神经网络

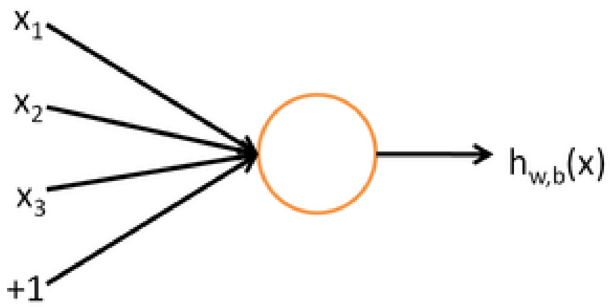
From Ufldl

Contents

概述

以监督学习为例，假设我们有训练样本集 $(x(^i), y(^i))$ ，那么神经网络算法能够提供一种复杂且非线性的假设模型 $h_{W,b}(x)$ ，它具有参数 W, b ，可以以此参数来拟合我们的数据。

为了描述神经网络，我们先从最简单的神经网络讲起，这个神经网络仅由一个“神经元”构成，以下即是这个“神经元”的图示：



这个“神经元”是一个以 x_1, x_2, x_3 及截距 $+1$ 为输入值的运算单元，其输出为 $h_{W,b}(x) = f(W^Tx) = f(\sum_{i=1}^3 W_ix_i + b)$ ，其中函数 $f : \Re \mapsto \Re$ 被称为“激活函数”。在本教程中，我们选用sigmoid函数作为激活函数 $f(\cdot)$

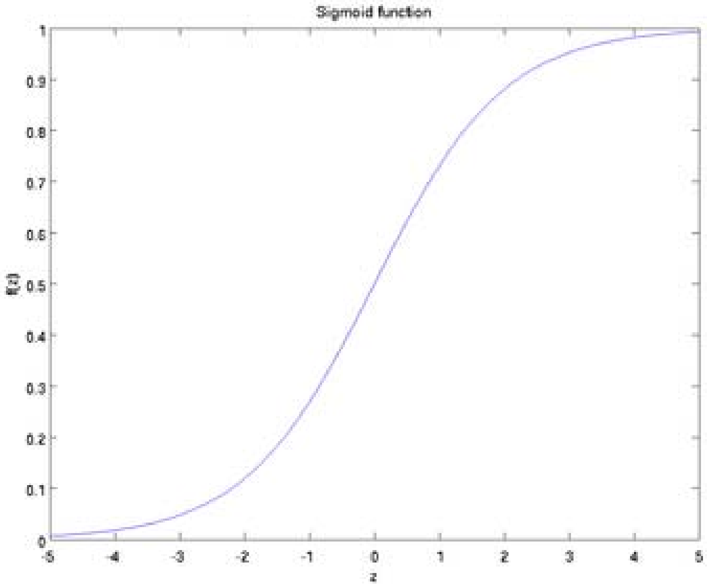
$$f(z) = \frac{1}{1 + \exp(-z)}.$$

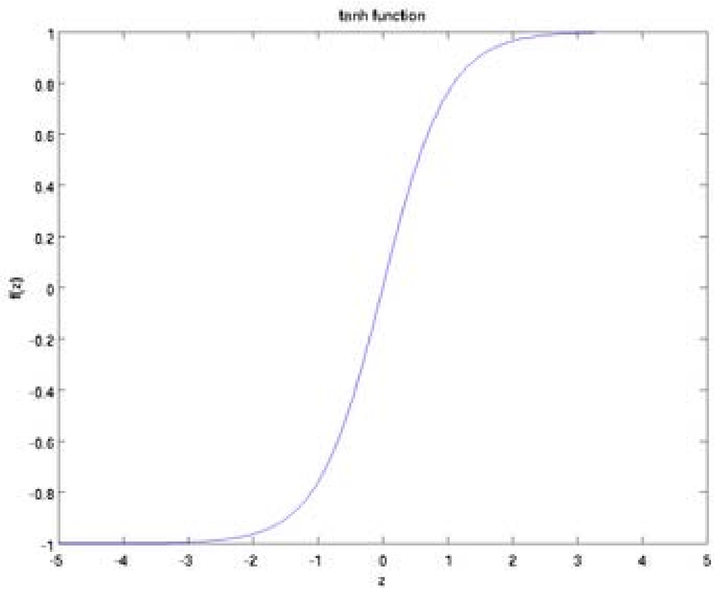
可以看出，这个单一“神经元”的输入－输出映射关系其实就是一个逻辑回归（logistic regression）。

虽然本系列教程采用sigmoid函数，但你也可以选择双曲正切函数（tanh）：

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

以下分别是sigmoid及tanh的函数图像





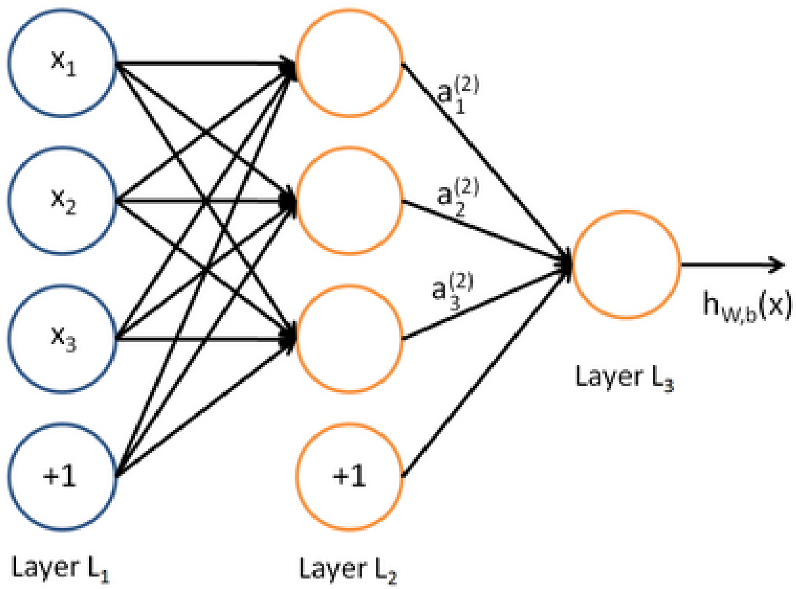
$\tanh(z)$ 函数是sigmoid函数的一种变体，它的取值范围为 $[-1,1]$ ，而不是sigmoid函数的 $[0,1]$ 。

注意，与其它地方（包括OpenClassroom公开课以及斯坦福大学CS229课程）不同的是，这里我们不再令 $x_0 = 1$ 。取而代之，我们用单独的参数 b 来表示截距。

最后要说明的是，有一个等式我们以后会经常用到：如果选择 $f(z) = 1/(1+\exp(-z))$ ，也就是sigmoid函数，那么它的导数就是 $f'(z) = f(z)(1-f(z))$ （如果选择tanh函数，那它的导数就是 $f'(z) = 1-(f(z))^2$ ，你可以根据sigmoid（或tanh）函数的定义自行推导这个等式。

神经网络模型

所谓神经网络就是将许多个单一“神经元”联结在一起，这样，一个“神经元”的输出就可以是另一个“神经元”的输入。例如，下图就是一个简单的神经网络：



我们使用圆圈来表示神经网络的输入，标上“$+1$”的圆圈被称为偏置节点，也就是截距项。神经网络最左边的一层叫做输入层，最右的一层叫做输出层（本例中，输出层只有一个节点）。中间所有节点组成的一层叫做隐藏层，因为我们不能在训练样本集中观测到它们的值。同时可以看到，以上神经网络的例子中有3个输入单元（偏置单元不计在内），3个隐藏单元及一个输出单元。

我们用 n_l 来表示网络的层数，本例中 $n_l = 3$ ，我们将第 l 层记为 L_l ，于是 L_1 是输入层，输出层是 L_{n_l} 。本例神经网络有参数 $(W,b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ ，其中 $W_{ij}^{(l)}$ （下面的式子中用到）是第 l 层第 j 单元与第 $l+1$ 层第 i 单元之间的联接参数（其实就是连接线上的权重，注意标号顺序）， $b_i^{(l)}$ 是第 $l+1$ 层第 i 单元的偏置项。因此在本例中， $W^{(1)} \in \Re^{3\times 3}$ ， $W^{(2)} \in \Re^{1\times 3}$ 。注意，没有其他单元连向偏置单元(即偏置单元没有输入)，因为它们总是输出 $+1$。同时，我们用 s_l 表示第 l 层的节点数（偏置单元不计在内）。

我们用 $a_i^{(l)}$ 表示第 l 层第 i 单元的激活值（输出值）。当 $l=1$ 时， $a_i^{(1)} = x_i$ ，也就是第 i 个输入值（输入值的第 i 个特征）。对于给定参数集合 W,b ，我们的神经网络就可以按照函数 $h_{W,b}(x)$ 来计算输出结果。本例神经网络的计算步骤如下：

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) = a_1^{(3)} &= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

我们用 $z_i^{(l)}$ 表示第 l 层第 i 单元输入加权和（包括偏置单元），比如，$z_i^{(2)} = \sum_{j=1}^n W_{ij}^{(1)} x_j + b_i^{(1)}$ ，则 $a_i^{(l)} = f(z_i^{(l)})$ 。

这样我们就可以得到一种更简洁的表示法。这里我们将激活函数 $f(\cdot)$ 扩展为用向量（分量的形式）来表示，即 $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$ ，那么，上面的等式可以更简洁地表示为：

$$
\begin{aligned}
z^{(2)} &= W^{(1)} x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)} a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

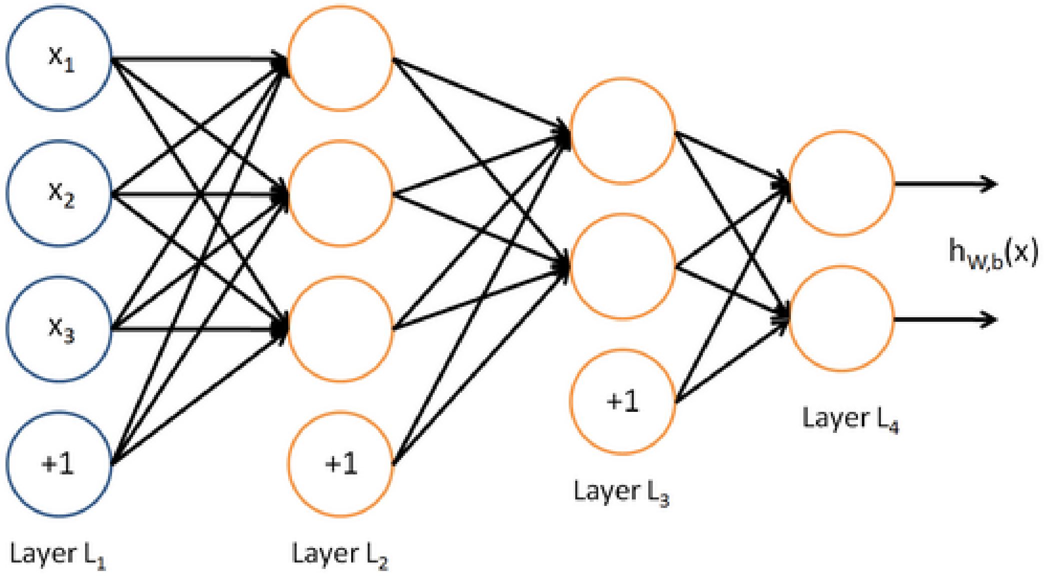
我们将上面的计算步骤叫作前向传播。回想一下，之前我们用 $a^{(1)} = x$ 表示输入层的激活值，那么给定第 l 层的激活值 $a^{(l)}$ 后，第 $l+1$ 层的激活值 $a^{(l+1)}$ 就可以按照下面步骤计算得到：

$$
\begin{aligned}
z^{(l+1)} &= W^{(l)} a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

将参数矩阵化，使用矩阵－向量运算方式，我们就可以利用线性代数的优势对神经网络进行快速求解。

目前为止，我们讨论了一种神经网络，我们也可以构建另一种结构的神经网络（这里结构指的是神经元之间的联接模式），也就是包含多个隐藏层的神经网络。最常见的一个例子是 n_l 层的神经网络，第 1 层是输入层，第 n_l 层是输出层，中间的每个层 l 与层 $l+1$ 紧密相联。这种模式下，要计算神经网络的输出结果，我们可以按照之前描述的等式，按部就班，进行前向传播，逐一计算第 L_2 层的所有激活值，然后是第 L_3 层的激活值，以此类推，直到第 L_{n_l} 层。这是一个前馈神经网络的例子，因为这种联接图没有闭环或回路。

神经网络也可以有多个输出单元。比如，下面的神经网络有两层隐藏层： L_2 及 L_3 ，输出层 L_4 有两个输出单元。



要求解这样的神经网络，需要样本集 $(x^{(i)}, y^{(i)})$ ，其中 $y^{(i)} \in \Re^2$ 。如果你想预测的输出是多个的，那这种神经网络很适用。（比如，在医疗诊断应用中，患者的体征指标就可以作为向量的输入值，而不同的输出值 y_i 可以表示不同的疾病存在与否。）

中英文对照

neural networks 神经网络

activation function 激活函数

hyperbolic tangent 双曲正切函数

bias units 偏置项

activation 激活值

forward propagation 前向传播

feedforward neural network 前馈神经网络(参照Mitchell的《机器学习》的翻译)

中文译者

孙逊（sunpaofu@foxmail.com），林锋（xlfg@yeah.net），刘鸿鹏飞（just.dark@foxmail.com），许利杰（csxulijie@gmail.com）

神经网络 | 反向传导算法 | 梯度检验与高级优化 | 自编码算法与稀疏性 | 可视化自编码器训练结果 | 稀疏自编码器符号一览表 | Exercise:Sparse_Autoencoder

Language : English