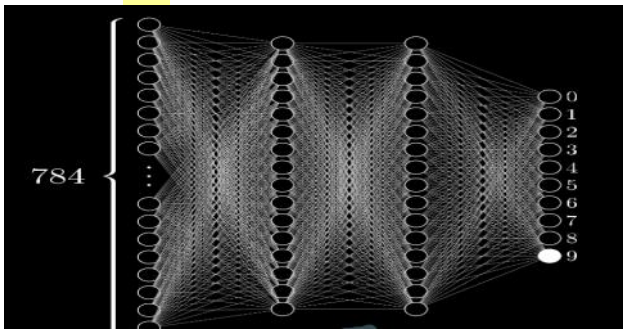
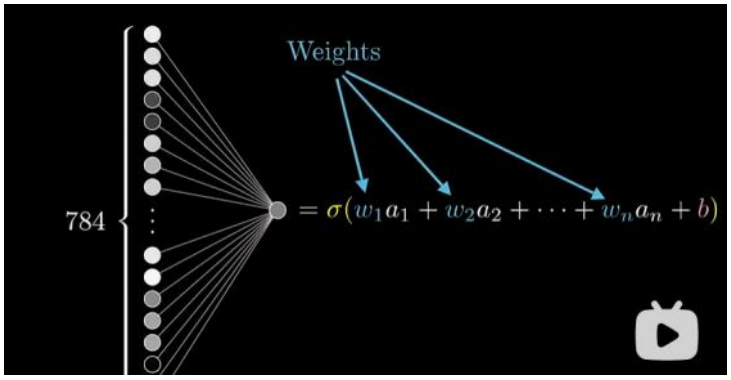
深度学习之梯度下降法

2022年5月20日　　21:51

以手写数字识别为例，把网络看作一个函数，有784个输入值，即每个像素的灰度值，被输入到网络第一层的784（28*28）个神经元里，输出值是10个数字（神经元的激活值），其中数值最高的代表识别到的数字，其中所有的权重和偏置值组成了这个函数的参数

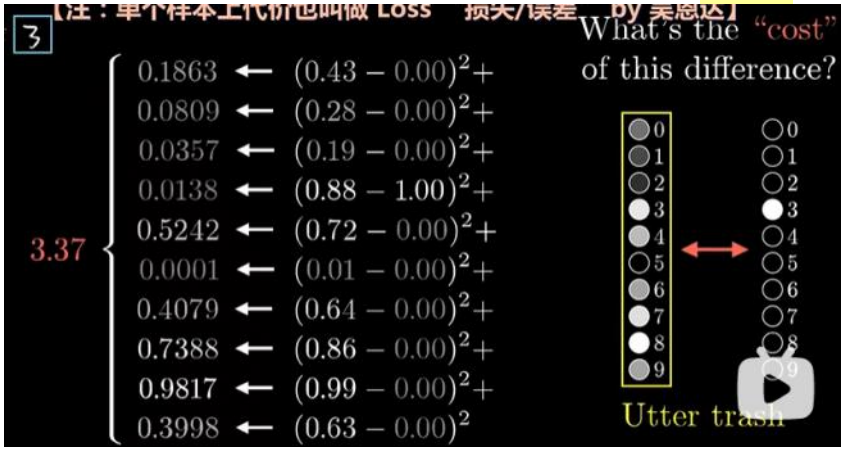


每一个神经元都与上一层所有的神经元相连接，决定其激活值的加权和中的权重，可以看作是那些连接的强弱，而偏置值决定了神经元是否更容易被激活，一开始随机初始化所有的权重和偏置值，让该网络识别用来训练的数据，误差会很大



问题1：如何反映（模型的好坏）测试的准确度呢?

我们可以把最后一层每个神经元的激活值与对应的期待值之间的差的平方加起来



把它称为训练单个样本的“误差”，显然，当网络对图像进行正确的识别时，这个“误差”就比较小，求出之后所有训练样本中“误差”的平均值，用这个值反映测试的准确度。

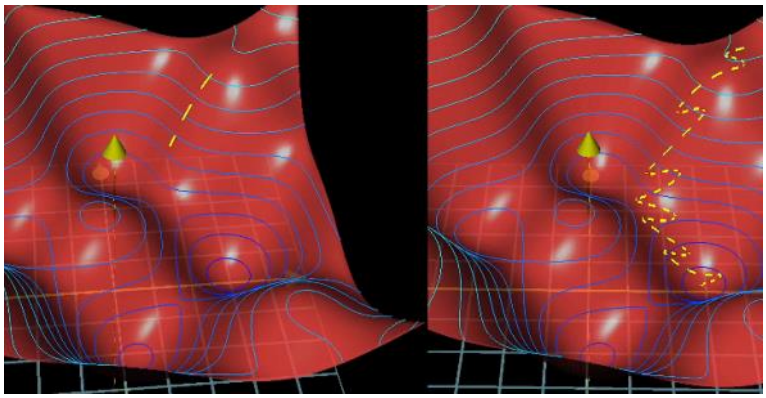
可以把“误差”看成一个函数，输入值是每层每个神经元对应的权重和偏置值，输出值是一个数字，即上面的“误差的平均值”，参数是成千上万个训练样本。

。

问题2：有没有什么提高计算效率的方法?

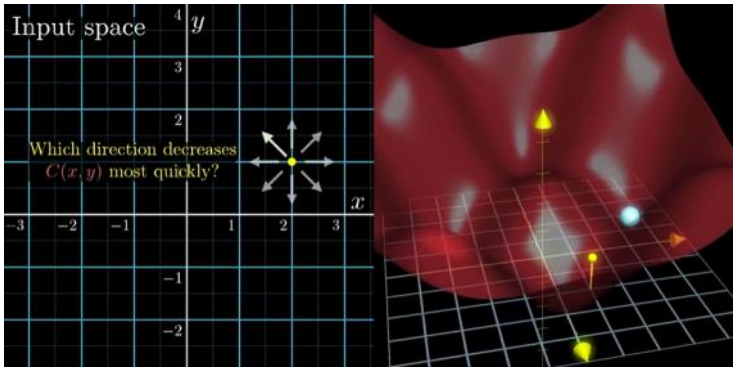
如果梯度下降的每一步都要用到所有的训练样本，花的时间太长了。可以首先把训练样本分成很多批，算出其中一批

样本下降的一步，之后的每一步都取决于随机抽取的一批样本，虽然这不能代表真实梯度，但也能在短时间内给出一个不错的近似，最终也能达到局部最低点。这个方法叫做随机梯度下降法。



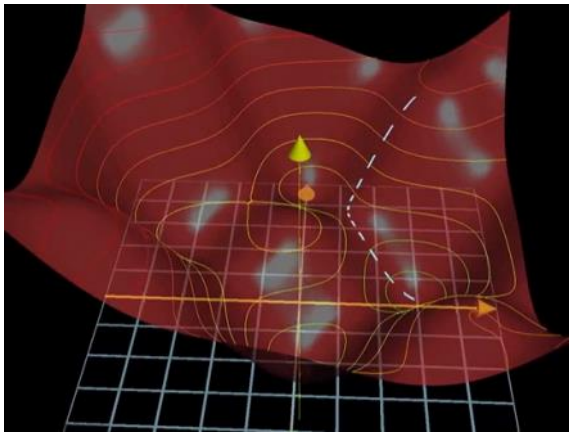
问题3：怎样通过改变权重和偏置值，使得“误差”达到最小?

以两个输入一个输出的二元函数为例，



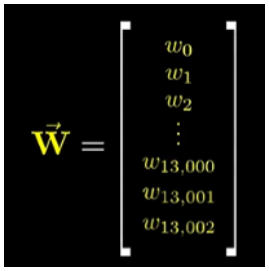
输入空间想象成xy平面，损失函数是平面上方的曲面，函数上任意取一个点，对应着随机初始化的权重和偏置值，这时可以求该点的梯度，显然梯度的负方向就是函数值下降最快的方向，设置合适的步长，即可使函数值逼近某个局部最小值。

为了取到最小的“损失”值，就要求“损失”函数必须是平滑的，这样才能每次挪一点点，达到局部最小值。

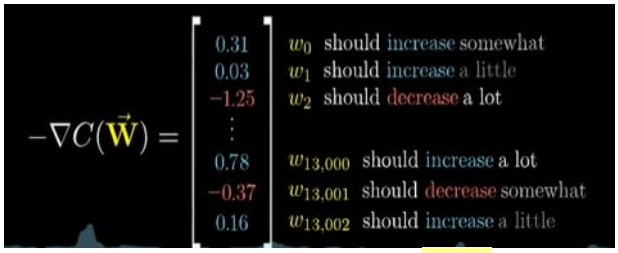


这顺便解释了为什么人工神经元的激活值是连续的

这种按照负梯度的倍数，不断调整函数输入值的过程，就叫做梯度下降法。



这是函数某个点上权重的一列向量



这是与权重对应的负梯度向量

负梯度中的每一项都告诉了我们两件事：正负号说明输入向量的这一项该调大还是调小，每一项绝对值的相对大小告诉了我们改变哪个值的影响更大，因此可以把负梯度向量理解为各个权重和偏置的相对重要度，表明改变哪个参数性价比更高，告诉我们如何微调权重和偏置的值才可以让“损失”函数的结果下降得最快