多层感知机

一、 简介

因为在一些任务中使用线性模型可能会存在较大的误差，所以可以通过增加隐蔽层的方式来来更好表示神经网络，这样就可以采用非线性的模型进行预测。网络可以表示为：

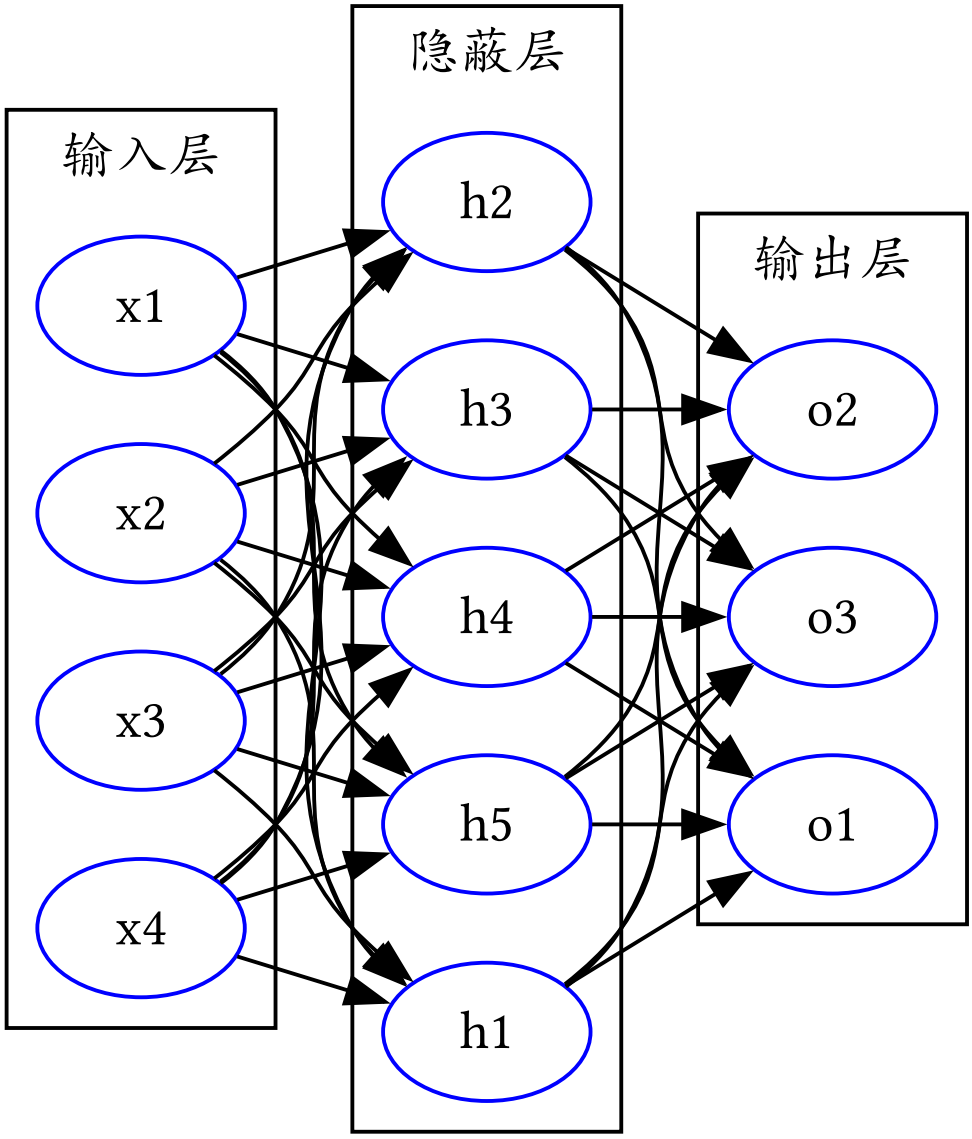

Figure 1 - 1: 网络图

再上图中的神经网络中，输入为 4，输出为 3，隐藏层的神经元个数为 5，总的层数为 2。 **全连接层，是每一个结点都与上一层的所有结点相连** 。隐藏层和输出层都是全连接层。

二、 隐蔽层

设隐藏层的输出为X_h，隐藏层的权重和偏差为W_h和b_h，输出为O_h。输出层的输入就是$X_o = O_h$也就是隐藏层的输出，输出层的权重和偏差为W_I和W_I，输出是O_I，这样就可以将这个网络表示为：

$$\begin{aligned} O_h &= W_h \cdot X_h + b_h \\ \hat{y} = O_I &= W_I \cdot O_h + b_I \end{aligned} \tag{2 - 1}$$

隐藏层的输出写成矩阵形式为:

$$\begin{pmatrix} o_{\mathrm{h1}} \\ o_{\mathrm{h2}} \\ o_{\mathrm{h3}} \\ o_{\mathrm{h4}} \\ o_{\mathrm{h5}} \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \\ h_{51} & h_{52} & h_{53} & h_{54} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_{\mathrm{h1}} \\ b_{\mathrm{h2}} \\ b_{\mathrm{h3}} \\ b_{\mathrm{h4}} \\ b_{\mathrm{h5}} \end{pmatrix} \tag{2 - 2}$$

输出层的输出写成矩阵形式为：

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} o_{11} & o_{12} & o_{13} \\ o_{21} & o_{22} & o_{23} \\ o_{31} & o_{32} & o_{33} \end{pmatrix} \times \begin{pmatrix} o_{\mathrm{h1}} \\ o_{\mathrm{h2}} \\ o_{\mathrm{h3}} \\ o_{\mathrm{h4}} \\ o_{\mathrm{h5}} \end{pmatrix} + \begin{pmatrix} b_{\mathrm{o1}} \\ b_{\mathrm{o2}} \\ b_{\mathrm{o3}} \end{pmatrix} \tag{2 - 3}$$

写到一起就是:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} o_{11} & o_{12} & o_{13} \\ o_{21} & o_{22} & o_{23} \\ o_{31} & o_{32} & o_{33} \end{pmatrix} \times \left(\begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \\ h_{51} & h_{52} & h_{53} & h_{54} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_{\mathrm{h1}} \\ b_{\mathrm{h2}} \\ b_{\mathrm{h3}} \\ b_{\mathrm{h4}} \\ b_{\mathrm{h5}} \end{pmatrix} \right) + \begin{pmatrix} b_{\mathrm{o1}} \\ b_{\mathrm{o2}} \\ b_{\mathrm{o3}} \end{pmatrix} \tag{2 - 4}$$

简写为：$\hat{y} = O_I * (H * X + b_h) + b_I = XW_hW_o + b_hW_o + b_o$

三、 激活函数

在仿射变换之后对每个隐藏单元应用非线性的激活函数（activation function）σ。激活函数的输出（例如，$\sigma(.)$）被称为活性值（activations）。一般来说，有了激活函数，就不可能再将我们的多层感知机退化成线性模型。

3.1 ReLU 函数

ReLU 函数可以写为：

$$\text{ReLU}(x) = \max(x, 0) \tag{3 - 1}$$

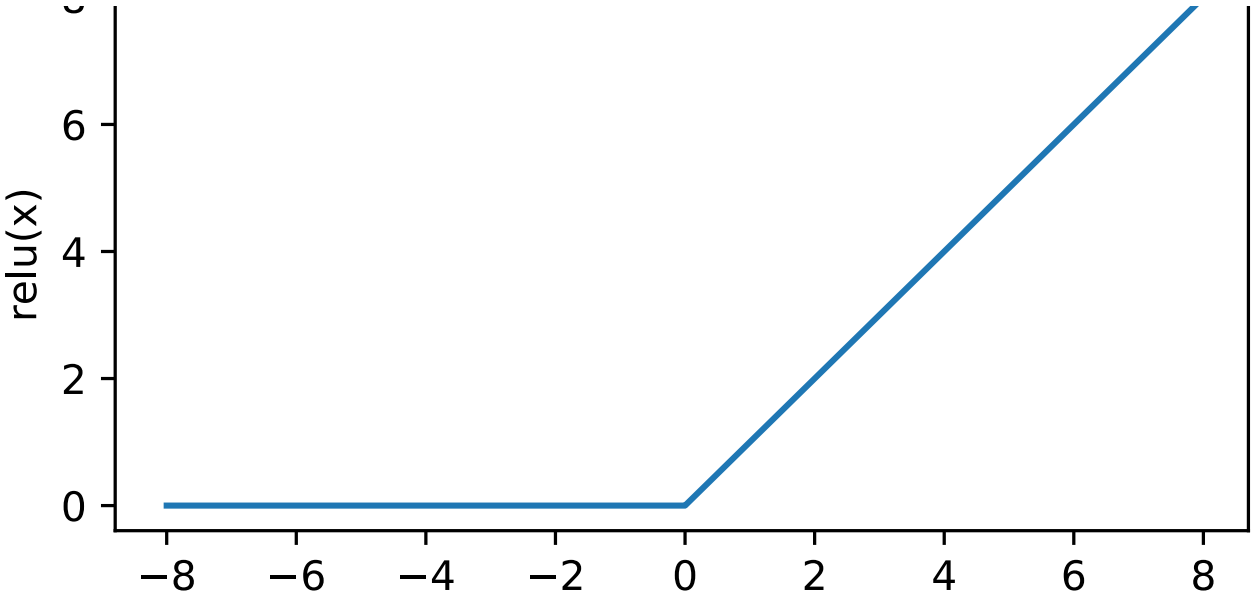

Figure 3 - 2: Relu 图像

可以看出在这个函数中只保留了正数，负数全部清零。

3.2 sigmoid 函数

simgmoid 函数可以表示为：

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{3 - 2}$$

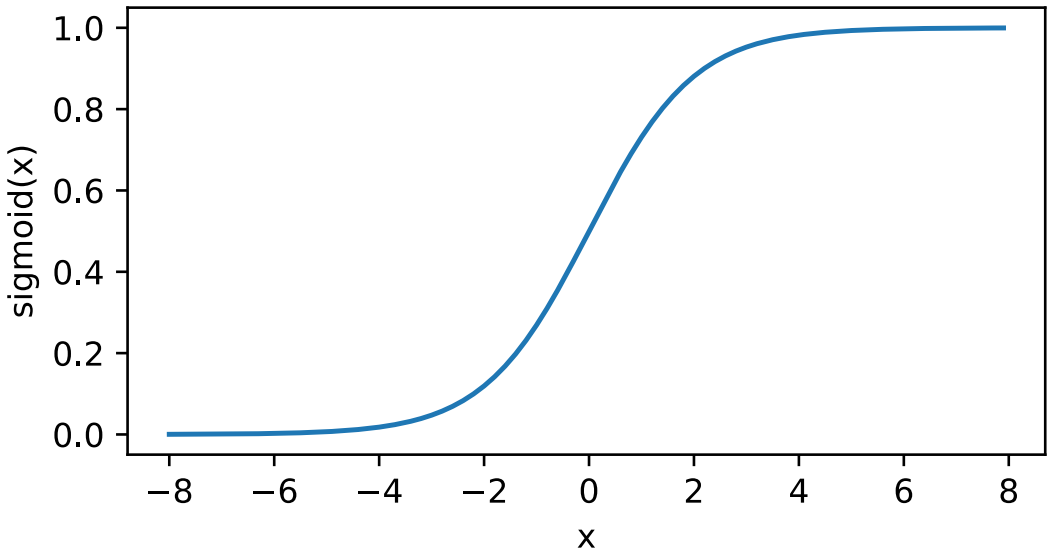

Figure 3 - 3: Sigmoid 图像

sigmoid 函数可以将元素的值变换到 0 和 1 之间

3.3 tanh 函数

tanh（双曲正切）函数可以表示为：

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{3 - 3}$$

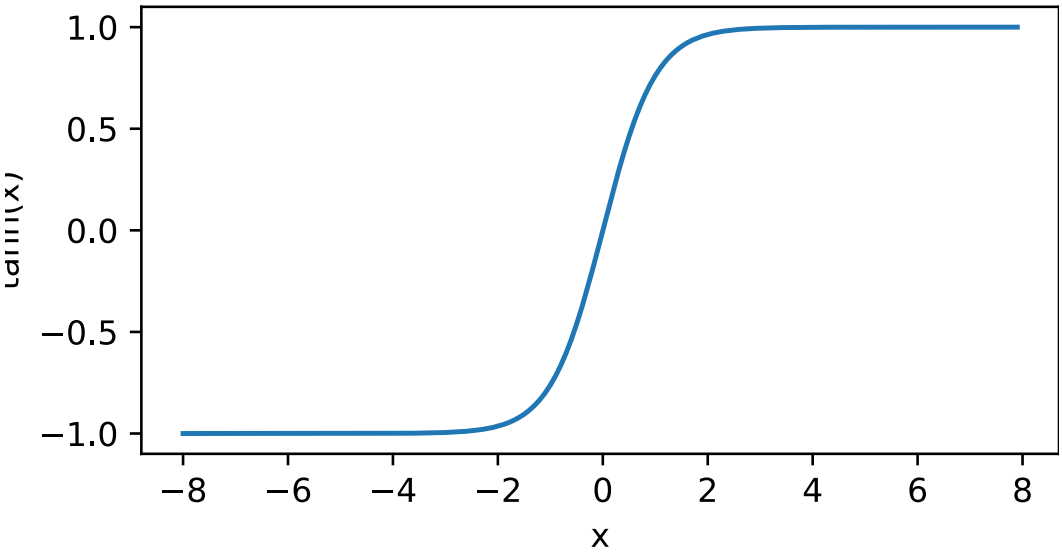

Figure 3 - 4: Tanh 图像

tanh（双曲正切）函数可以将元素的值变换到-1 和 1 之间。

四、 多层感知机

在分类问题中，我们可以对输出 O 做 softmax 运算，并使用 softmax 回归中的交叉熵损失函数。在回归问题中，我们将输出层的输出个数设为 1，并将输出 O 直接提供给线性回归中使用的平方损失函数。

五、 引用

[1] 4.1. 多层感知机 — 动手学深度学习 2.0.0 documentation[EB]. [2024-05-07]. https://zh.d2l.ai/chapter_multilayer-perceptrons/mlp.html#id2.

[2] 3.8. 多层感知机 — 《动手学深度学习》 文档[EB]. [2024-05-07]. https://zh-v1.d2l.ai/chapter_deep-learning-basics/mlp.html.

[3] MLP 公式推导及 numpy 实现 numpy 实现 mlp-CSDN 博客[EB]. [2024-05-07]. https://blog.csdn.net/qq_38955142/article/details/117475283.