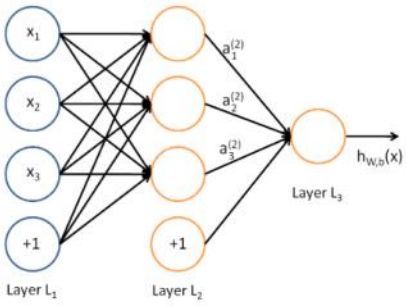
神经网络

2023年4月12日　　下午 01:55

神经网络是通过将众多简单的神经元连接在一起得到的，一个神经元的输出可作为另一神经元的输入。例如，这里有一个小神经网络：



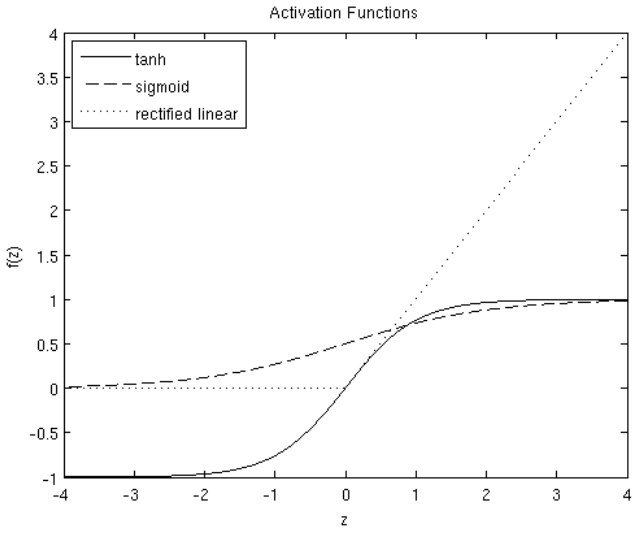
图中，用圆圈表示网络输入。圈里被标为 “+1” 的圆圈称为**偏置单元**，对应于截距项。网络最左边的那一层称为**输入层**，而**输出层**即最右层（在这个例子中，输出层只有一个节点）。介于最左（输入层）和最右（输出层）的中间层称为**隐藏层**，称为“隐藏”是因为其值在训练集中无法观察到。该例中的神经网络有 3 个输入单元（不计偏置单元计算在内），3 个隐藏单元，和 1 个输出单元。

激活函数（Activation Function）是神经网络中的一种非线性变换，用于将神经元的输入映射到输出。它的作用是引入非线性因素，增加网络的表达能力，从而使神经网络可以学习更复杂的函数关系。**激活值**指的是神经元或神经网络中每个节点的输出值，它是由该节点的输入值经过激活函数进行变换得到的。在神经网络中，激活值会作为下一层神经元的输入值，从而进行信息的传递和处理。

常见的激活函数包括：

1. Sigmoid函数（S型函数）：将输入映射到[0, 1]之间，常用于二分类问题。
2. Tanh函数：将输入映射到[-1, 1]之间，常用于多分类问题。
3. ReLU函数（Rectified Linear Unit）：将负数输入置为0，将正数输入保持不变，常用于深度神经网络中。
4. Leaky ReLU函数：与ReLU类似，但负数输入的输出不为0而是一个很小的负数。
5. Softmax函数：用于多分类问题，将输入映射到[0, 1]之间，且所有输出的和为1。
6. 双曲正切函数（Hyperbolic Tangent Function）：在深度神经网络中，双曲正切函数通常优于Sigmoid函数。它是S型函数的一个缩放版本。

不同的激活适用于不同的神经网络架构和任务。例如，ReLU函数在深度神经网络中通常表现良好，因为它可以有效地解决梯度消失问题；而在输出层使用Softmax函数通常可以提高分类的准确性。



前向传播算法（Forward propagation）用于计算输入数据经过神经网络后得到的输出。具体来说，前向传播算法是将输入数据按照网络的结构从输入层开始逐层向前传递，并通过激活函数计算每个神经元的输出值，最终得到输出层的输出结果。就是输入神经元乘以权重加上偏置作为下一层的输入，经过activation function激活之后又作为输出传递下去。

反向传播算法（Backpropagation）是一种用于训练神经网络的常见算法。它可以通过调整网络中的权重和偏置，使网络的输出更接近于期望输出。权重决定了一个神经元对上一层神经元输出的影响程度。在反向传播算法中，我们通过调整这些权重来最小化神经网络的输出与期望输出之间的差距。反向传播算法基于梯度下降优化算法，通过计算损失函数关于网络参数的梯度，来更新参数值以最小化损失函数。反向传播算法通过计算损失函数对权重的梯度来更新权重。梯度是一个指示损失函数随着权重变化的速率和方向的向量。通过计算梯度并使用梯度下降等优化算法，我们可以逐步调整权重，以使神经网络的输出逼近期望输出。

梯度下降算法的基本思想是沿着损失函数的梯度方向逐步迭代优化模型参数。具体地，从一个随机初始化的点开始，计算损失函数在该点处的梯度，然后朝着梯度的反方向更新参数，直到达到一个局部最小值或者算法达到收敛条件。

具体来说，在每次迭代中，我们计算损失函数对于每个参数的偏导数（即梯度），然后使用学习率乘以梯度的负值，将其作为参数的更新量。学习率是一个重要的超参数，控制每次迭代的步长大小。

梯度下降算法有多种变体，如批量梯度下降、随机梯度下降和小批量梯度下降等。批量梯度下降在每次迭代中使用所有样本来计算梯度，因此计算量较大，但可以获得更准确的梯度。随机梯度下降在每次迭代中只使用一个样本来计算梯度，计算量较小但更新方向较不稳定。小批量梯度下降是两者的折中，使用一小批样本来计算梯度。

梯度下降法中每一次迭代都按照如下公式对参数 W 和 b 进行更新：

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)$$