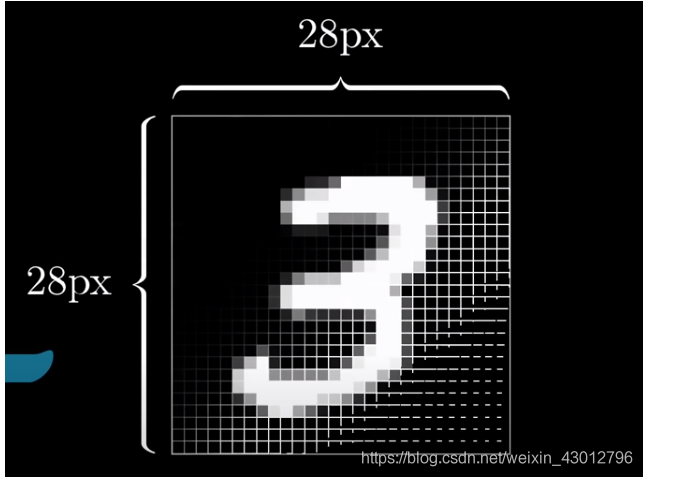
1. 如何识别像素28\*28的手写数字3



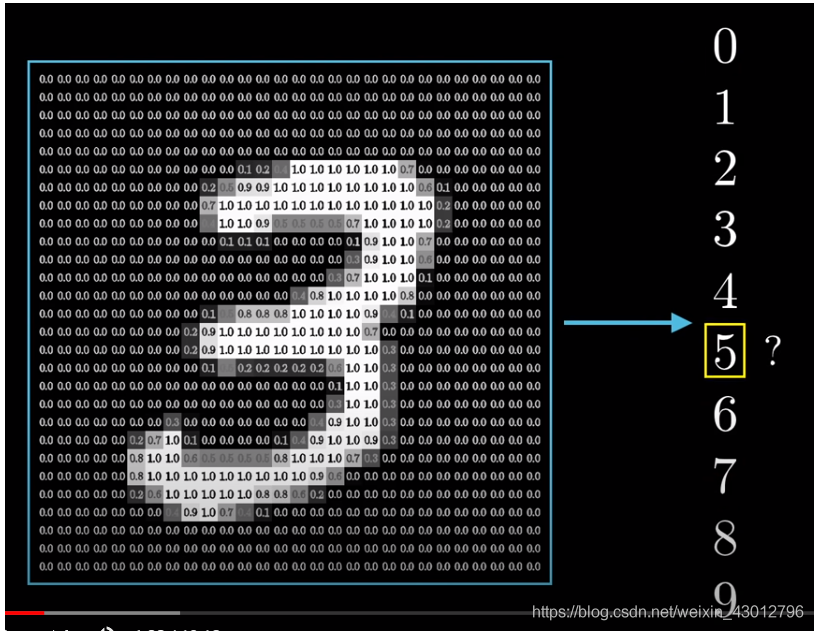
1. 如果分成28\*28的网格，具体如下：



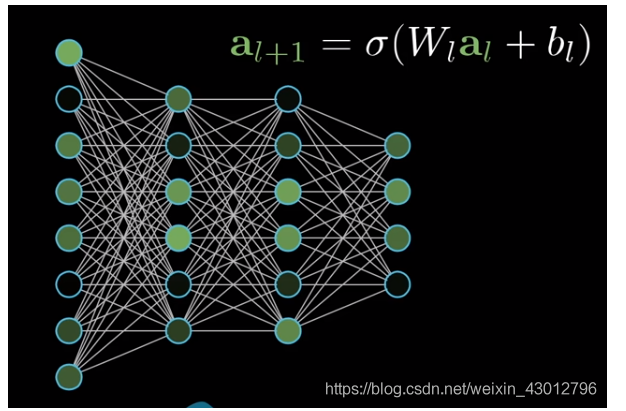
1. 28\*28 =784，可以由784个像素点代表3。大脑很容易区分3和6，但是计算机如何处理呢？



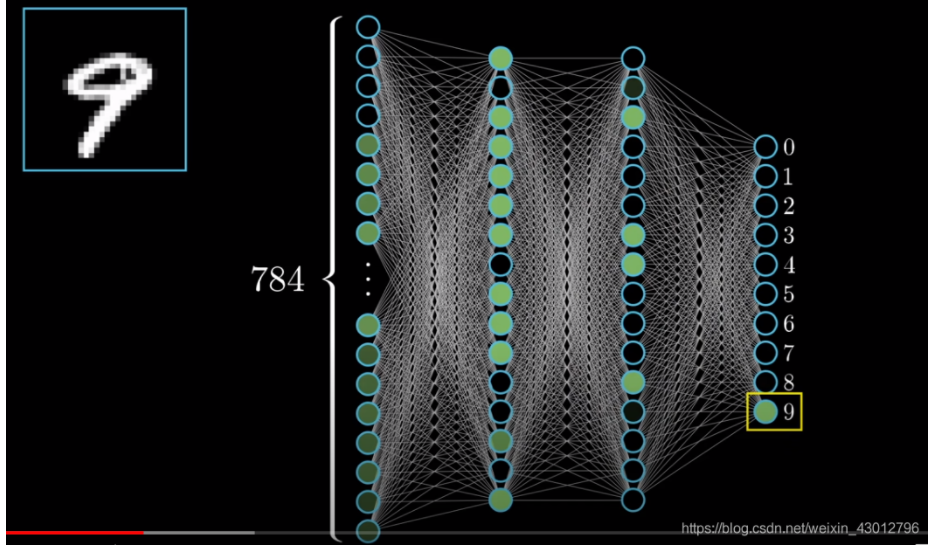
1. 如果是黑白图片，可以根据灰度值作出标记：黑色用0.0表示，白色用1.0表示



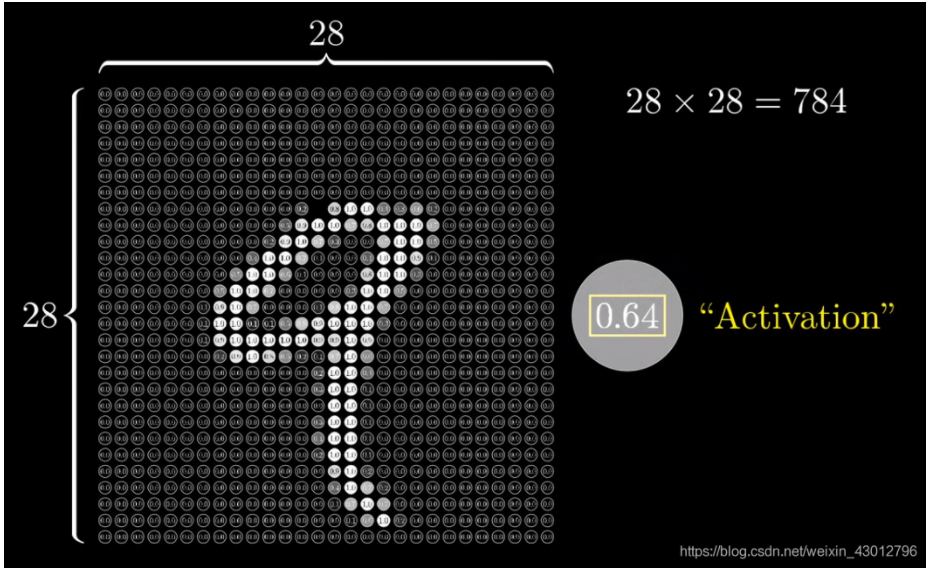
1. 神经网络如何识别：



1. 通过对每一个神经元赋值， 再乘以权重w, 再加上bias值， 再把计算结果传到激活函数中sigmoid, 用于激活下一层的神经元， 具体是如何实现的：

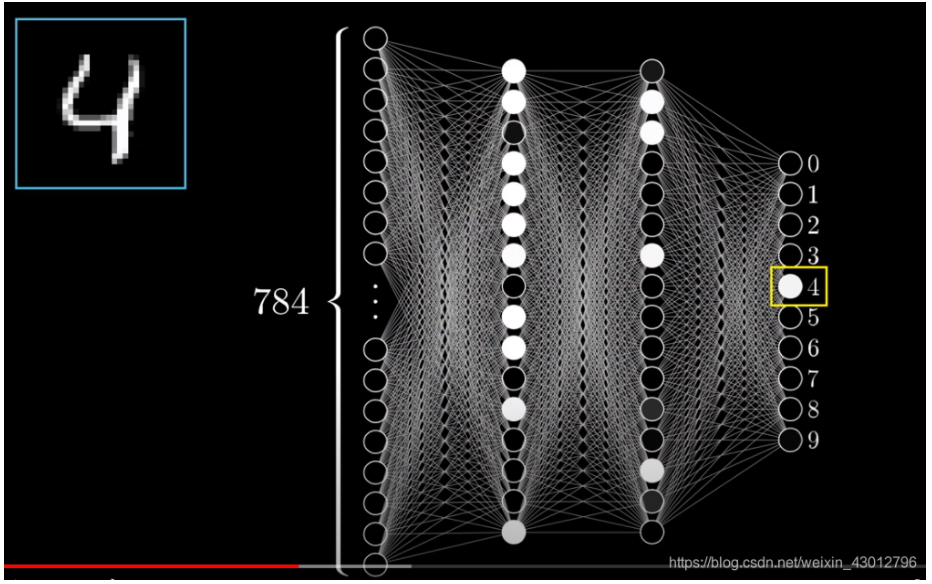


1. 神经元



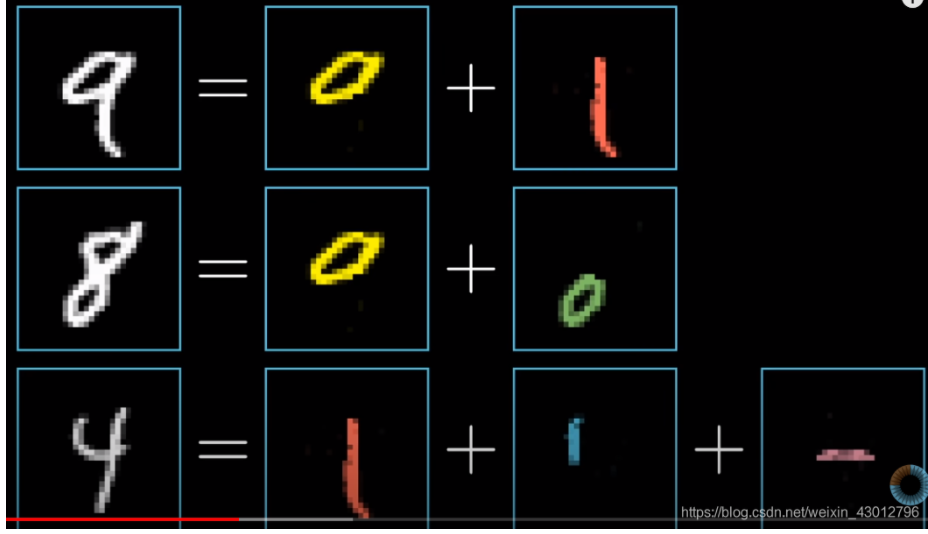
每一个像素代表一个神经元，一张28 \*28 的图片有784个神经元，神经元中的数字用灰度值表示，白色表示1，黑色表示0，至于最终的结果，我们暂时不需要关心。

1. 输入和输出



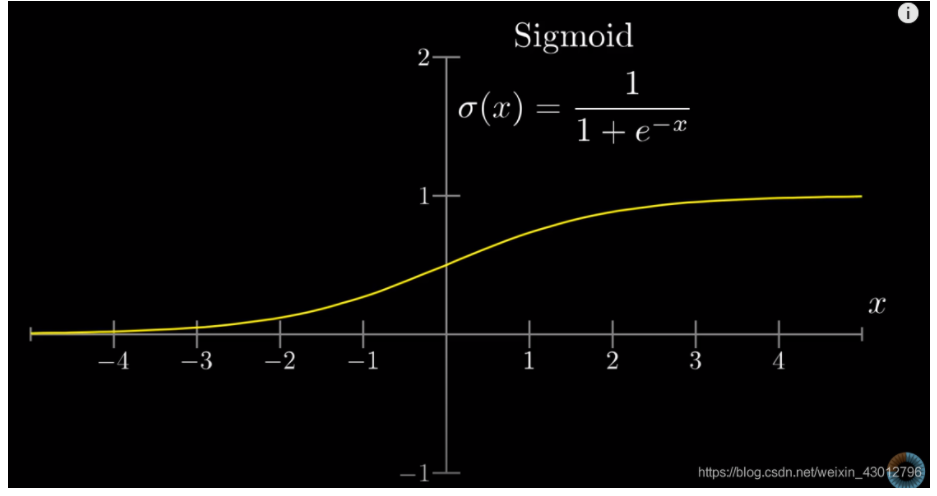
首先我们需要明确，输入和输出是确定的，给定图片大小，像素是确定的，输出是0-9，也是确定的，隐藏层(hidden layer)需要有多少个神经元需要自己根据情况设定，只要最终输出值=目标值即可。

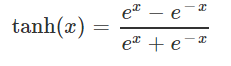
1. 假设，当我们识别数字时，其实也是根据一笔一划识别的：



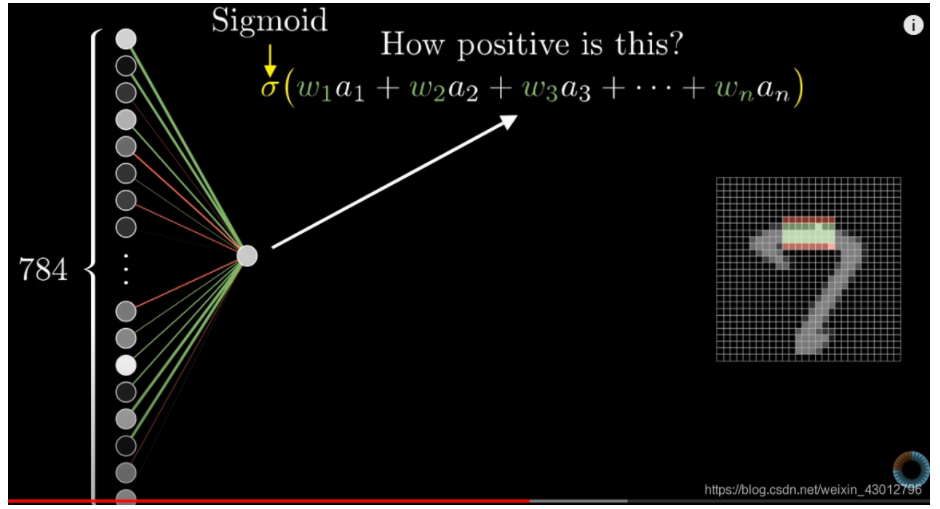
如果让每个神经元负责识别数字的一部分，最终达到识别所有数字，那么就需要分层

1. 激活函数



为了实现0,1的输出，需要选择激活函数，一个理想的激活函数可以是sigmoid函数，也可以是tanh(x)，或者其他函数，总之激活函数需要满足输出为0或1

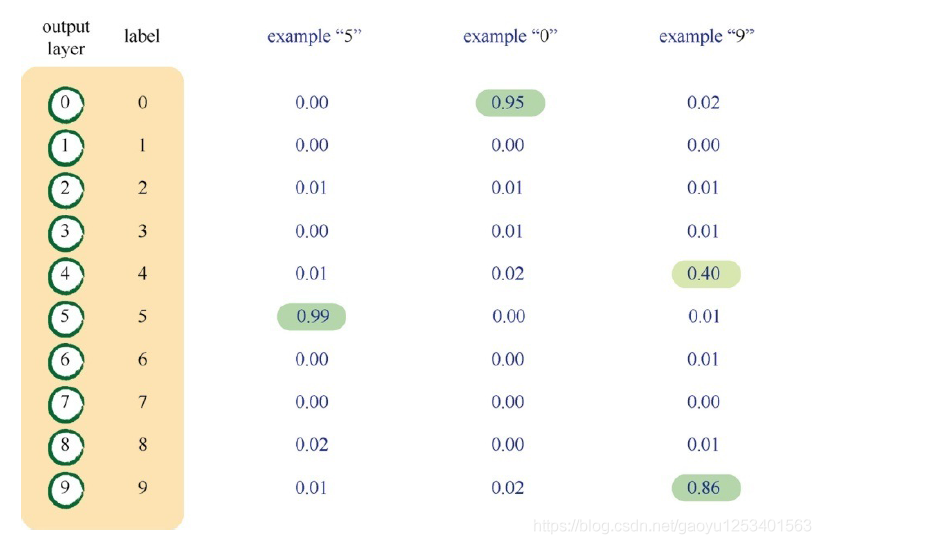
1. 激活函数作用：



通过input layer(输入层)的权证和灰度值相乘，再累加，传入到第二层的神经元中，通过激活函数计算输出结果，判断是否激活，如果激活，证明该数字中的某一部分存在，如果数字中的所有部分都能激活，那么就可识别这个数字，一般神经网络第一层识别数字的边角，深层识别具体的图案。

1. 如何识别：

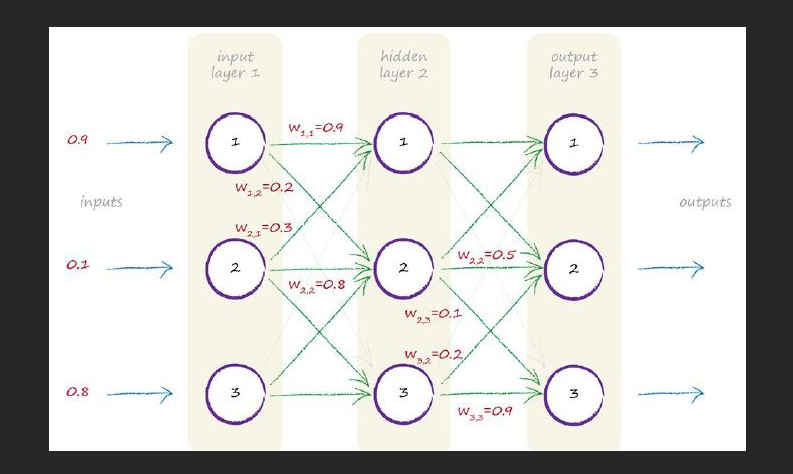
我们要求神经网络对图像进行分类，分配正确的标签，这些标签是0到9共10个数字中的一个，这意味着神经网络应该有10个输出层节点，每个节点对应一个可能的答案或标签，例如：如果答案是“0”，输出层第一个节点激发，而其余的输出节点则保持抑制状态。如下图所示：对于example“0”，可以看出输出层的最大信号来自于标签为“0”的节点，其余的输出节点产生非常小的信号输出。通常我们使用最大信号的输出节点对应的标签作为答案。



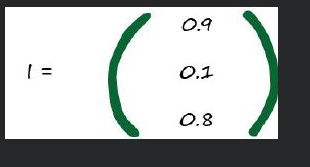
如果训练样本的标签为“0”，则需要创建输出节点的目标数组，其中除了对应于标签“0”的节点，其余所有的节点的值都应该很小，例如[0.99,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01]

1. 各层之间如何计算：

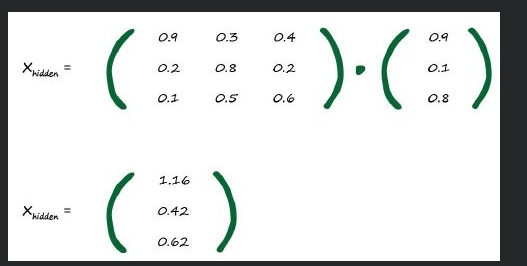
为了清晰地表示计算结果，未列出所有权重及节点



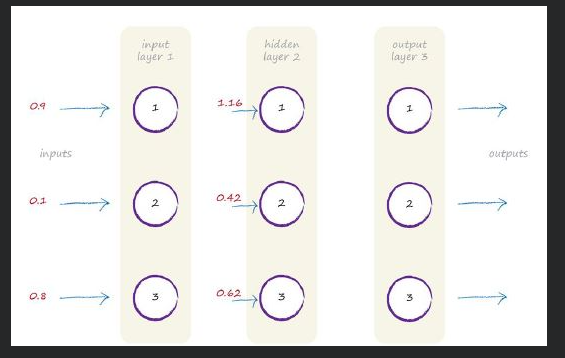
假设我们的输入值如图所示：



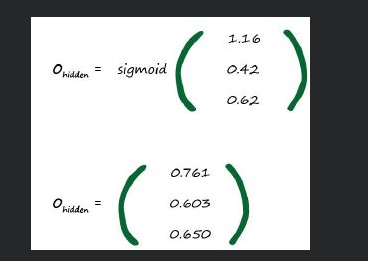
现在需要计算输入值经过权重矩阵计算后，给到隐藏层的值，计算结果如下：



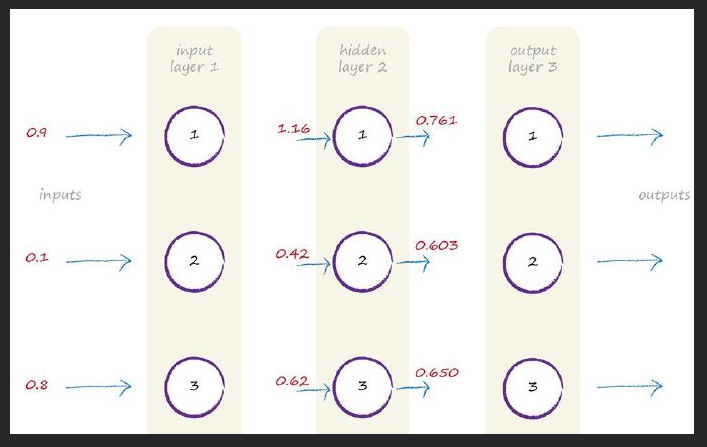
即：



隐藏层的输出值采用激活函数来计算：

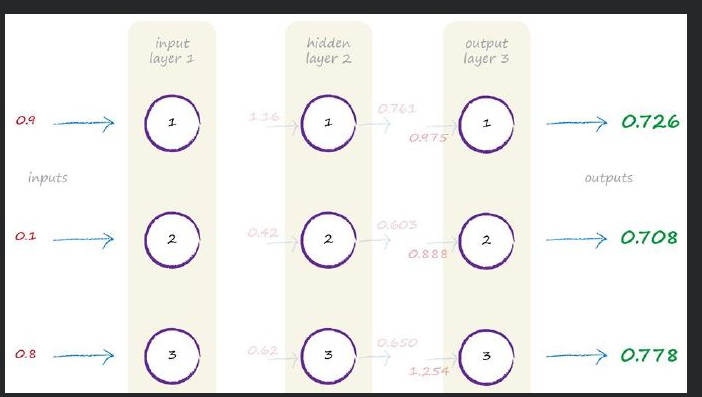


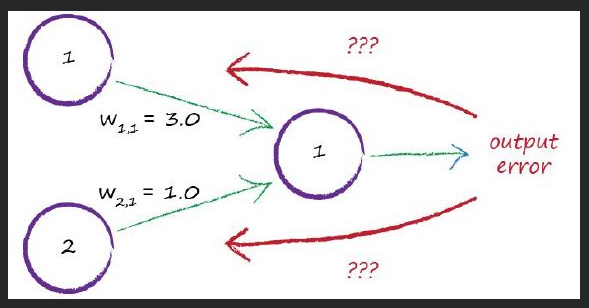
此时：



如果这是一个两层的神经网络，我们此时已经可以输出结果了。

重复输入层到隐藏层输出的计算过程：



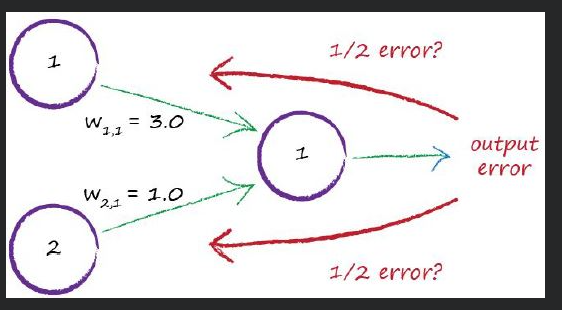
1. 从多个节点更新权重

如果输出节点的输出结果只受一个输入节点的影响，那计算很简单。但是，当输出节点的输出结果受到不止一个输入节点的影响时，我们如何利用输出结果的误差调整对应的权重呢？

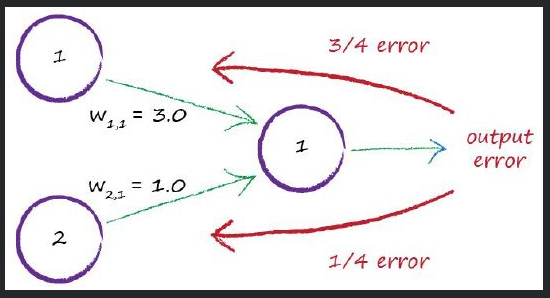
用所有误差值用于更新一个权重值显然是不合适的。因为此时忽视了其他节点对输出节点的影响。输出节点的误差是由多个输入节点影响的，而不止是受一个节点的影响。

输出结果的误差只受到一个节点的影响这种情况概率很小。

一种做法是把误差平分给受那些起作用的节点。如下图所示：



另一种想法是我们把误差给那些作用比较大的相连节点，而不是把误差平分给那些起作用的节点。为什么呢？因为这些节点对误差的影响更大。下图表示的这种思考方法：



梯度下降算法，反向传播