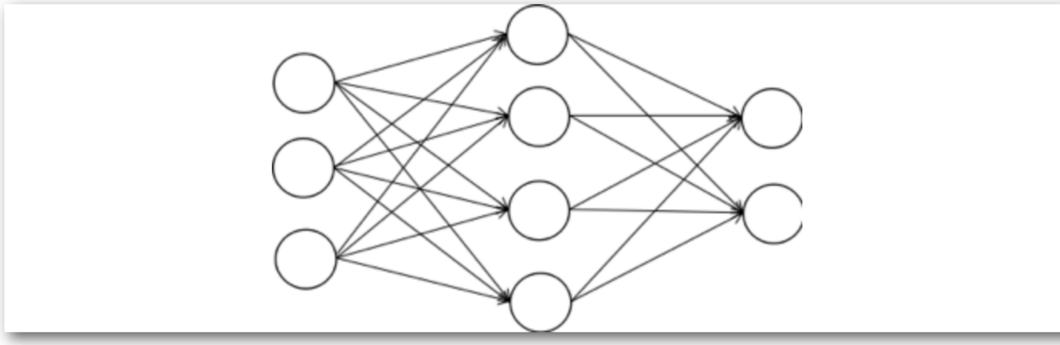
foward

下面我们来看一个简单神经网络的例子，如下图所示：



这个网络有3层。第一层是输入层，对应的输入向量为x，有3个神经元，写成分量形式为（x_1,x_2,x_3）它不对数据做任何处理，直接原样送入下一层。中间层有4个神经元，接受的输入数据为向量x，输出向量为y，写成分量形式为(y_1,y_2,y_3,y_4,y_5)。第三个层为输出层，接受的输入数据为向量y，输出向量为z，写成分量形式为(z_1,z_2)。第一层到第二层的权重矩阵为$W^{(1)}$，第二层到第三层的权重矩阵为$W^{(2)}$。权重矩阵的每一行为一个权重向量，是上一层所有神经元到本层某一个神经元的连接权重，这里的上标表示层数。

如果激活函数选用sigmoid函数，则第二层神经元的输出值为：

$$y_1 = \frac{1}{1+\exp\left(-\left(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_1^{(1)}\right)\right)}$$
$$y_2 = \frac{1}{1+\exp\left(-\left(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_2^{(1)}\right)\right)}$$
$$y_3 = \frac{1}{1+\exp\left(-\left(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)}\right)\right)}$$
$$y_4 = \frac{1}{1+\exp\left(-\left(w_{41}^{(1)}x_1 + w_{42}^{(1)}x_2 + w_{43}^{(1)}x_3 + b_4^{(1)}\right)\right)}$$

第三层神经元的输出值为：

$$z_1 = \frac{1}{1+\exp\left(-\left(w_{11}^{(2)}y_1 + w_{12}^{(2)}y_2 + w_{13}^{(2)}y_3 + w_{14}^{(2)}y_4 + b_1^{(2)}\right)\right)}$$
$$z_2 = \frac{1}{1+\exp\left(-\left(w_{21}^{(2)}y_1 + w_{22}^{(2)}y_2 + w_{23}^{(2)}y_3 + w_{24}^{(2)}y_4 + b_2^{(2)}\right)\right)}$$

如果把y_i代入上面二式中，可以将输出向量z表示成输出向量x的函数。通过调整权重矩阵和偏置项可以实现不同的函数映射，因此神经网络就是一个复合函数。

需要解决的一个核心问题是一旦神经网络的结构（即神经元层数，每层神经元数量）确定之后，怎样得到权重矩阵和偏置项。这些参数是通过训练得到的，这是本文推导的核心任务。

backward

首先以前面的3层神经网络为例，推导损失函数对神经网络所有参数梯度的计算方法。假设训练样本集中有m个样本(x_i, z_i)。其中x为输入向量，z为标签向量。现在要确定神经网络的映射函数：

$$\mathrm{z} = h(\mathrm{x})$$

什么样的函数能很好的解释这批训练样本？答案是神经网络的预测输出要尽可能的接近样本的标签值，即在训练集上最小化预测误差。如果使用均方误差，则优化的目标为：

$$L = \frac{1}{2m}\sum_{i=1}^{m}\left\| h(\mathrm{x}_i) - \mathrm{z}_i \right\|^2$$

其中h(x)和z_i都是向量，求和项内部是向量的2范数平方，即各个分量的平方和。上面的误差也称为欧氏距离损失函数，除此之外还可以使用其他损失函数，如交叉熵、对比损失等。

优化目标函数的自变量是各层的权重矩阵和梯度向量，一般情况下无法保证目标函数是凸函数，因此这不是一个凸优化问题，有陷入局部极小值和鞍点的风险（对于这些概念和问题，SIGAI之前的公众号文章“理解梯度下降法”，“理解凸优化”中已经做了详细介绍），这是神经网络之前一直被诟病的一个问题。可以使用梯度下降法进行求解，使用梯度下降法需要计算出损失函数对所有权重矩阵、偏置向量的梯度值，接下来的关键是这些梯度值的计算。在这里我们先将问题简化，只考虑对单个样本的损失函数：

$$L = \frac{1}{2}\left\| h(\mathrm{x}) - \mathrm{z} \right\|^2$$

后面如果不加说明，都使用这种单样本的损失函数。如果计算出了对单个样本损失函数的梯度值，对这些梯度值计算均值即可得到整个目标函数的梯度值。

$W^{(1)}$和$b^{(1)}$要被代入到网络的后一层中，是复合函数的内层变量，我们先考虑外层的$W^{(2)}$和$b^{(2)}$。权重矩阵$W^{(2)}$是一个2x4的矩阵，它的两个行分别为向量$\mathrm{w}_1^{(1)}$和$\mathrm{w}_2^{(1)}$，$b^{(2)}$是一个2维的列向量，它的两个元素为$b_1^{(2)}$和$b_2^{(2)}$。网络的输入是向量x，第一层映射之后的输出是向量y。

首先计算损失函数对权重矩阵每个元素的偏导数，将欧氏距离损失函数展开，有：

$$\frac{\partial L}{\partial w_{ij}^{(2)}} = \frac{\partial \frac{1}{2}\left(\left(f\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) - z_1\right)^2 + \left(f\left(\mathrm{w}_2^{(2)}\mathrm{y} + b_2^{(2)}\right) - z_2\right)^2\right)}{\partial w_{ij}^{(2)}}$$

如果i = 1，即对权重矩阵第一行的元素求导，上式分子中的后半部分对w_{ij}来说是常数。根据链式法则有：

$$\begin{aligned}
\frac{\partial L}{\partial w_{ij}^{(2)}} &= \left(f\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) - z_1\right) f'\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) \frac{\partial\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right)}{\partial w_{ij}^{(2)}} \\
&= \left(f\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) - z_1\right) f'\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) \frac{\partial\left(\sum_{k=1}^{4} w_{1k}^{(2)} y_k + b_1^{(2)}\right)}{\partial w_{ij}^{(2)}} \\
&= \left(f\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) - z_1\right) f'\left(\mathrm{w}_1^{(2)}\mathrm{y} + b_1^{(2)}\right) y_j
\end{aligned}$$

如果i = 2，即对矩阵第二行的元素求导，类似的有：

$$\frac{\partial L}{\partial w_{ij}^{(2)}} = \left(f\left(\mathrm{w}_2^{(2)}\mathrm{y} + b_2^{(2)}\right) - z_2\right) f'\left(\mathrm{w}_2^{(2)}\mathrm{y} + b_2^{(2)}\right) y_j$$

可以统一写成：

$$\frac{\partial L}{\partial w_{ij}^{(2)}} = \left(f\left(\mathrm{w}_i^{(2)}\mathrm{y} + b_i^{(2)}\right) - z_i\right) f'\left(\mathrm{w}_i^{(2)}\mathrm{y} + b_i^{(2)}\right) y_j$$

可以发现，第一个下标i决定了权重矩阵的第i行和偏置向量的第i个分量，第二个下标j决定了向量y的第j个分量。这可以看成是一个列向量与一个行向量相乘的结果，写成矩阵形式为：

$$\nabla_{\mathrm{W}^{(2)}} L = \left(f\left(\mathrm{W}^{(2)}\mathrm{y} + \mathrm{b}^{(2)}\right) - \mathrm{z} \right) \odot f'\left(\mathrm{W}^{(2)}\mathrm{y} + \mathrm{b}^{(2)}\right)\mathrm{y}^{\mathrm{T}}$$

上式中乘法 $\odot$ 为向量对应元素相乘，第二个乘法是矩阵乘法。$f\left(\mathrm{W}^{(2)}\mathrm{y}+\mathrm{b}^{(2)}\right)-\mathrm{z}$是一个2维列向量，$f'\left(\mathrm{W}^{(2)}\mathrm{y}+\mathrm{b}^{(2)}\right)$也是一个2维列向量，两个向量执行 $\odot$ 运算的结果还是一个2维列向量。y是一个4元素的列向量，其转置为4维行向量，前面这个二维列向量与y^{T}的乘积为2x4的矩阵，这正好与矩阵$\mathrm{W}^{(2)}$的尺寸相等。在上面的公式中，权重的偏导数在求和项中由3部分组成，分别是网络输出值与真实标签值的误差$f\left(\mathrm{W}^{(2)}\mathrm{y}+\mathrm{b}^{(2)}\right)-\mathrm{z}$，激活函数的导数$f'\left(\mathrm{W}^{(2)}\mathrm{y}+\mathrm{b}^{(2)}\right)$，本层的输入值y。神经网络的输出值、激活函数的导数值、本层的输入值都可以在正向传播时得到，因此可以高效的计算出来。对所有训练样本的偏导数计算均值，可以得到总的偏导数。

对偏置项的偏导数为：

$$\frac{\partial\left(\left(f\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)-z_1\right)^2+\left(f\left(\mathrm{w}_2^{(2)}\mathrm{y}+b_2^{(2)}\right)-z_2\right)^2\right)}{\partial b_i^{(2)}}$$

如果i = 1，上式分子中的后半部分对b_1来说是常数，有：

$$\begin{aligned}\frac{\partial L}{\partial b_1^{(2)}} &= \left(f\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)-z_1\right)f'\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)\frac{\partial\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)}{\partial b_1^{(2)}}\\ &= \left(f\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)-z_1\right)f'\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)\end{aligned}$$

如果i = 2，类似的有：

$$\frac{\partial L}{\partial b_2^{(2)}} = \left(f\left(\mathrm{w}_2^{(2)}\mathrm{y}+b_2^{(2)}\right)-z_2\right)f'\left(\mathrm{w}_2^{(2)}\mathrm{y}+b_2^{(2)}\right)$$

这可以统一写成：

$$\frac{\partial L}{\partial b_i^{(2)}} = \left(f\left(\mathrm{w}_i^{(2)}\mathrm{y}+b_i^{(2)}\right)-z_i\right)f'\left(\mathrm{w}_i^{(2)}\mathrm{y}+b_i^{(2)}\right)$$

写成矩阵形式为：

$$\nabla_{\mathrm{b}^{(2)}} L = \left(f\left(\mathrm{W}^{(2)}\mathrm{y} + \mathrm{b}^{(2)}\right) - \mathrm{z} \right) \odot f'\left(\mathrm{W}^{(2)}\mathrm{y} + \mathrm{b}^{(2)}\right)$$

偏置项的导数由两部分组成，分别是神经网络预测值与真实值之间的误差，激活函数的导数值，与权重矩阵的偏导数相比唯一的区别是少了y^{T}。接下来计算对$\mathrm{W}^{(1)}$和$\mathrm{b}^{(1)}$的偏导数，由于是复合函数的内层，情况更为复杂。$\mathrm{W}^{(1)}$是一个4x3的矩阵，它的4个行向量为$\mathrm{w}_1^{(1)}$，$\mathrm{w}_2^{(1)}$，$\mathrm{w}_3^{(1)}$，$\mathrm{w}_4^{(1)}$。偏置项$\mathrm{b}^{(1)}$是4维向量，4个分量分别是$b_1^{(1)}$，$b_2^{(1)}$，$b_3^{(1)}$，$b_4^{(1)}$。首先计算损失函数对$\mathrm{W}^{(1)}$的元素的偏导数：

$$\frac{\partial L}{\partial w_{ij}^{(1)}} = \frac{\partial \frac{1}{2}\left(\left(f\left(\mathrm{w}_1^{(2)}\mathrm{y}+b_1^{(2)}\right)-z_1\right)^2+\left(f\left(\mathrm{w}_2^{(2)}\mathrm{y}+b_2^{(2)}\right)-z_2\right)^2\right)}{\partial w_{ij}^{(1)}}$$

而：

$$\mathrm{y} = f\left(\mathrm{W}^{(1)}\mathrm{x}+\mathrm{b}^{(1)}\right)$$

上式分子中的两部分都有y，因此都与$\mathrm{W}^{(1)}$有关。为了表述简洁，我们令：

$$\mathrm{u}^{(2)} = \mathrm{W}^{(2)}\mathrm{y}+\mathrm{b}^{(2)}$$

根据链式法则有：

$$\frac{\partial L}{\partial w_{ij}^{(1)}} = \left(f\left(u_1^{(2)}\right)-z_1\right)f'\left(u_1^{(2)}\right)\frac{\partial \mathrm{w}_1^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}} + \left(f\left(u_2^{(2)}\right)-z_2\right)f'\left(u_2^{(2)}\right)\frac{\partial \mathrm{w}_2^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}$$

其中 $f\left(u_1^{(2)}\right)-z_1$ 和 $f'\left(u_1^{(2)}\right)$，$f\left(u_2^{(2)}\right)-z_2$ 和 $f'\left(u_2^{(2)}\right)$ 都是标量，$\mathrm{w}_1^{(2)}\mathrm{y}$ 和 $\mathrm{w}_2^{(2)}\mathrm{y}$ 是两个向量的内积，y 的每一个分量都是 $w_{ij}^{(1)}$ 的函数。接下来计算 $\frac{\partial \mathrm{w}_1^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}$ 和 $\frac{\partial \mathrm{w}_2^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}$：

$$\frac{\partial \mathrm{w}_1^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}} = \mathrm{w}_1^{(2)}\frac{\partial \mathrm{y}}{\partial w_{ij}^{(1)}}$$

这里的$\frac{\partial \mathrm{y}}{\partial w_{ij}^{(1)}}$是一个向量，表示y的每个分量分别对$w_{ij}^{(1)}$求导。当i = 1时有：

$$\frac{\partial \mathrm{y}}{\partial w_{ij}^{(1)}}=\begin{bmatrix}\frac{\partial y_1}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_2}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_3}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_4}{\partial w_{ij}^{(1)}}\end{bmatrix}=\begin{bmatrix}f'\left(\mathrm{w}_1^{(1)}\mathrm{x}+b_1^{(1)}\right)x_j\\ 0\\ 0\\ 0\end{bmatrix}$$

后面3个分量相对于求导变量$w_{ij}^{(1)}$都是常数。类似的当i = 2时有：

$$\frac{\partial \mathrm{y}}{\partial w_{ij}^{(1)}}=\begin{bmatrix}\frac{\partial y_1}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_2}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_3}{\partial w_{ij}^{(1)}}\\ \frac{\partial y_4}{\partial w_{ij}^{(1)}}\end{bmatrix}=\begin{bmatrix}0\\ f'\left(\mathrm{w}_2^{(1)}\mathrm{x}+b_1^{(1)}\right)x_j\\ 0\\ 0\end{bmatrix}$$

i = 3和i = 4时的结果以此类推。综合起来有：

$$\frac{\partial \mathrm{w}_1^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}=w_{1i}^{(2)}f'\left(\mathrm{w}_i^{(1)}\mathrm{x}+b_i^{(1)}\right)x_j$$

同理有：

$$\frac{\partial \mathrm{w}_2^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}=w_{2i}^{(2)}f'\left(\mathrm{w}_i^{(1)}\mathrm{x}+b_i^{(1)}\right)x_j$$

如果令：

$$\mathrm{u}^{(1)}=\mathrm{W}^{(1)}\mathrm{x}+\mathrm{b}^{(1)}$$

合并得到：

$$\begin{aligned}\frac{\partial L}{\partial w_{ij}^{(1)}}&=\left(f\left(u_1^{(2)}\right)-z_1\right)f'\left(u_1^{(2)}\right)\frac{\partial \mathrm{w}_1^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}+\left(f\left(u_2^{(2)}\right)-z_2\right)f'\left(u_2^{(2)}\right)\frac{\partial \mathrm{w}_2^{(2)}\mathrm{y}}{\partial w_{ij}^{(1)}}\\&=\left(f\left(u_1^{(2)}\right)-z_1\right)f'\left(u_1^{(2)}\right)w_{1i}^{(2)}f'\left(u_1^{(1)}\right)x_j+\left(f\left(u_2^{(2)}\right)-z_2\right)f'\left(u_2^{(2)}\right)w_{2i}^{(2)}f'\left(u_2^{(1)}\right)x_j\\&=\begin{bmatrix}w_{1i}^{(2)} & w_{2i}^{(2)}\end{bmatrix}\left(\left(f\left(\mathrm{u}^{(2)}\right)-\mathrm{z}\right)\odot f'\left(\mathrm{u}^{(2)}\right)\odot f'\left(\mathrm{u}^{(1)}\right)\right)x_j\end{aligned}$$

写成矩阵形式为：

$$\nabla_{\mathrm{W}^{(1)}}L=\left(\mathrm{W}^{(2)}\right)^{\mathrm{T}}\left(\left(f\left(\mathrm{u}^{(2)}\right)-\mathrm{z}\right)\odot f'\left(\mathrm{u}^{(2)}\right)\odot f'\left(\mathrm{u}^{(1)}\right)\right)\mathrm{x}^{\mathrm{T}}$$

至此，我得到了这个简单网络对所有参数的偏导数，接下来我们将这种做法推广到更一般的情况。从上面的结果可以看出一个规律，输出层的权重矩阵和偏置向量梯度计算公式中共用了$\left(f\left(\mathrm{u}^{(2)}\right)-\mathrm{z}\right)\odot f'\left(\mathrm{u}^{(2)}\right)$。对于隐含层也有类似的结果。

完整算法

现在考虑一般的情况。假设有m个训练样本(x_i, y_i)，其中x_i为输入向量，y_i为标签向量。训练的目标是最小化样本标签值与神经网络预测值之间的误差，如果使用均方误差，则优化的目标为：

$$L(W)=\frac{1}{2m}\sum_{i=1}^{m}\left\|h(\mathrm{x}_i)-\mathrm{y}_i\right\|^2$$

其中W为神经网络所有参数的集合，包括各层的权重和偏置。这个最优化问题是一个不带约束条件的问题，可以用梯度下降法求解。

上面的误差函数定义在整个训练样本集上，梯度下降法每一次迭代利用了所有训练样本，称为批量梯度下降法。如果样本数量很大，每次迭代都用所有样本进计算成本太高。为了解决这个问题，可以采用单样本梯度下降法，我们将上面的损失函数写成对单个样本的损失函数之和：

$$L(W)=\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h(\mathrm{x}_i)-\mathrm{y}_i\right\|^2\right)$$

定义对单个样本(x_i, y_i)的损失函数为：

$$L_i=L(W,\mathrm{x}_i,\mathrm{y}_i)=\frac{1}{2}\left\|h(\mathrm{x}_i)-\mathrm{y}_i\right\|^2$$

如果采用单个样本进行迭代，梯度下降法第t + 1次迭代时参数的更新公式为：

$$W_{t+1}=W_t-\eta\nabla_W L_i(W_t)$$

如果要用所有样本进行迭代，根据单个样本的损失函数梯度计算总损失梯度即可，即所有样本梯度的均值。

用梯度下降法求解需要初始化优化变量的值。一般初始化为一个随机数，如用正态分布$N(0,\sigma^2)$产生这些随机数，其中σ是一个很小的正数。

到目前为止还有一个关键问题没有解决：目标函数是一个多层的复合函数，因为神经网络中每一层都有权重矩阵和偏置向量，且每一层的输出将会作为下一层的输入。因此，直接计算损失函数对所有权重和偏置的梯度很复杂，需要使用复合函数的求导公式进行递推计算。