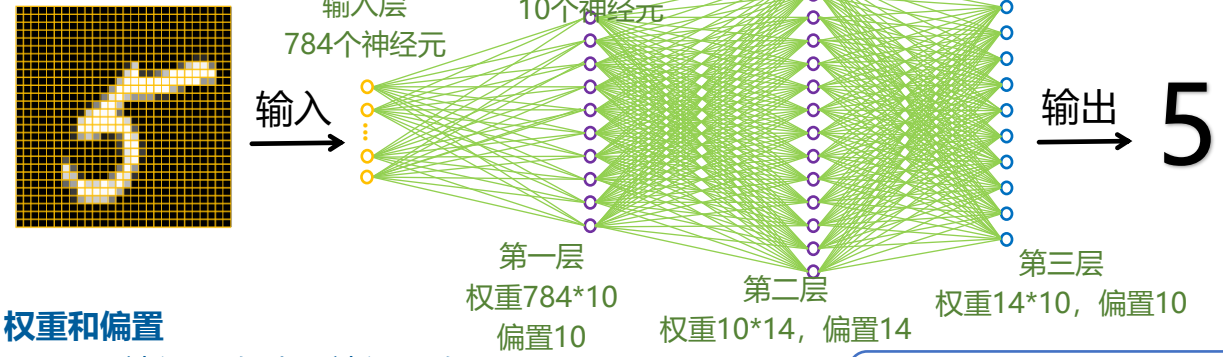
神经网络-基本结构

全连接神经网络：相邻层的所有神经元之间都有连接，称为全连接（fully-connected）



全连接层可以用Linear层实现，本质是**输入矩阵和参数矩阵做相乘运算**（乘权重、加偏置）
全连接层先使用Linear层，并且在隐藏层中都使用ReLU激活函数，在输出层使用Softmax激活函数，那么一个3层的全连接神经网络可以表示为上图。

神经网络-参数



参数：权重和偏置

权重数=上一层神经元数*本层神经元数

偏置数=本层神经元数

权重weight：784*10+10*14+14*10=**8120** 个

偏置bias：10+14+10=**34** 个

总参数数量：权重8120+偏置34=8154

（训练神经网络：找到合适的权重和偏置的过程。）

面试问题1：全连接神经网络中的参数包括哪两部分？

面试问题2：如何计算全连接神经网络中的参数个数？权重和偏置的参数个数分别怎么计算？（可举例说明）

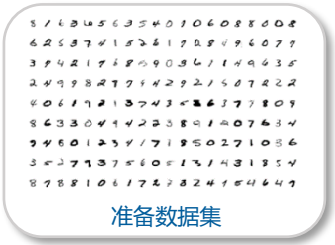
2.神经网络训练

模型的参数

我们要学习的就是参数（parameter）。
模型的参数主要是：权重与偏置。

训练神经网络：找到合适的权重和偏置

训练流程


准备数据集

+


确定网络结构，
参数初始化

+

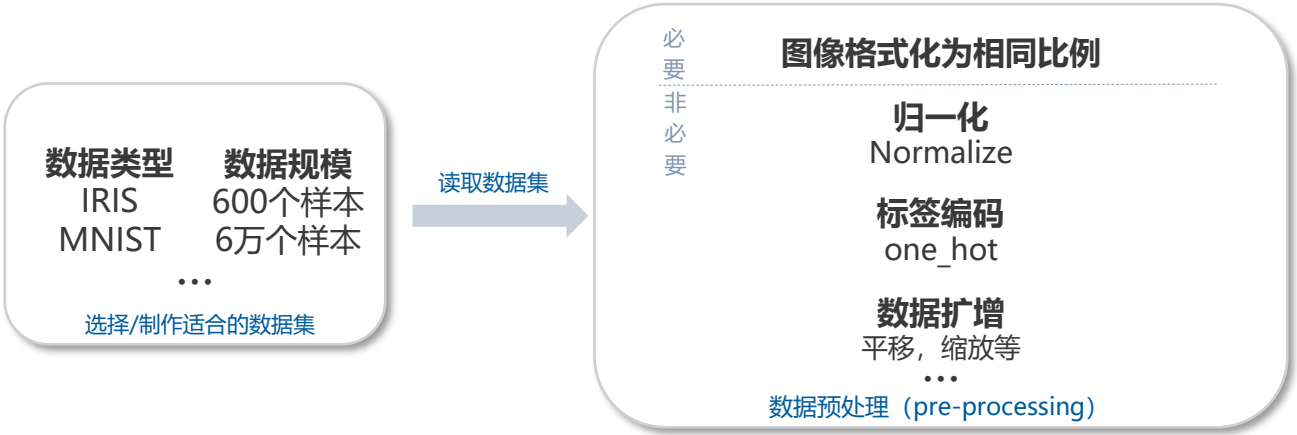

工具，训练神经网络

=

确定参数
获得模型

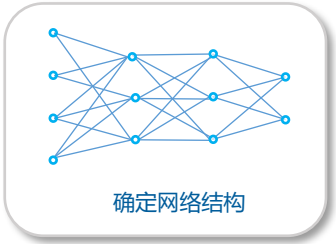
以数据集为原料
确定好网络结构
利用程序设计语言、软件库、工作站等工具进行训练
得到一个较好的模型。

数据集

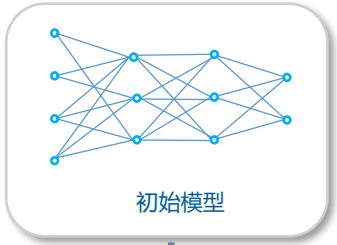


- ✓ 需要对数据集进行预处理，包括归一化，标签变为one-hot。
- ✓ 可以根据需求进行**数据扩增**，数据扩增是非常重要的**训练集**预处理的方式，本质上是在增加训练集的数据量，为了提高模型**泛化**能力。验证集和测试集一般不需要数据扩增。

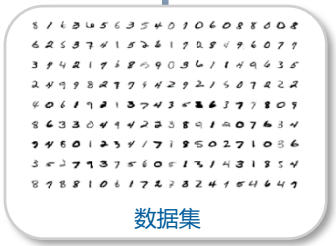
初始模型：确定网络结构，初始化参数



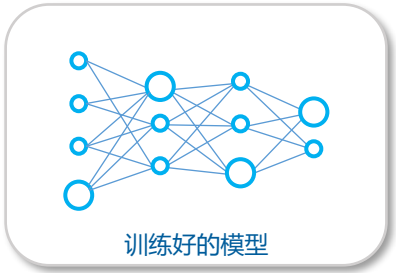
初始化参数



+



训练



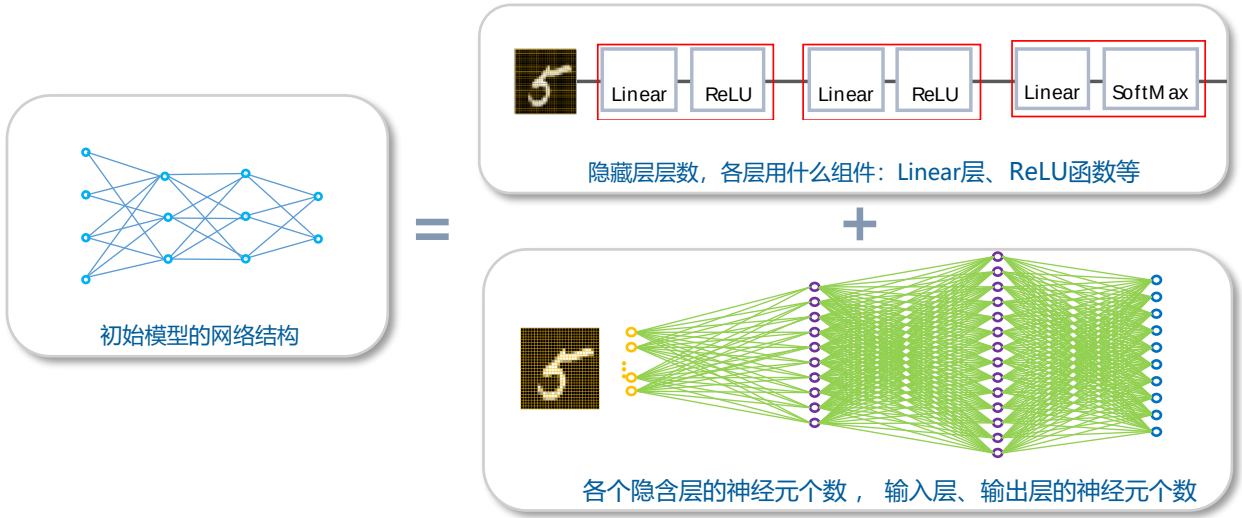
网络初始化：在训练开始时需要设置初始模型

初始模型包含**网络结构**与**一组初始参数**

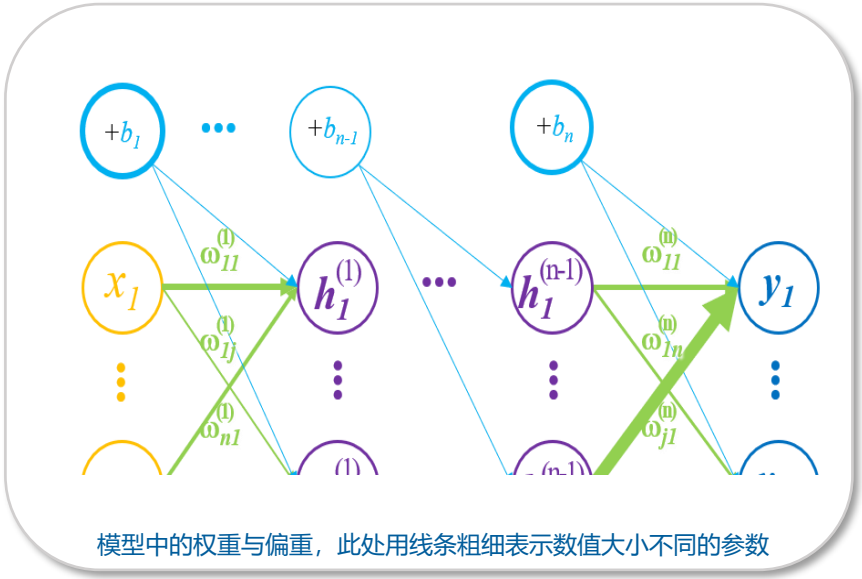
再将数据喂入这个初始模型去迭代训练，最后获得合适的参数

确定网络结构

需要确定：神经网络有几层，每层用什么类型，每层的神经元个数



初始化参数



模型中的权重与偏重，此处用线条粗细表示数值大小不同的参数

参数初始化时，会在一个固定区间内**随机**取一组数值作为初始参数。

一般使用高斯随机数或者均匀随机数

比如全连接层的权重可如下的均匀分布中获取（m为输入节点数）：

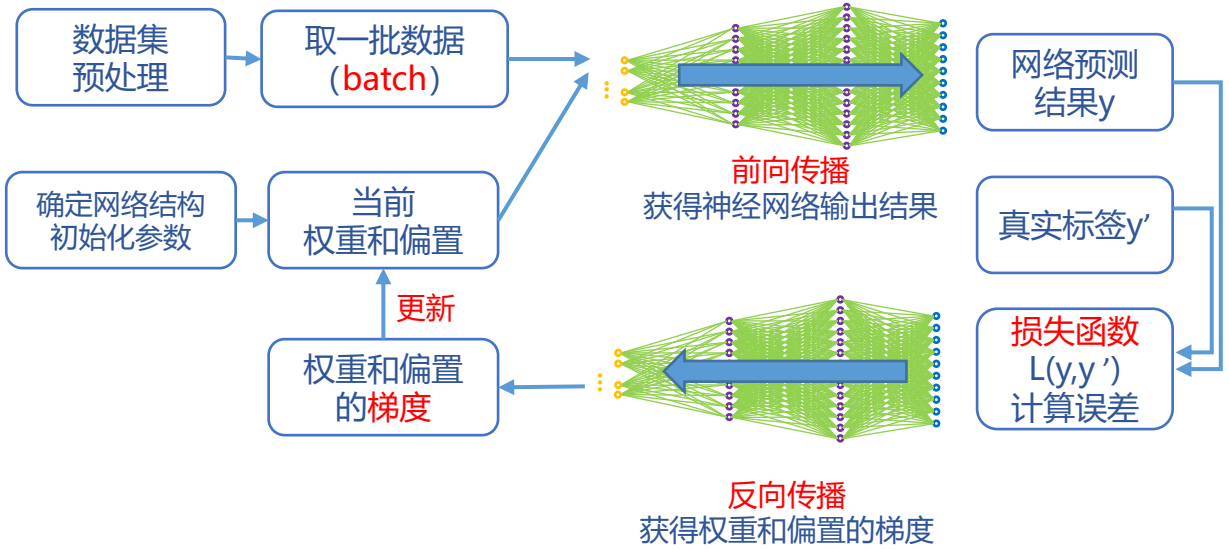
$$U\left(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right)$$

举例：若m=10(即网络有10个输入)则该神经网络的初始权重需从区间 $\left(-\frac{1}{\sqrt{10}}, \frac{1}{\sqrt{10}}\right)$ 中取随机取一组数。

同理，偏置也要在某一区间内取值。

神经网络训练总览

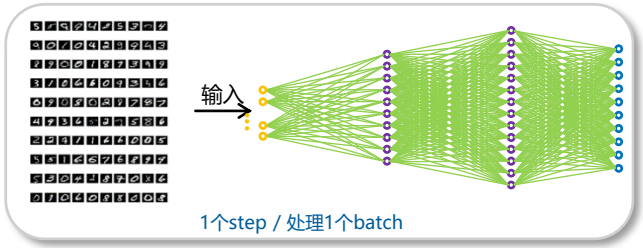
使用随机梯度下降法对神经网络进行训练，迭代更新神经网络的参数



batch

九曲阑干

batch：批
batch-size：1次迭代所使用的样本量。
每次取一个batch的数据，输入到神经网络计算
计算一个batch即一次迭代
一次迭代会对参数进行一次更新

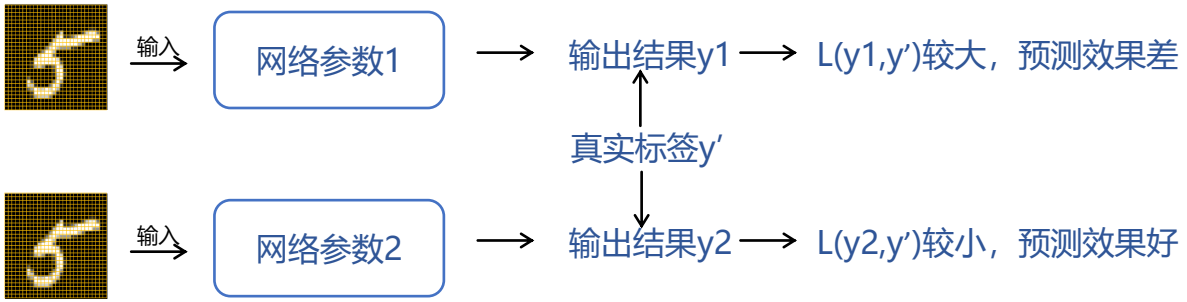


epoch：轮
遍历（跑完）一遍训练集称为跑完一个epoch
一般需要迭代非常多次，即非常多epoch，才能完成训练

— 举例 —
✓ 训练目的：识别手写数字
✓ 使用MNIST数据集，训练集设为55,000个样本（分出5,000个样本做验证集），测试集设为10,000个样本。
✓ 先来设置一下batch-size，如设为100，处理完一个batch（100个样本）的过程称为一次迭代（一个step）
✓ 那么需要跑550个step（有索引号0~549）才能跑完一遍训练集，即一个epoch中进行550次迭代。

损失函数（目标函数）

损失函数loss function（或目标函数objective function）
作用：衡量输出结果与真实标签的误差（损失）。
训练目标：最小化损失函数

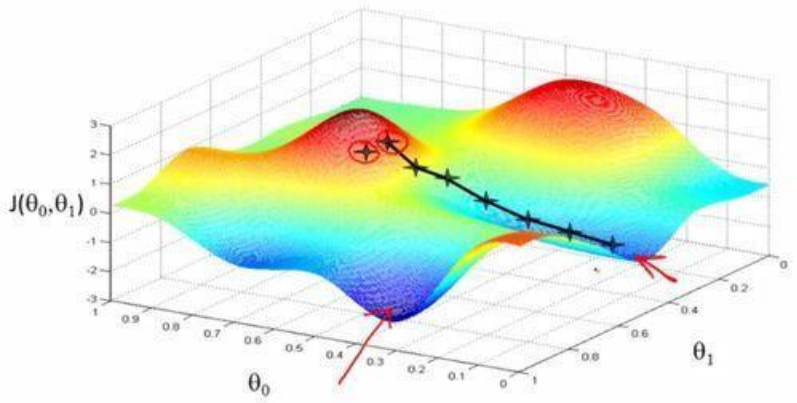


损失函数（目标函数）

损失函数loss function（或目标函数objective function）
作用：衡量输出结果与真实标签的误差（损失）。
训练目标：最小化损失函数

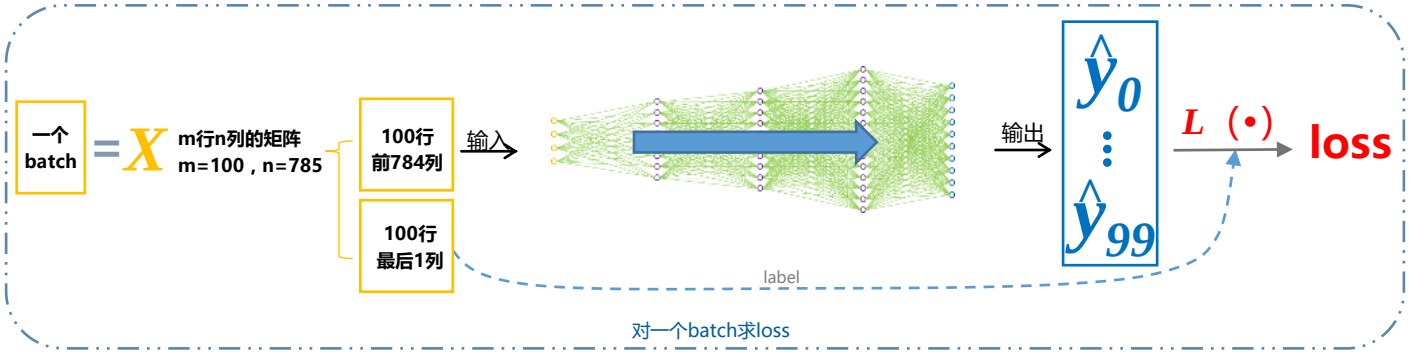
随机梯度下降基本思路：
- ✓ 根据损失，利用链式求导法则计算梯度，
- ✓ 利用梯度更新参数，
- ✓ 参数更新后的损失会减小
- ✓ 不断向着损失更小的方向优化

梯度的反方向是损失函数下降最快的方向



注意：本次课程仅对网络训练流程进行简要介绍，后面课程中会进行更加详细的介绍，包括损失函数计算，梯度公式推导，相关的面试题等等

前向传播-求Loss



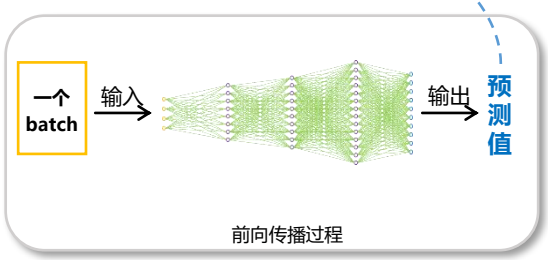
前向传播
方向：从前向后
计算：利用输入batch，计算输出结果和损失

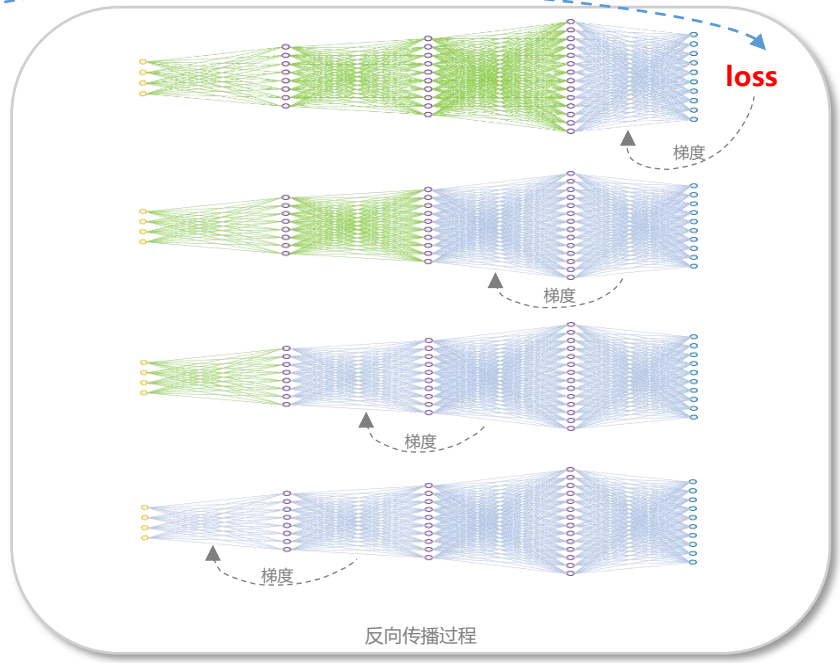
反向传播

反向传播
方向：从后向前
计算：权重和偏置的梯度

根据loss求**梯度**

结合真实值 *y* 求loss





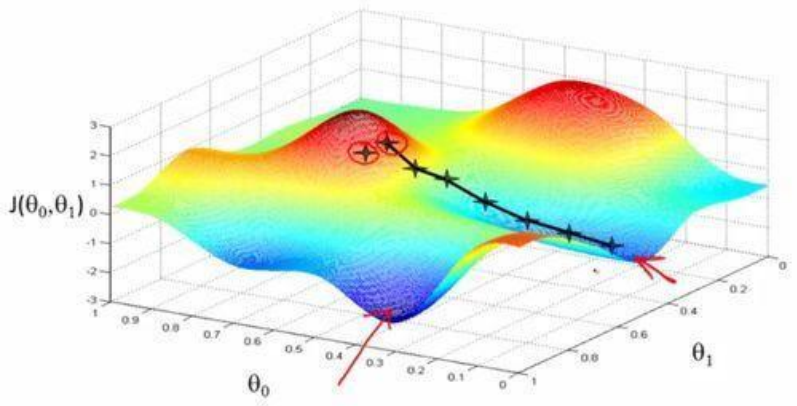
参数更新

学习率learning rate：参数更新的步长，用 η 表示
梯度的反方向是损失函数下降最快的方向

参数更新公式：

$$W' \leftarrow W - \eta \frac{\partial L}{\partial W}$$

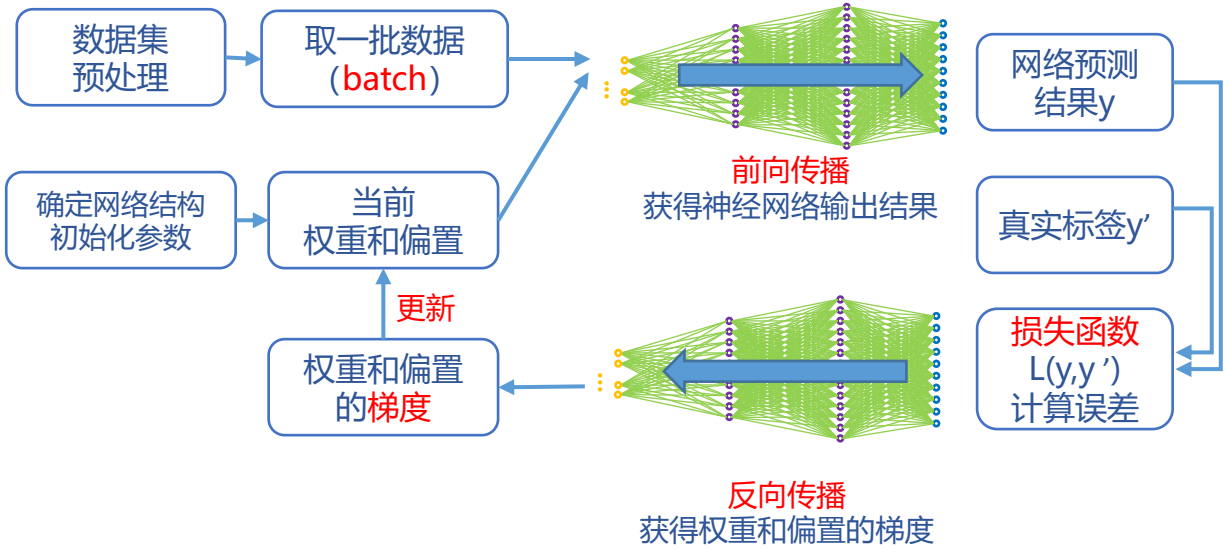
$$b' \leftarrow b - \eta \frac{\partial L}{\partial b}$$



注意：本次课程仅对网络训练流程进行简要介绍，后面课程中会进行更加详细的介绍，包括损失函数计算，梯度公式推导，相关的面试题等等

神经网络训练总览

使用随机梯度下降法对神经网络进行训练，迭代更新神经网络的参数



超参数

超参数 hyperparameter

描述模型的参数，用于控制算法行为。

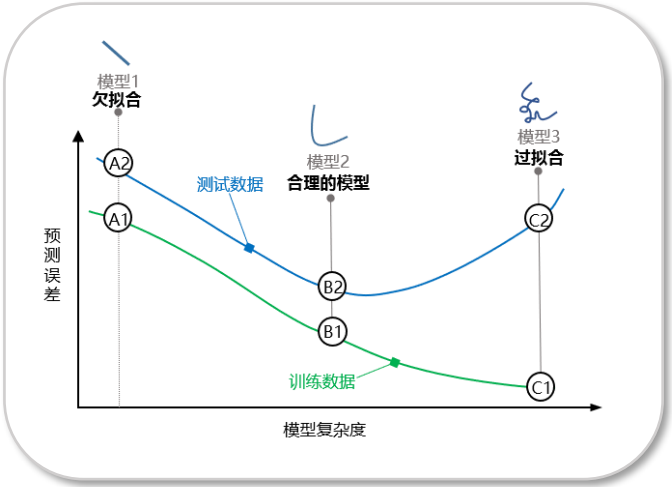
参数首先是一个变量，超参数也是；

验证集就是用于学习超参数的数据子集。

常见的超参数举例：

1. 神经网络的层数 L
2. 每一个隐藏层中神经元的个数 j
3. 神经元激活函数的种类
4. 权重初始化的方法
5. batch的大小 batch-size
6. 学习的轮数 epochs
7. 损失函数的选择
8. 训练集规模
9. 学习率 η
10. 正则化参数 λ

验证集验证



✓ 每经过一次或多个epoch后，可以用验证集测试当前模型的精度

✓ 验证集就是用于调整超参数的数据子集；

— 举例 —

症状：在验证集上表现出训练误差小，但测试误差大，表现为泛化误差大。

诊断：模型训练过拟合。

药方：需提前终止训练（early stopping），调整epochs，记录模型合理时的epochs。