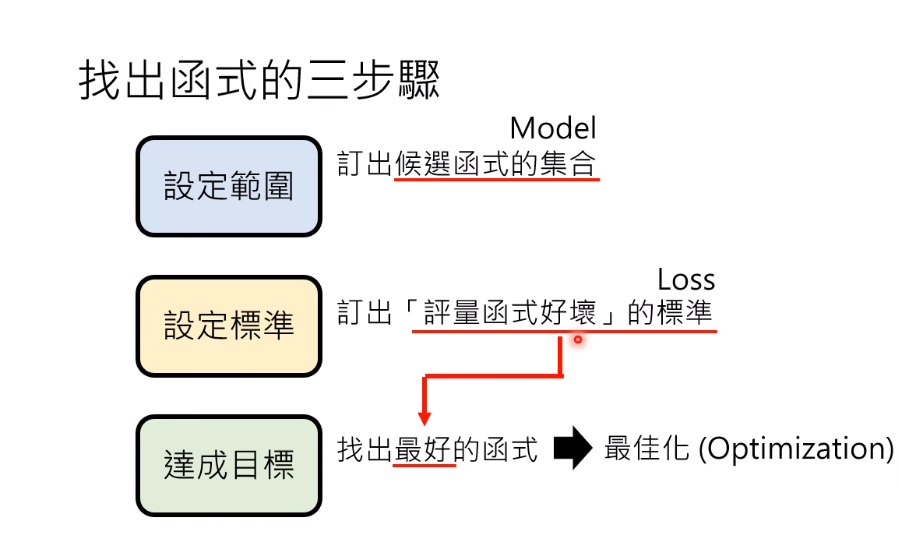
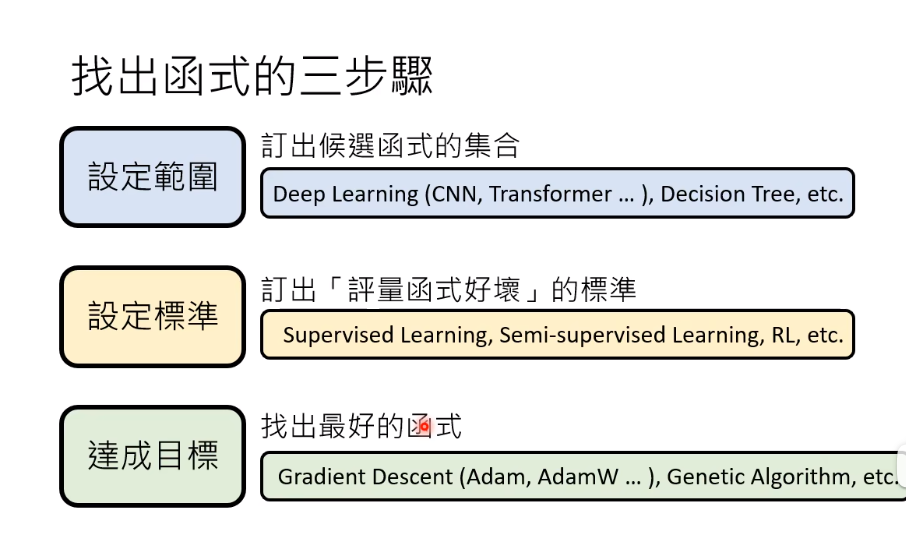
# 机器学习基本原理

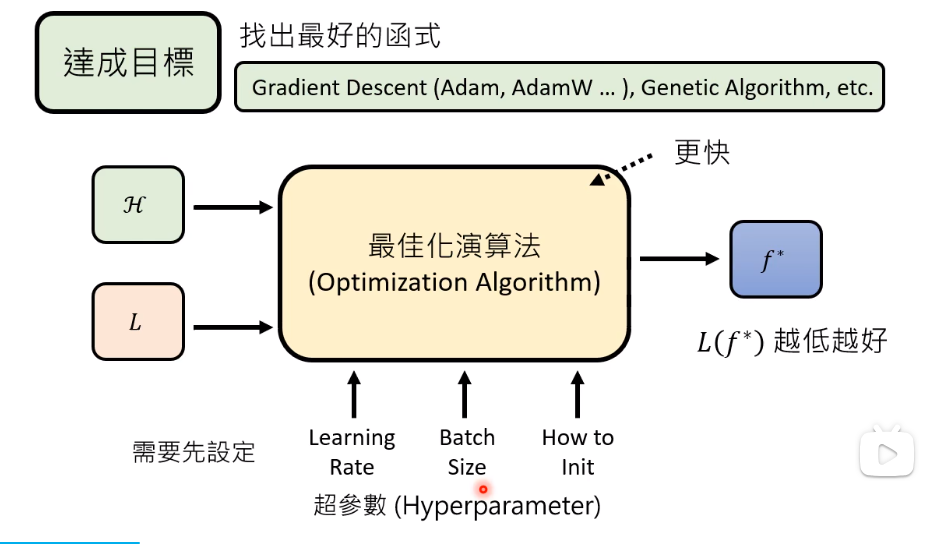
机器学习：机器自动寻找一个函数f，输出通常归为回归（Regression）和分类（Classification）两大类，以及如今变成更复杂的生成式学习。

Structured Learning：结构化学习，又称为Generative Learning生成式学习，生成有结构的物件（如影像、文句）等

机器学习的类别有多种，大致可以分为四种类型：监督学习、半监督学习，无监督学习和强化学习。

寻找基本需要三个步骤：





找到最好的函数，也就是找到最合适的一组参数，参数一部分是不可预测的，一部分是自身设定的，自己设定的参数，也就是超参数。找到一个函数使其在训练集和测试集上都有较好的结果，需要设置合适的函数集合，如深度学习、决策树等。

# 生成式AI

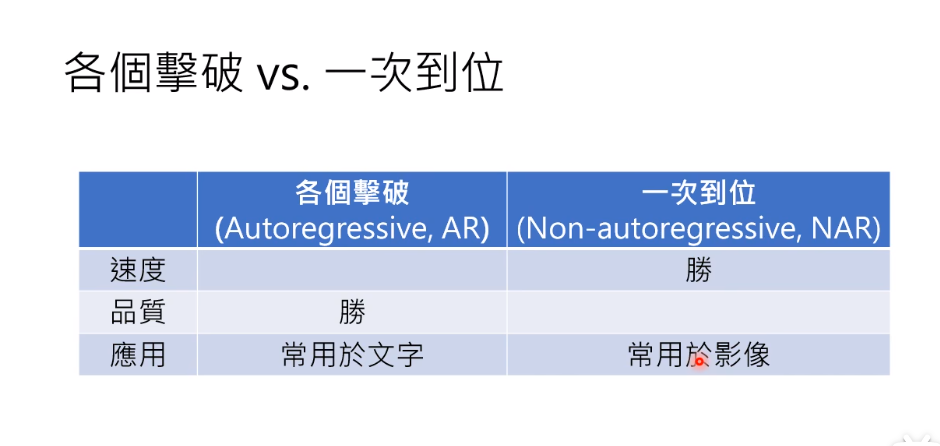
## 2.1生成式学习的两种策略

**Autoregressive Model 自回归模型**

自回归模型通过将生成问题视为一个序列过程来简化生成模型，速度较慢，效果较好。

**Non-autoregressive Model 非自回归模型**

同时生成整个序列，速度较快，精度效果相比较次。



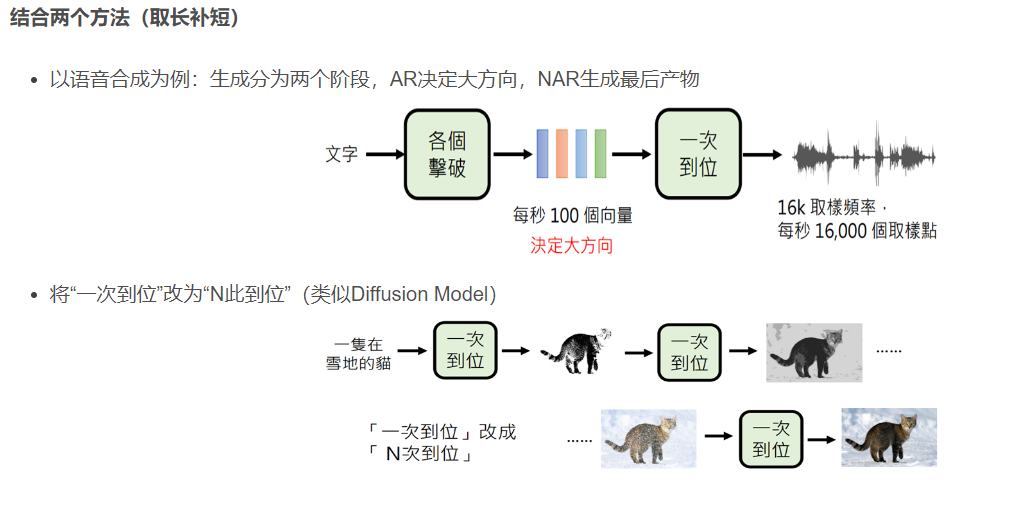
**两者的对比如下：**

在自回归生成模型中，生成数据的过程是逐步进行的。以文本生成为例，模型会从左到右一个字一个字地生成文本，每个字的生成都依赖于前面已经生成的内容。典型的自回归模型包括循环神经网络（RNN）和变换器（Transformer）的解码器部分。

自回归生成模型的优点是生成的数据在语法和连贯性上通常很好，但缺点是生成速度较慢，因为必须等待前一个元素生成后才能生成下一个元素。

相比之下，非自回归生成模型不需要按顺序生成数据。它们可以同时生成整个序列，因此生成速度更快。这通常通过将所有生成步骤并行化来实现。非自回归生成模型的一个典型示例是生成式对抗网络（GAN）中的生成器部分。GAN的生成器可以一次性生成整个图像，而不需要逐像素生成。这使得GAN在图像生成任务中非常有效。

两者结合使用可以组成较好的生成策略。

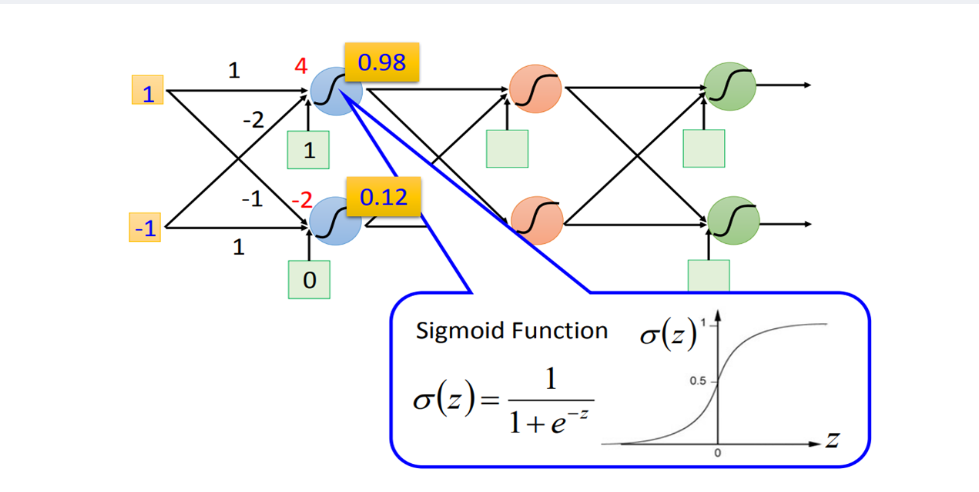


# 深度学习

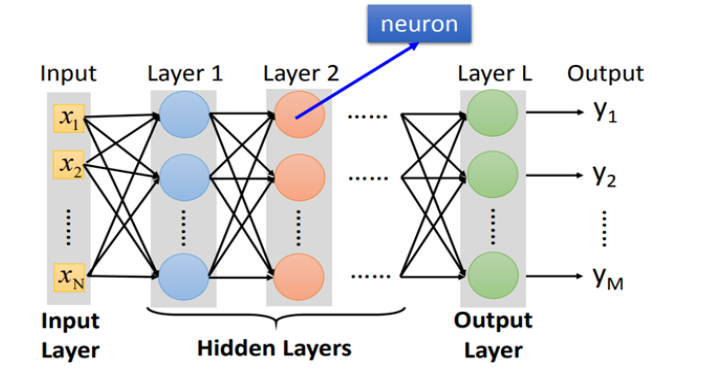
## 3.1 Step1：define a set of function （function即Neural Network）



将logistics regression连接，一个逻辑回归称为Neuron神经元，不同的方法连接Neural Network就得到不同的结构structure，每个logistics regression都有属于自己不同的weight权重，和bias偏移量，weight和bias的集合称为network的函数θ

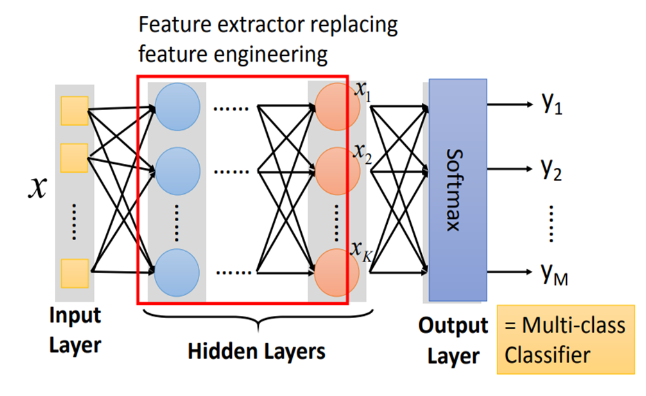
神经元的连接方式有多种，最常见的有全连接，如下：

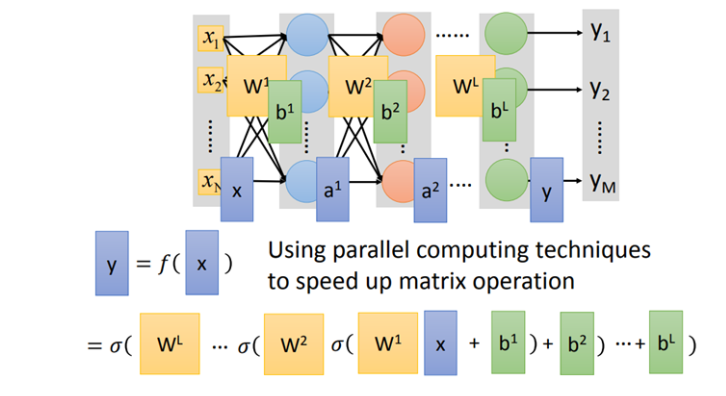
此处的参数是已知的，在参数未知时，则称为一个function set.

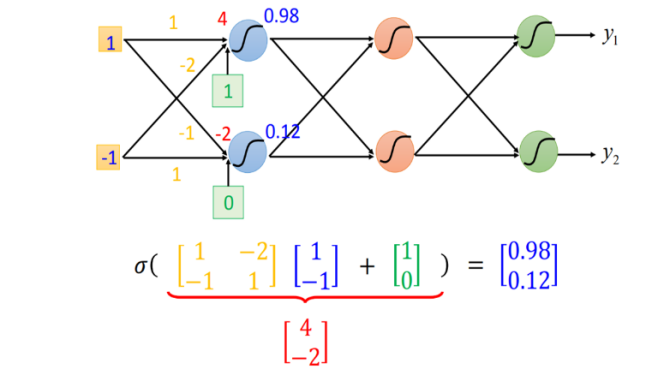
一般化一个神经网络的示意图：

network通过hidden Layers（隐藏层）来提取特征，替代了原来的特征工程（即手动选择特征），在隐藏层最后一层输出的就是新的特征，而输出层就拿着这些新的特征作为输入，通过一个多分类器（softmax函数），得到最后输出y。

Softmax 函数是一种常用的激活函数，通常用于多类别分类问题中。它的原理是将一个向量的元素转化为概率分布，使得每个元素的取值范围在0到1之间，并且所有元素的和为1。Softmax 函数的作用是将原始的线性得分转换为概率分布，使得模型能够对不同类别进行概率预测。在多类别分类问题中，Softmax 函数可以帮助模型选择概率最大的类别作为预测结果。

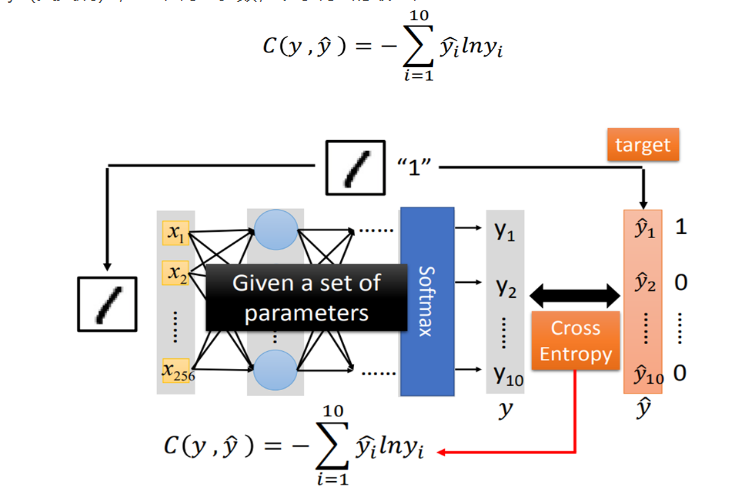


每一层的运算可以用矩阵来表示，称为Matrix operation，neural network的运算可以等同于一连串的matrix操作。



## 3.2 Step2：goodness of function

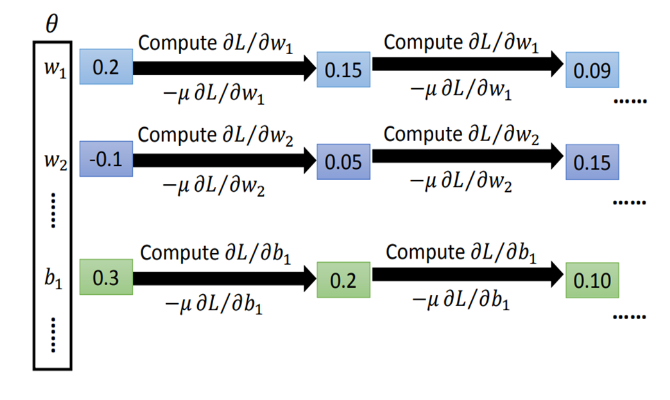
例如输入手写数字“1”，得到一组十维限量输出yi（理想输出是y^ 1000000000），计算y与y^的Cross Entropy（交叉熵），手动调整参数，以求得C的最小值。



将训练集输入后计算Cost，将每一个function的Cost求和，得Loss值，选择function set 中Loss值最小的function即可。

## 3.3 Step3：pick the best function

利用梯度下降方法优化参数



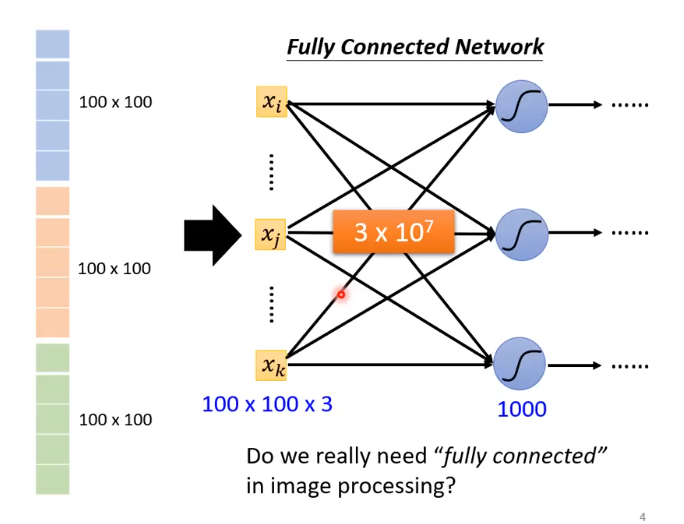
可以使用工具来计算，例如TensorFlow。

# 卷积神经网络

## 为什么利用卷积神经网络CNN来做图像处理？

当我们用fully connect feedforward network（全连接前馈神经网络）来做[图像处理](https://so.csdn.net/so/search?q=%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_43328313/article/details/_blank)的时候，往往我们会需要太多的参数，假设有一张100 \*100的彩色图，把它拉成一个向量，它有100 \*100\* 3的像素（pixel）。

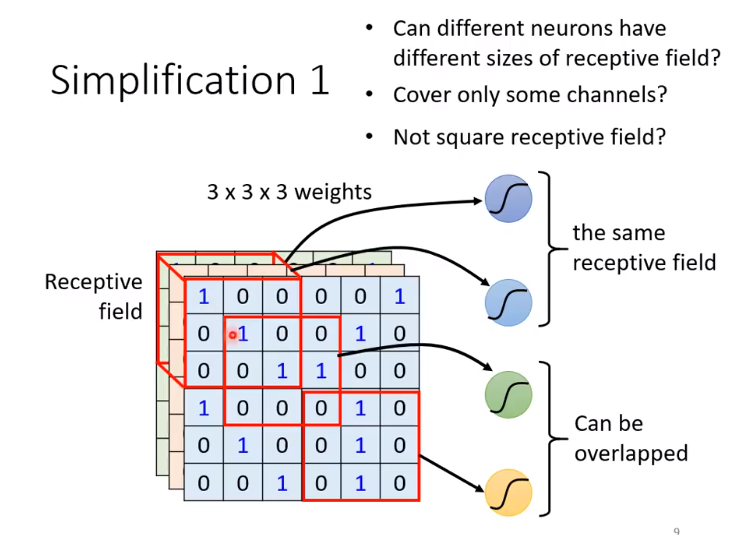
如果是彩色图的话，每个pixel需要三个value来描述它，就是30000维(30000 dimension)，那input vector假如是30000dimension，那这个hidden layer假设是1000个neural，那么这个hidden layer的参数就是有30000 \*1000，那这样参数就太多了。

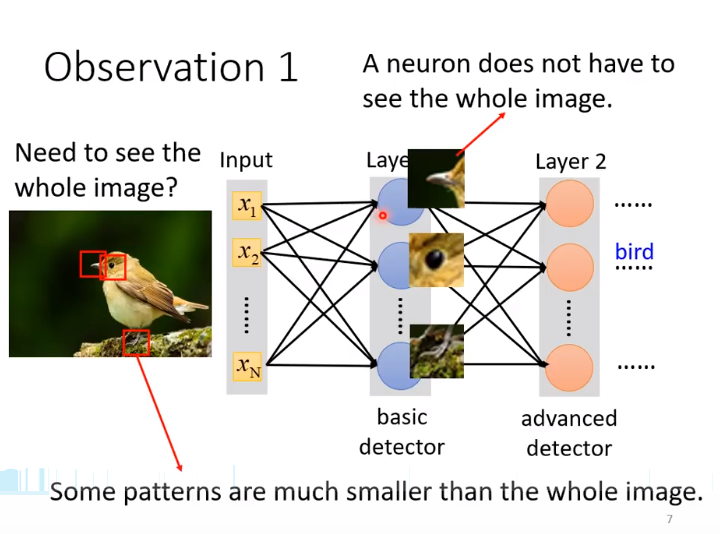
那么CNN做的事就是简化neural network的架构。我们根据对图像的认识，某些weight用不上的，我们一开始就把它滤掉。不是用fully connect feedforward network，而是用比较少的参数来做图像处理这件事。所以CNN比一般的DNN还要简单。

CNN就是用power-knowledge去把原来fully connect layer中一些参数拿掉。

## 为什么可以用比较少的参数可以来进行图像处理（CNN中的power-knowledge）？

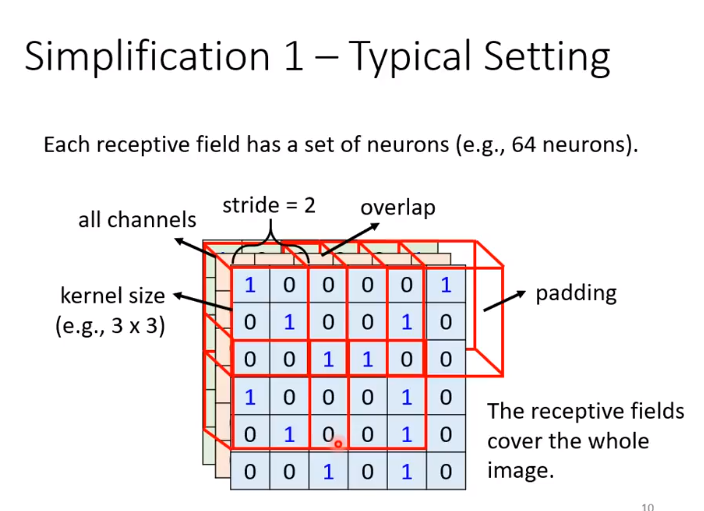
1. 特征图像远小于整个图像（small region：大部分的pattern比整张image小）



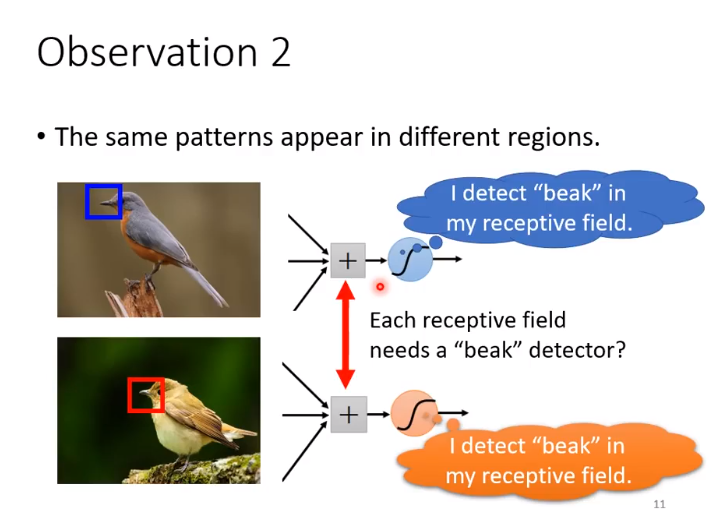


解决办法：可以通过选取（感受特征区域）分配神经元用于计算。

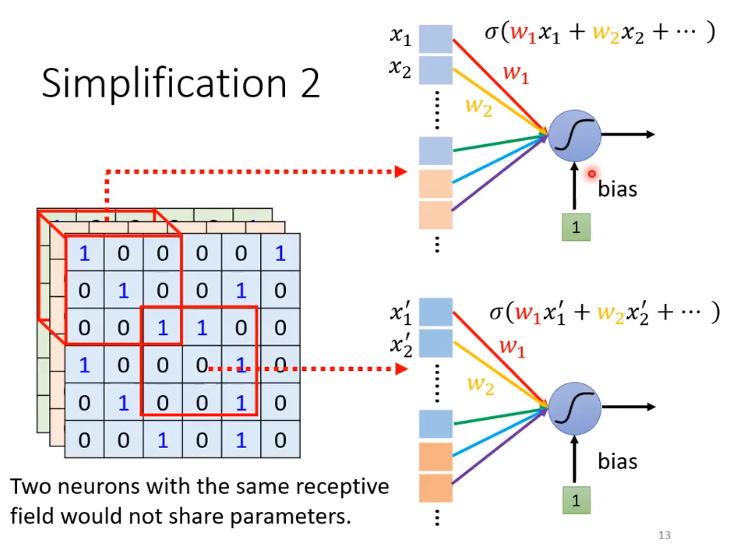
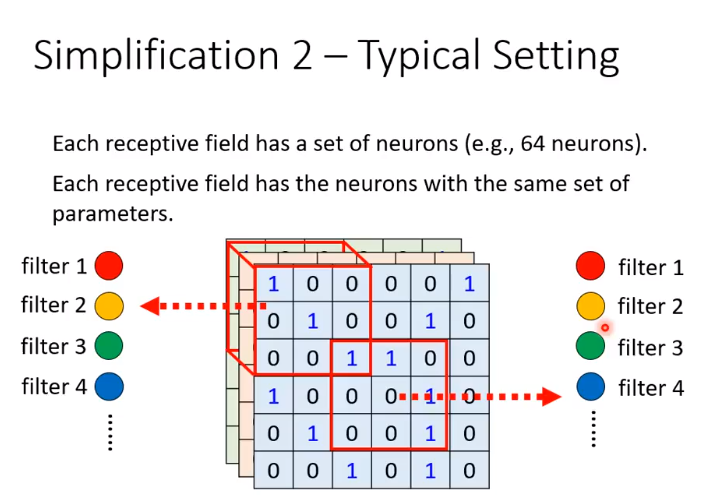
在这里列出一种经典的处理方式：



选取前进格数，到边缘出不够组成新矢量时可以采用不同方法填充，例如填0.

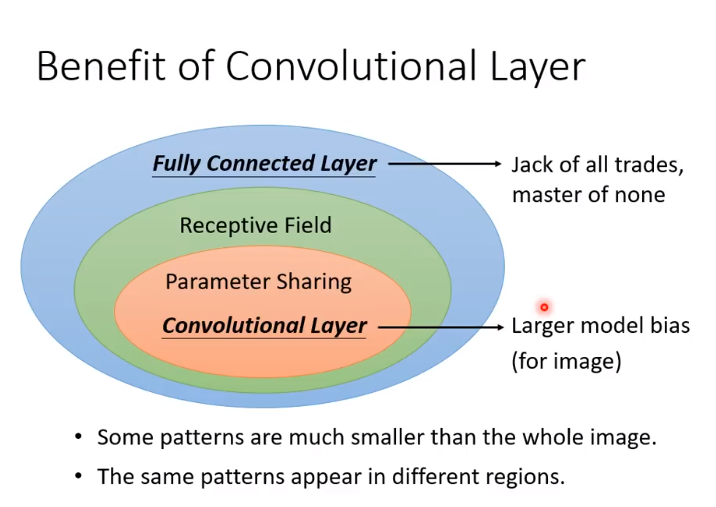
1. 同样的特征会出现在不同的区域（same patterns：相同的pattern可能出现在image不同的地方，但代表的是相同的含义）

解决方式：共享参数

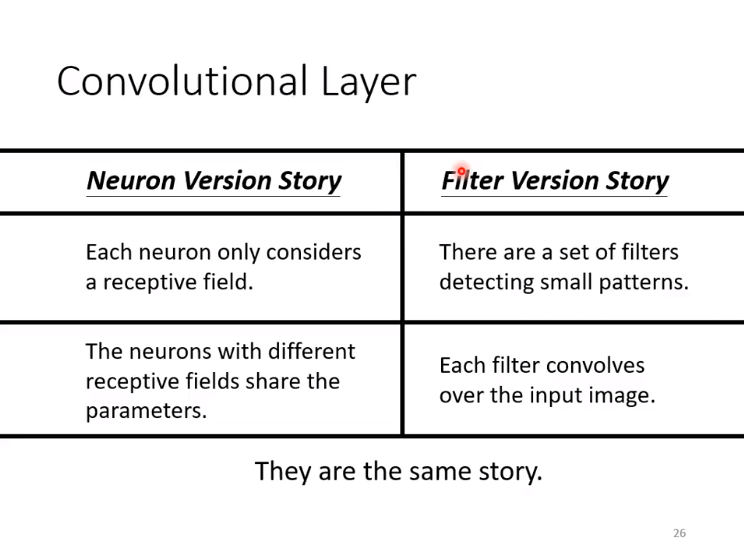


不同向量计算使用相同的神经元，也就是权值与偏差相同。

在这里列出全连接与卷积的层次，卷积虽然偏差值较大，但在图像处理上有较好的效果。



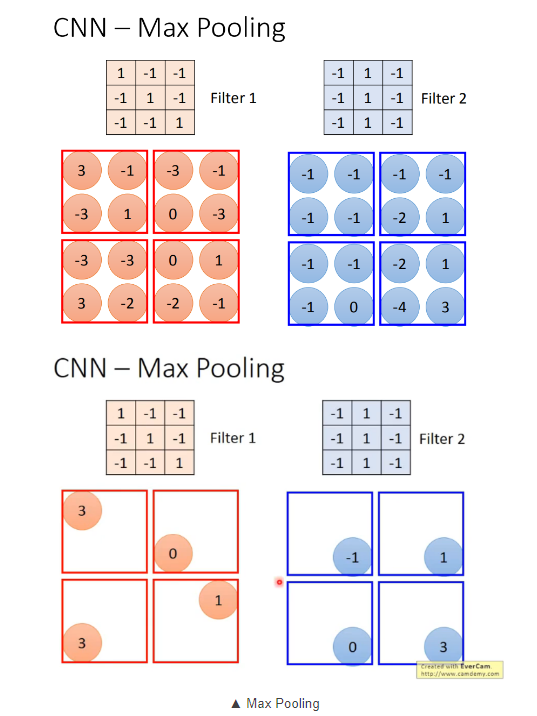
上述的处理也就是卷积的过程，用filter扫描过一张图片就得到了卷积层。



1. 像素缩放不会对图像造成太大影响（subsampling：下采样一张image，得到小的image可能不会影响对原始image的理解），也就是池化过程。

池化本质上的作用是：缩小图像，减少特征，也就是减少计算量，卷积和池化过程可以交替重复用于图像处理。

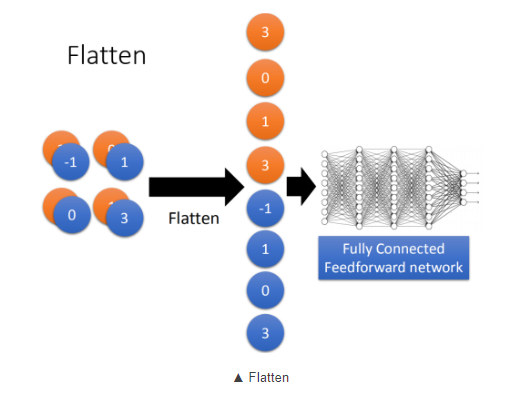
在此处把四个值合为一个值，可以取最大值，也可以取平均值等。



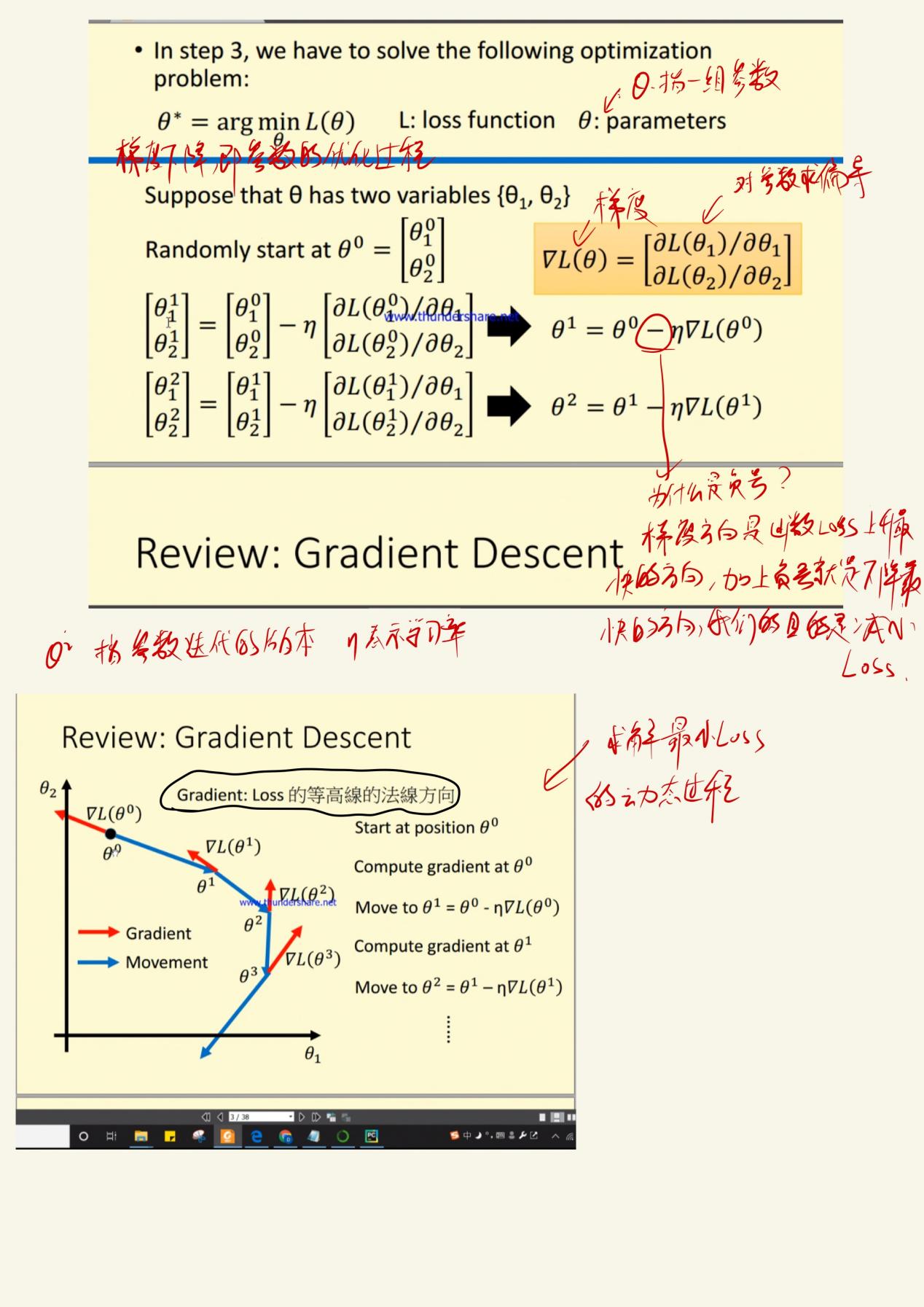
## CNN的完整过程

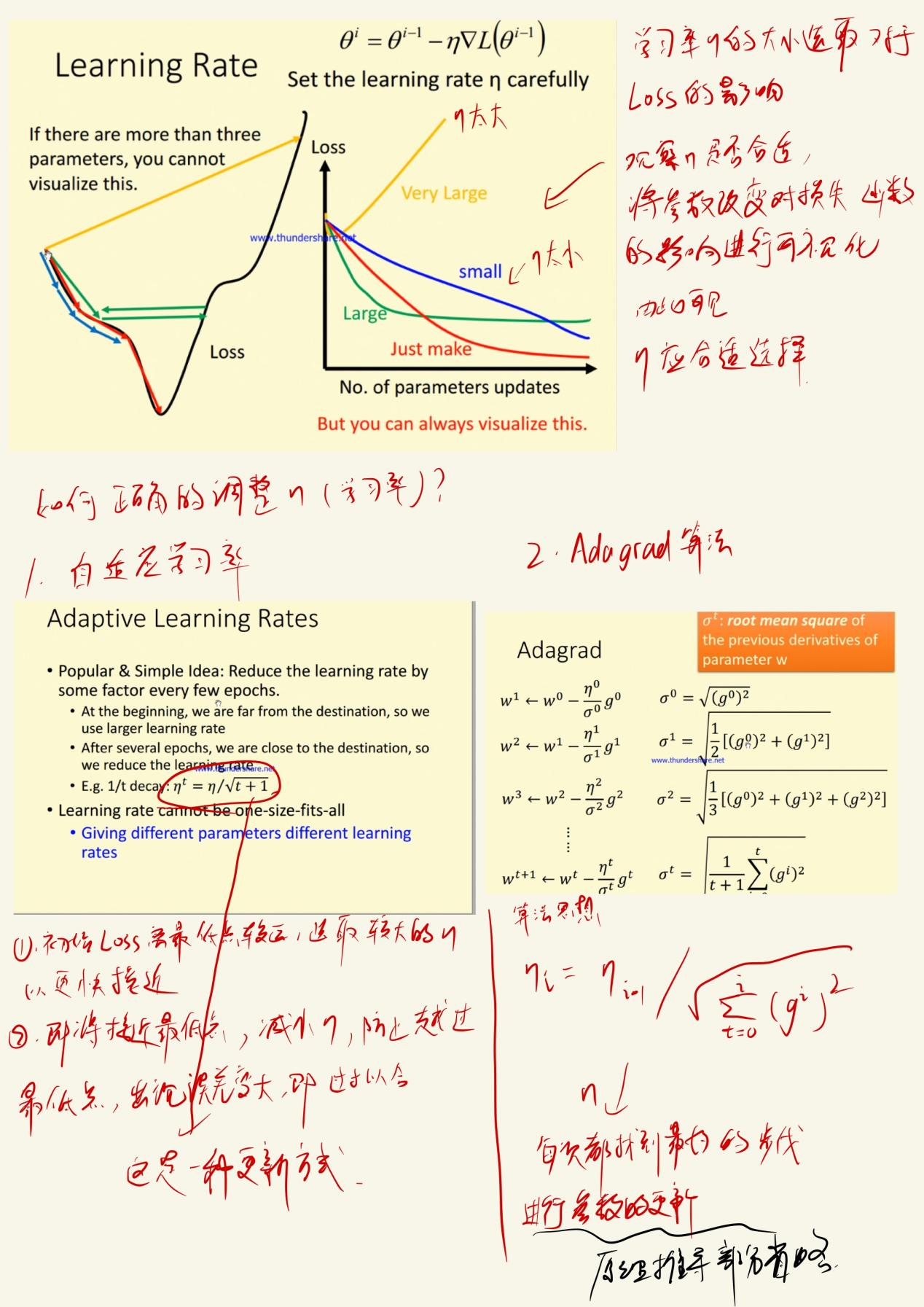
压平（Flatten），就是把多维的输入一维化，常用在从卷积层到全连接层的过渡。

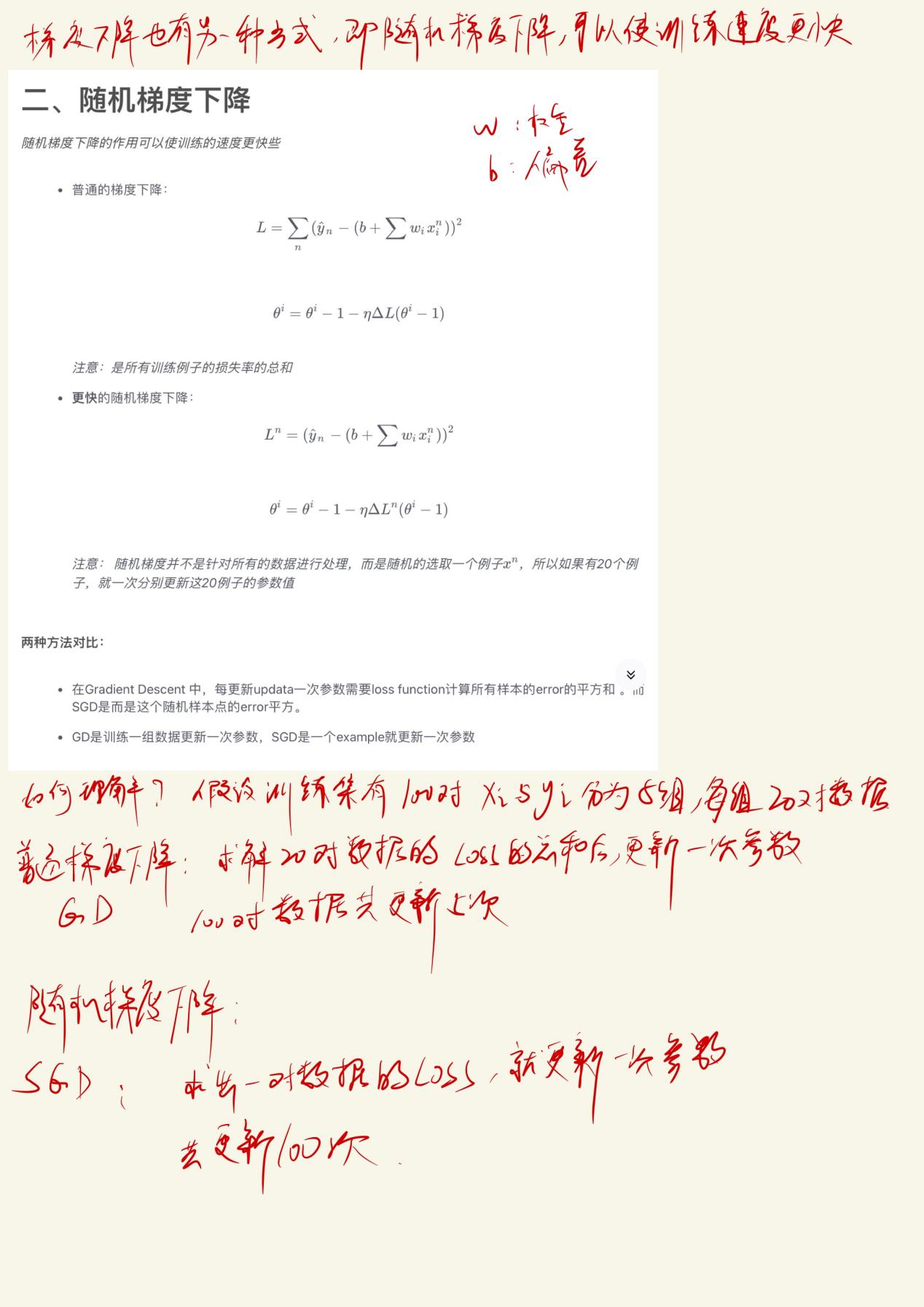
flatten就是feature map拉直，拉直之后就可以丢到fully connected feedforward netwwork。

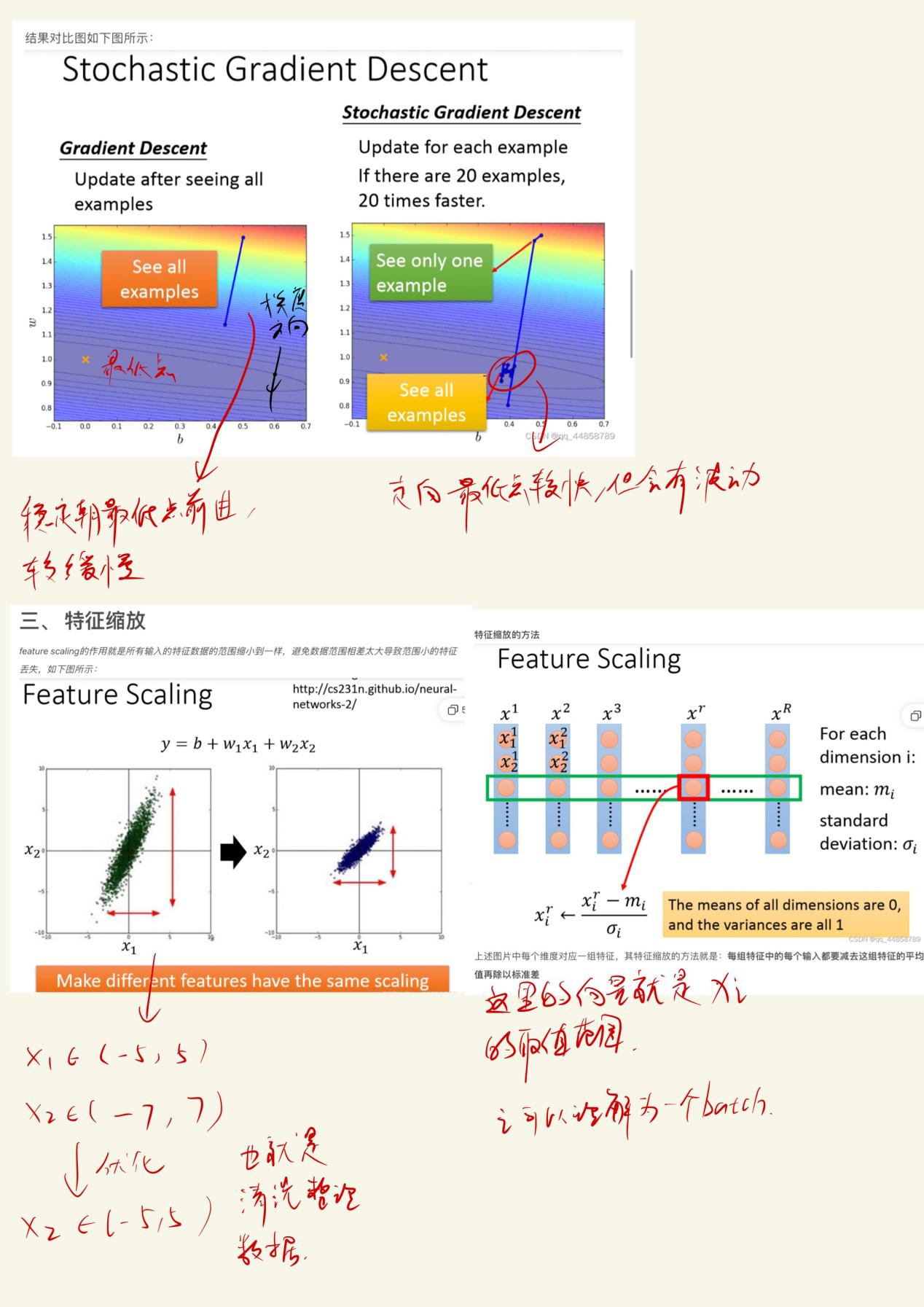


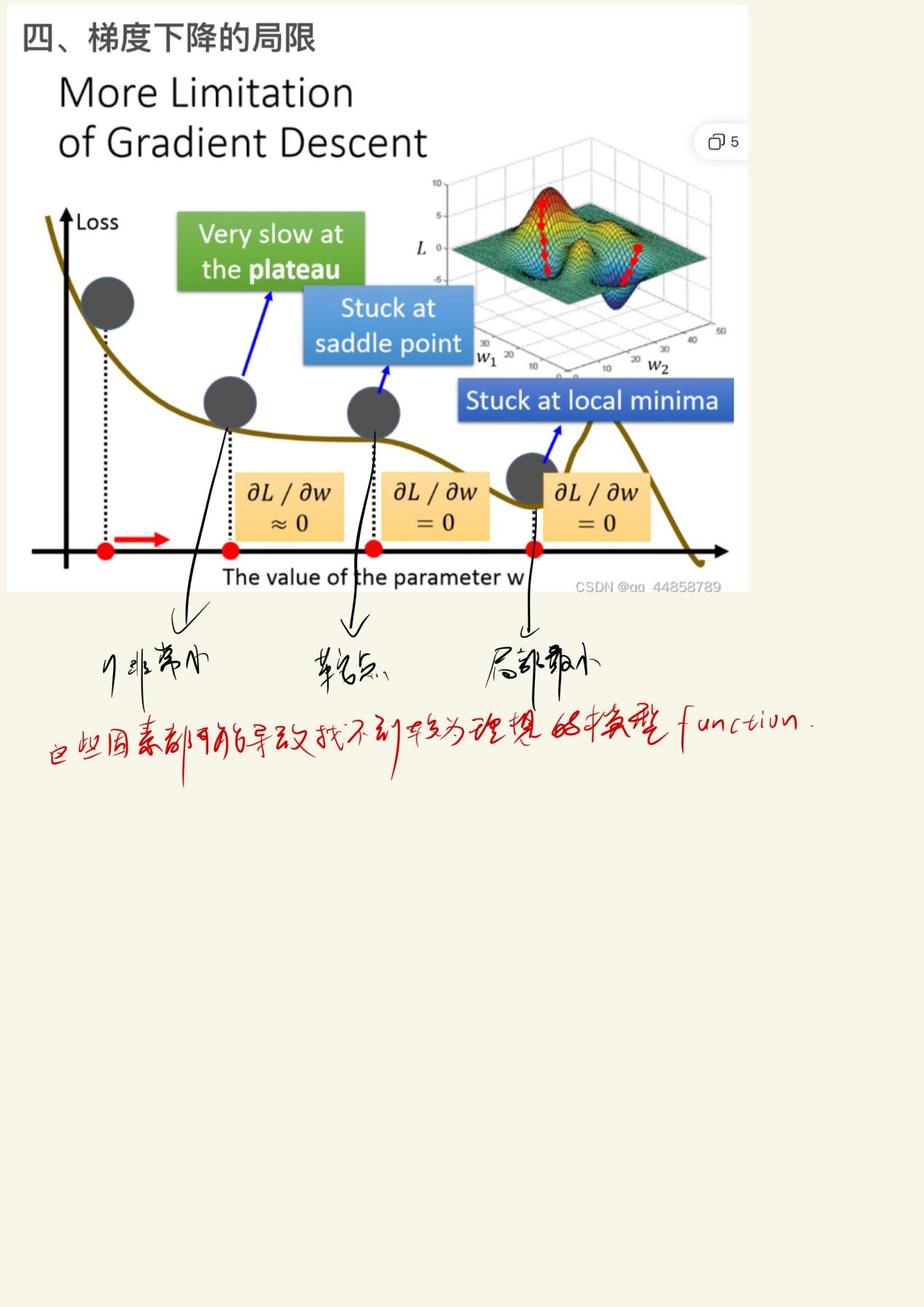
# 梯度下降







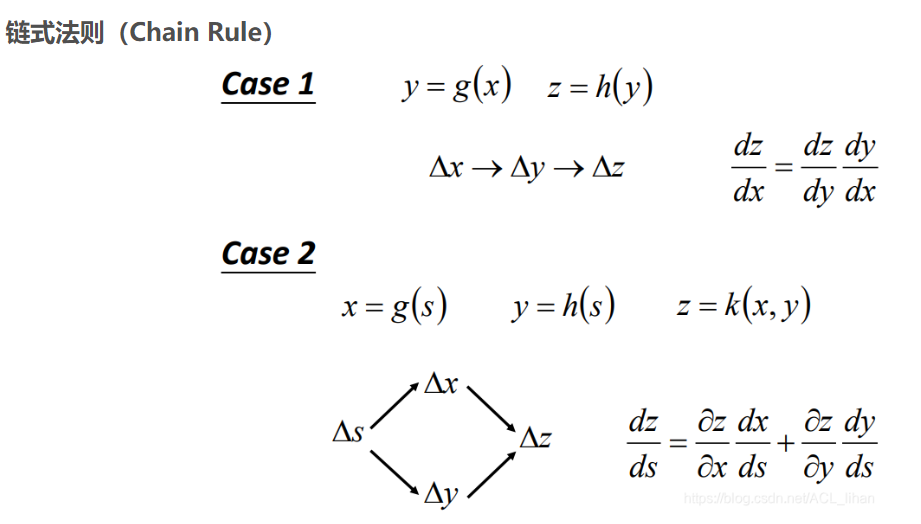




# 反向传播（Backpropagation）

## 使用反向传播的原因

gradient descent用来更新参数，使Loss Function最小化，但是面对Neural Network可能上百万的参数，做gradient descent会很困难。所以需要引入反向传播（Backpropagation）使gradient descent 做得更有效率一点。反向传播就是梯度下降的一种算法。

反向传播（Backpropagation）用到了链式法则（Chain Rule）的知识：

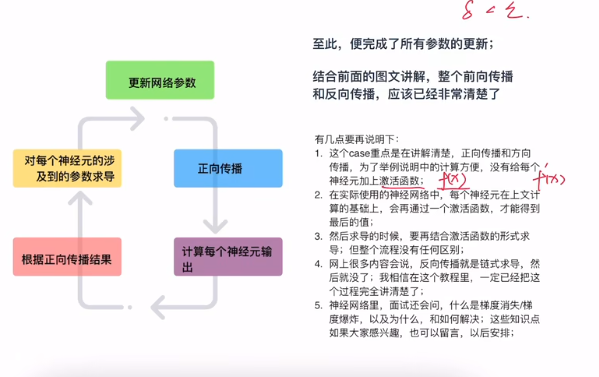
损失函数Loss，代价函数Cost，总体损失函数Total Loss区别

损失函数：定义在单个训练样本上的，单指一个样本的误差

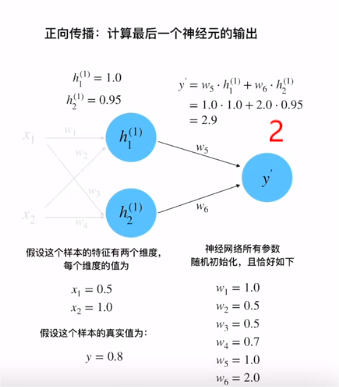
代价函数：定义在整个训练集上的，即所有样本的误差的总和的平均

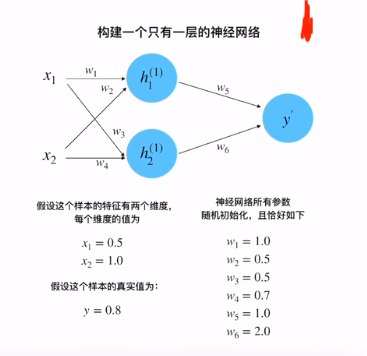
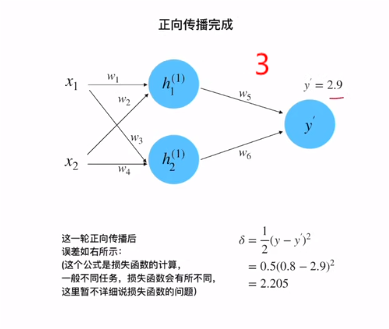
总体损失函数：定义在整个训练集上的，即所有样本的误差的总和，也是我们反向传播需要最小化的值

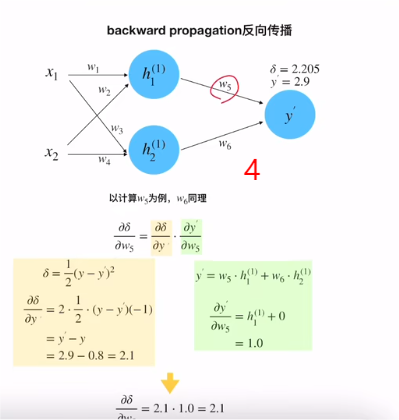
在这里用具体数值计算的方式来描述反向传播来调参的过程。

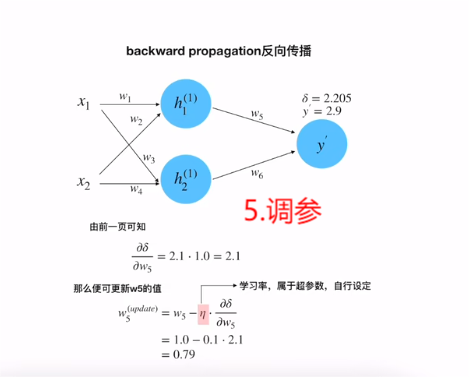
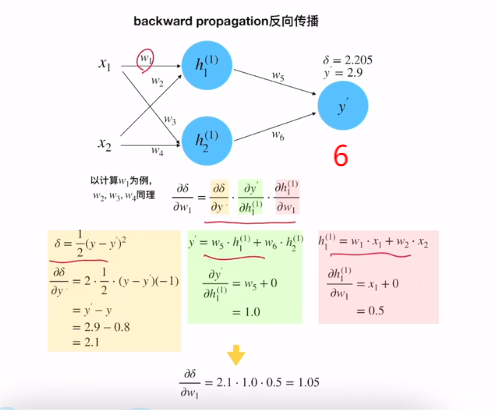
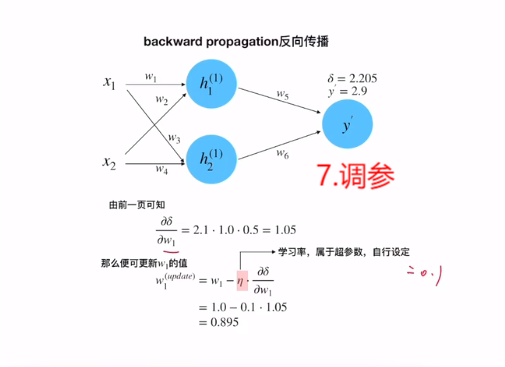
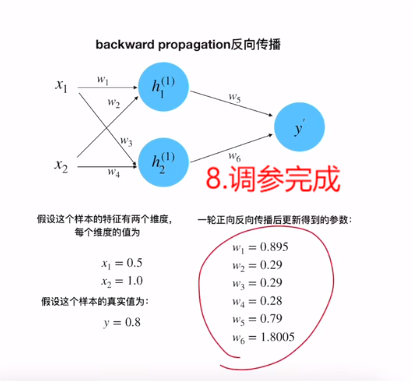
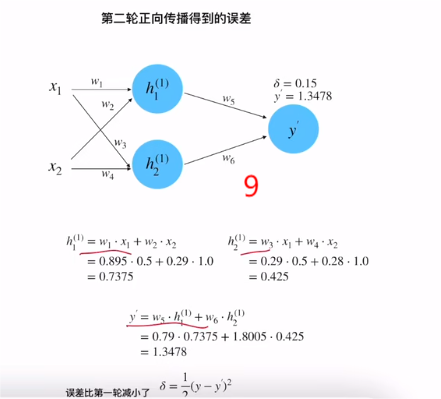


## 反向传播演示

一组特征x输入，通过一个神经元后会得到一个预测的标签（输出y），这叫做正向传播的过程，预测y与实际y的差值带入损失函数Loss的表达式得到一个值，这个值用来评价参数w与b的好坏，反向传播就是损失函数L对每个神经元涉及到的参数求导的过程，对参数的求导需要用到链式法则，求导后会得到一个值C,原参数减去C\*学习率=新参数，这样就完成了对参数的更新，对每个参数都进行这样的计算，一整个反向传播的过程就算完成了，下面将给出计算演示。







经过第一轮的反向传播，第二轮再带入同样的数据进行计算时，发现Loss值变小了。这里并未涉及到计算结果经过激活函数，但有激活函数也是同样的过程，即损失函数对激活函数求导。