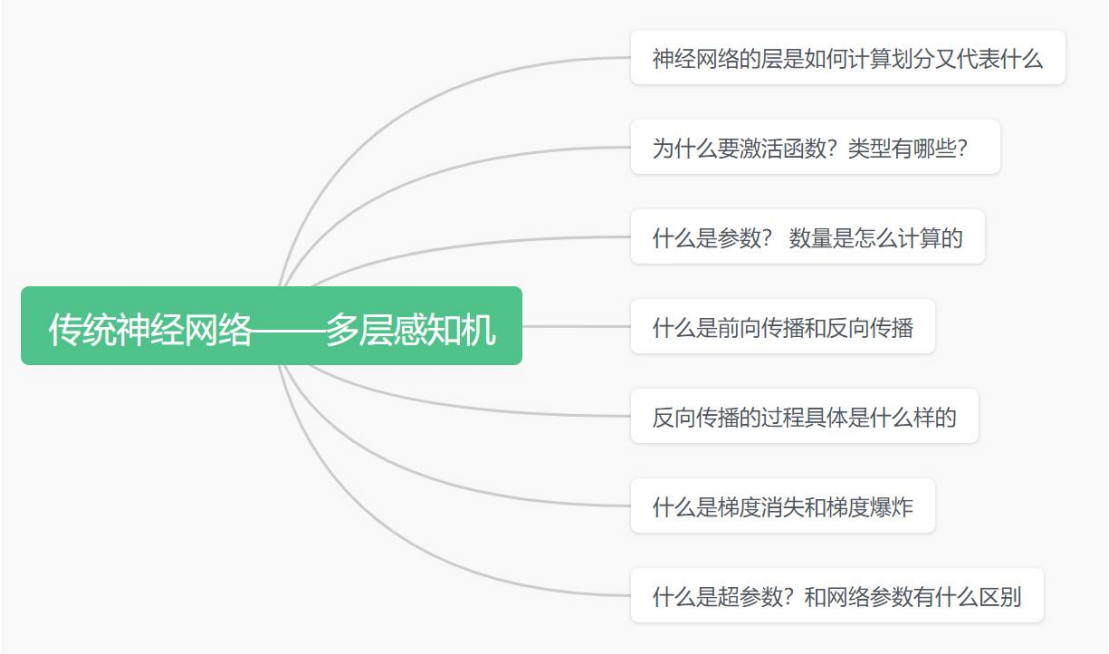
## 问题




## 从线性回归到多层感知机

上节我们介绍了最基础的传统神经网络——线性回归，讲述了神经网络最基础的几个部分，模型创建，数据收集，参数初始化，损失函数，优化算法以及循环训练。后面的神经网络就是在此基础上不断地增补优化，目的就是为了更好地提取特征。

线性回归里我们利用了房价预测这一例子来初步介绍了特征是什么，就是可以影响最终结果的数据。而在房价预测例子中，我们做了很多简化，而且输入特征是可以直接量化的，比如建成时间，和市中心距离等，而且最终的房价也是利用这些特征做了一次简单的线性组合。但我们想象一下，对于“指鹿为马”这一图片识别的场景，我们输入的不是有无角，角的长度，毛皮花纹和颜色的类型，而是一张由一个个数字像素点组成的图片，，这些像素点是特征，不过不是我们所能理解的“特征”，所以研究人员就在输入层和输出层之间增加了更多的层，用以从这些基础原始特征里面一层层提取出抽象的特征，抽取出类似“有无角，角的长度，毛皮花纹和颜色的类型”这些特征，最终根据这些高级抽象特征来输出最终的结果。

# 模型创建

## 全连接层

上节我们创建了线性回归的神经网络模型，包括输入层和输出层，由于输入层并不涉及计算，按照惯例不算到神经网络的层数里。所以，线性回归是一个单层神经网络。

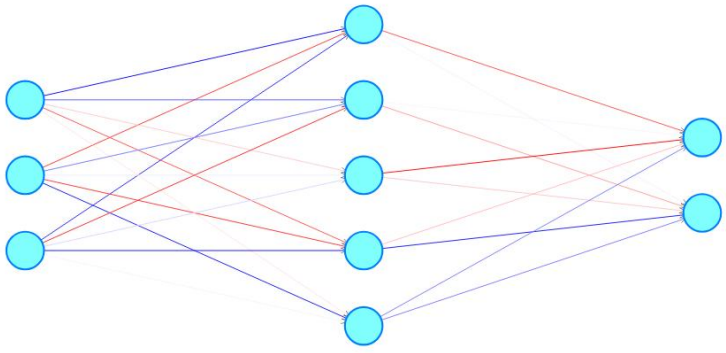
多层神经网络在单层神经网络的基础上引入了一到多个隐藏层（hidden layer）。隐藏层位于输入层和输出层之间。隐藏层的本质就是对输入数据的特征进行提取变换，深度学习网络设计的核心也是寻找更有效的“隐藏层”，能够更高效地提取挖掘特征。

多层神经网络有如下结构：

输入层（Input layer）：将要输入到神经网络的原始数据特征，可以是上节讲到的影响房价的特征向量，也可以是鹿或者马的图片（多层矩阵），也可以是人的声音或语言文字信息，总之，包含一切可以表示信息特征的数据形式。但值得注意的是输入层一般不计入神经网络的总层数里。

输出层（Output layer）：数据特征在神经网络中不断地提取抽象，最后在输出层输出期望的结果。输出层一般是全连接层。

隐藏层（Hidden layer），简称“隐层”，是输入层和输出层之间众多神经元和链接组成的各个层。隐层可以有一层或多层。隐层的节点（神经元）数目不定，但数目越多，网络的非线性越显著，从而神经网络的强健性（robustness）也越显著，与此同时，计算复杂度也会大大提升。隐藏层的本质也是数据特征，只是对上层特征进行了提取组合优化。




输出层中的神经元和上层中每个输入都连接。因此，这里的输出层又叫全连接层（fully-connected layer）或稠密层（dense layer）。

从图中可以看到，第 N 层的每个神经元和第 N-1 层的所有神经元相连（这就是 full connected 的含义），第 N-1 层神经元的输出就是第 N 层神经元的输入。所谓的全连接层就是在前一层的输出的基础上进行一次 Y=Wx+b 的变化(不考虑激活函数的情况下就是一次线性变化，所谓线性变化就是平移(+b)和缩放的

组合(*w)。

模型创建后，同样再思考第一个问题：输入是什么？房价预测例子中，我们借用影响房价的特征来说明输入，这次我们用更一般的方式来说明。多层感知机里的输入特征是 x∈R^1×d 的向量（房价预测中 d=3)，这里就是说输入是 d 的特征向量,然后经过第一个隐藏层的特征变换，数据特征变为了 x∈R^1×5 的特征向量，最后输出结果。

若是图片这种 2 维的矩阵，则可以先展开成 1 维的向量，然后再送到下一层中。再进一步思考，网络训练时经常采用的批量训练，输入和特征又该如何表示？后面学习卷积神经网络时再介绍。

再来考虑第二个问题，这个函数模型里的参数数量是多少？我们知道，参数用来将上一层的所有神经元加权计算得到下一层的每个神经元，以输入层到第一个隐藏层为例，输入层的神经元数为 3，隐藏层的神经元数为 5，输出层的神经元数为 2，则第一层的参数量为 3*5=15，第二层的参数量为 5*2=10，也就是说每层网络的参数是本层神经元数乘上上层神经元数（注意这是全连接层网络的计算方式。）

## 激活函数

上面我们从神经网络的整体架构出发，探讨了多层感知机和线性回归的联系和改进。如果仅是线性网络的叠加，那么还会存在一个问题，即再多的线性网络都可以等效为一个线性网络。我们以 2 个线性网络为例（第一个线性网络的输出作为第 2 个线性网络的输入），即：

$$Y^1 = W_1X + b_1$$

$$Y = W_2Y^1 + b_2$$

第 1 个线性网络层的权重参数和偏差参数分别为$W_1 \in R^{h\times d}$和$b_1 \in R^{h\times 1}$，第 2 个线性网络层的权重和偏差参数分别为$W_2 \in R^{q\times h}$和$b_2 \in R^{q\times 1}$。最开始输入的维度$X \in R^{d\times 1}$，第 1 个线性网络层的输出特征维度是$Y^1 \in R^{h\times 1}$，第 2 个线性网络层的输出特征维度是$Y \in R^{q\times 1}$

将上述公式联立可得：

$$Y = W_2Y^1 + b_2 = W_2(W_1X + b_1) + b_2 = W_2W_1X + W_2b_1 + b_2$$

可以看到，即使中间加了一层线性网络，输出和最初输入之间仍是线性相关，其和单层神经网络仍是等价的，差别仅在于权重参数做了乘加运算，这样看来，中间加线性网络层的作用和单层神经网络是等效的。为了解决这个问题，研究人员提出了神经网络的非线性单元——激活函数，来为网络模型提供非线性特性。

激活函数作为一种非线性因素加入到网络中去，能够大大提升网络表达能力。如图所示，卷积神经网络中常用的激活函数包括 Tanh 函数、Sigmoid 函数、可

调参线性整流单元(Parametric Rectified Linear Unit, PReLU)函数和线性整流单元(Rectified Linear Unit, ReLU)函数函数等。

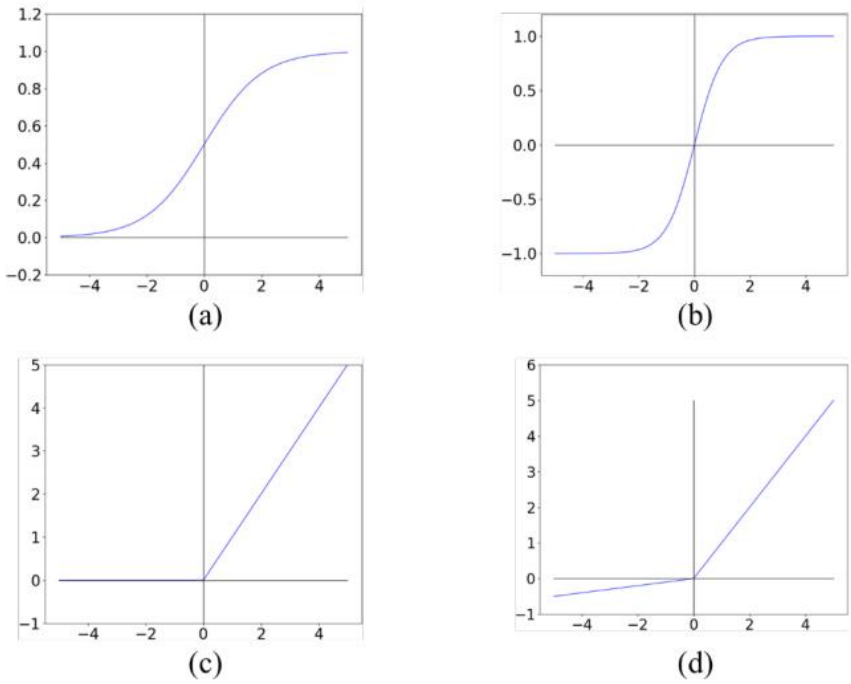



图 2-10 四种激活函数图像 (a)Sigmoid (b)Tanh (c)ReLU (d)PReLU

（1）Sigmoid 函数

Sigmoid 函数是一种被广泛加入到网络中去的 S 型激活函数，其表达式如下所示：

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

该函数可以将属于$(-\infty, +\infty)$的输入 x 映射到(0,1)区间内，函数的输出区间小，因此数据在前向传播的过程当中能够聚合不发散，而且输出区间(0,1)还可以作为概率的表示范围。但是 Sigmoid 函数也有其缺点，如图所示，在函数的饱和区域十分容易产生梯度消失现象，增加网络的训练难度。

（2）Tanh 函数

Tanh 函数是 Sigmoid 函数的变体，其表达式如下所示：

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

式中，x 的范围是$(-\infty, +\infty)$，对应的输出范围是(-1,1)，Tanh 函数以零为中心，相比 Sigmoid 函数收敛速度更快，但是仍然存在梯度消失的问题

（3）ReLU 函数

ReLU 函数是一种最常用的激活函数，其表达式如下所示：

$$\text{ReLU}(x) = \begin{cases} 0, (x < 0) \\ x, (x \geq 0) \end{cases}$$

ReLU 函数的运算非常简单，直接取 0 或 x 的最大值即可且不包含指数运算，间接提升了机器的运行效率。而且该函数与人体神经元细胞对信号的响应非常相似，积极回应正向信号，消极回应负向信号。虽然在 ReLU 函数的正区间内不存在饱和区有效避免了梯度消失现象，但是当输入为负数时输出为 0，神经元被抑制。

（4）PReLU 函数

PReLU 函数针对 ReLU 函数的负值输入完全被抑制问题进行了改进，其表达式如下：

$$PReLU(x) = \begin{cases} ax, (x < 0) \\ x, (x \geq 0) \end{cases}$$

式中，a 是一个可以在训练中学习的参数，并不是预先设置的常数

我们将在后面的代码中对比看一下几种激活函数的差异。

## 小结

多层感知机就是含有至少一个隐藏层的由全连接层组成的神经网络，且每个隐藏层的输出通过激活函数进行变换。多层感知机的层数和各隐藏层中隐藏单元个数都是超参数。以单隐藏层为例并沿用本节之前定义的符号，多层感知机按以下方式计算输出

$$H = \emptyset(XW_h + b_h)$$

$$O = HW_o + b_o$$

其中∅表示激活函数。

# 模型训练

模型训练的本质就是利用已知的数据获得想要的参数。具体过程就是，利用数据集中已有的输入和当前的参数，计算得到一个预测结果，然后和数据集中的真实结果（标签）作比较，利用误差去优化模型参数，不断训练，直到达到设置的训练次数或者预测结果和真实结果的误差在一定范围。

## 数据收集

数据是人工智能的基础和核心，本节暂且简单说明一下，卷积神经网络章节再详细介绍。

通过收集大量的真实数据，每组数据包括房子的真实售价和它们对应的面积，房龄和到市中心距离，我们希望在这个数据上面利用上面创建的网络模型进行训

练，学习得到模型参数来使模型的预测价格与真实价格的误差最小。

在机器学习术语里，该数据集被称为训练数据集（training data set）或训练集（training set），房价预测数据里房子的真实售价叫作标签（label），用来预测标签的三个输入叫作特征（feature），特征用来表征样本的特点。一组数据被称为一个样本（sample），包含输入和标签。

假设我们采集的样本数为 n，索引为 i 的样本的特征为$\mathbf{x^i}$，标签为$\hat{\mathbf{y}}^{\mathbf{i}}$。对于索引为 i 的房屋，线性回归模型的房屋价格预测表达式为

$$\hat{y}^i = x_1^i w_1 + x_2^i w_2 + x_3^i w_3 + b$$

也就是说输入数据的维度是 n*3*1,输出维度是 1，中间无隐藏层，权重参数维度为 1*3。

## 参数初始化

前面模型创建章节建立好模型后，却并不知道参数是多少，那如何得到预测售价呢？答案就是先手动设置参数的初始值，方法有很多，比如全设置为 0，或者随机初始化。

神经网络参数初始化是深度学习中的关键步骤之一。参数初始化的选择和设置对于网络的训练和性能具有重要影响。在后面我们将进一步讨论。这里我们先选择全设置为 0。

## 损失函数

有了预测模型，知道了影响房价的特征输入和参数，便可以像单价*数量=总价那样计算得到房价了。但问题在于影响房价的特征是凭经验选的，参数更是随机初始化的，并不是数学意义上严格的计算公式。

此时思考一下小孩子在学习辨认鹿和马的过程是怎样的？映入眼帘一只鹿，观察到它的鹿角，皮毛，颜色，四肢，大人告诉这是一只鹿，小孩记住了，再次看到的时候，如果小孩子说是马，大人多次进行纠正，最后针对这些特征便可以辨认出这是鹿，只是我们并不知道大脑进行了怎样的完善。

同样的，对于神经网络的训练，首先根据特征输入和初始的参数，计算出预测结果，然后与真实结果进行比较，得到它们之间的差值。

研究人员提出来损失函数这个概念，损失函数又可称为代价函数或目标函数，是用来衡量算法模型预测结果和真实标签之间吻合程度（误差）的函数。

通常会选择非负数作为预测值和真实值之间的误差，误差越小，则模型越好。本例中选用简单的平方函数，即：

$$L^i(w_1, w_2, b) = \frac{1}{2}(\hat{y}^i - y^i)^2$$

训练模型的目标便是将此损失函数的值尽可能的小。后面再介绍其它的损失函数。通常，我们用训练数据集中所有样本误差的平均来衡量模型预测的质量，在模型训练中，我们希望找出一组模型参数，来使训练样本平均损失最小。

## 优化算法

有了模型和初始参数，我们便得到了预测目标值；有了预测目标值和真实标签值，代入到损失函数便可以得到误差；那么如何根据这个误差去优化模型，就像在辨认鹿的过程中给予的纠正一样呢？神经网络训练使用的就是优化算法。

损失函数即预测值和真实值的误差是关于参数的函数，优化算法就是用来寻找模型参数，使得训练样本的平均损失最小。

当模型和损失函数形式较为简单时，上面的误差最小化问题的解可以直接用公式表达出来。这类解叫作解析解（analytical solution）。本节使用的线性回归和平方误差刚好属于这个范畴。

然而，大多数深度学习模型并没有解析解，只能通过优化算法有限次迭代模型参数来尽可能降低损失函数的值。这类解叫作数值解（numerical solution）。

## 循环训练

设置循环次数，每一个遍历循环过程，都是根据输入的特征计算预测结果，然后求得损失函数值，再利用优化算法更新模型参数。

在达到设定的训练次数或者误差值在一定范围之后训练停止，这时得到一个拟合好的函数模型。面对新的未知的数据，只需要将特征输入到神经网络模型中，便可以像普通函数公式那样计算得到较为准确的输出结果了。

**参考资料**

https://zh.d2l.ai/index.html