第 3 章

Neural Network Methods for Natural Language Processing

从线性模型到多层感知器

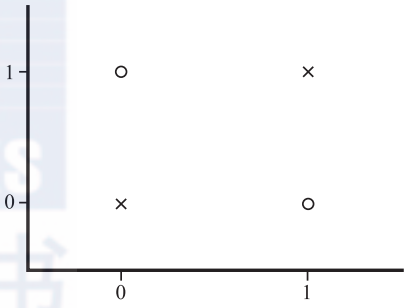
3.1 线性模型的局限性：异或问题

线性(对数-线性)模型的假设严格受限。例如，它不能表示异或(XOR)函数，其定义为：

$$\begin{aligned}\text{xor}(0,\ 0) &= 0\\ \text{xor}(1,\ 0) &= 1\\ \text{xor}(0,\ 1) &= 1\\ \text{xor}(1,\ 1) &= 0.\end{aligned}$$

也就是说，没有参数 $\boldsymbol{w}\in\mathbb{R}^2$，$b\in\mathbb{R}$ 满足：

$$\begin{aligned}(0,\ 0)\cdot\boldsymbol{w}+b &< 0\\ (1,\ 0)\cdot\boldsymbol{w}+b &\geqslant 0\\ (0,\ 1)\cdot\boldsymbol{w}+b &\geqslant 0\\ (1,\ 1)\cdot\boldsymbol{w}+b &< 0\end{aligned}$$



为了说明原因，考虑右侧异或函数的图形，其中o表示正类，×表示负类。

显然，没有一条直线能够分割这两个类别。

3.2 非线性输入转换

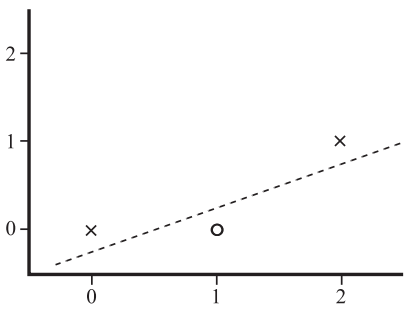
然而，如果我们通过将这些点输入给非线性函数 $\phi(x_1,\ x_2)=[x_1\times x_2,\ x_1+x_2]$进行转换，则异或问题就变成了线性可分的问题。

函数 ϕ 将数据映射为适合线性分类的表示。有了函数 ϕ，我们能够很容易训练一个线性分类器来解决异或问题。

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x})\phi(\boldsymbol{x})\boldsymbol{W}+\boldsymbol{b}$$

通常，我们可以定义一个函数，将非线性可分的数据集映射为线性可分的表示，然后

在最终表示上训练一个线性分类器。在异或的例子中，被转换后的数据和原始数据具有相同的维度，然而为了使数据线性可分，我们经常需要将其映射到更高的维度。



该解决方案有一个明显的问题，我们需要人工定义函数 ϕ，此过程需要依赖特定的数据集，并且需要大量人类的直觉。

3.3　核方法

核化的支持向量机[Boser 等人，1992]，或者通常的核(Kernel)方法[Shawe-Taylor and Cristianini，2004]通过定义一些通用的映射来解决这一问题，每个映射都将数据映射到非常高的维度空间(有时甚至是无限的)，然后在映射后的空间中执行线性分类。在非常高维的空间中进行分类显著提高了找到一个合适的线性分类器的概率。

一个映射的例子是多项式映射，$\phi(\boldsymbol{x})=(\boldsymbol{x})^d$。对于 $d=2$，我们得到 $\phi(x_1, x_2)=(x_1x_1, x_1x_2, x_2x_1, x_2x_2)$。其对所有的变量进行两两组合，通过增加参数的数量，可以使用线性分类器解决异或问题。在异或问题中，此映射增加了输入的维度(同时增加了参数的数量)，从 2 变为 4。对于语言识别的例子，输入维度将从 784 变为 $784^2=614\ 656$。

在非常高的维度上操作在计算上是无法完成的，核方法的创新性在于使用了*核技巧*(kernel trick)[Aizerman et al.，1964，Schölkopf，2001]，允许不用计算转换后的表示，而在转换后的空间上进行工作。对于许多常用的情况，人们设计了许多通用的映射，用户需要为具体的任务选择合适的映射，通常采用反复实验的方法。该方法的一个缺点是核技巧的应用使得支持向量机的分类过程线性依赖于训练集的大小，使其无法应用于非常大的训练集。高维空间的另一个缺点是它们增加了过拟合的风险。

3.4　可训练的映射函数

一种不同的方法是定义一个可训练的非线性映射函数，并和线性分类器一起训练。也就是说，找到合适的表示成为训练算法的责任。例如，映射函数可以采用参数化的线性模型形式，接一个作用于每一个输出维度上的非线性激活函数 g：

$$\hat{y}=\phi(\boldsymbol{x})\boldsymbol{W}+\boldsymbol{b}$$

$$\phi(\boldsymbol{x}) = g(\boldsymbol{x}\boldsymbol{W}' + \boldsymbol{b}') \tag{3.1}$$

通过采用 $g(x)=\max(0, x)$ 和 $\boldsymbol{W}'=\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$，$\boldsymbol{b}'=(-1 \quad 0)$，对于四个我们感兴趣的点(0，0)，(0，1)，(1，0)和(1，1)，我们可以获得一个和$(x_1 \times x_2, x_1 + x_2)$等价的映射，成功地解决异或问题。整个表达式 $g(\boldsymbol{x}\boldsymbol{W}'+\boldsymbol{b}')\boldsymbol{W}+\boldsymbol{b}$ 是可微的(也不是凸的)，使得应用基于梯度的技术进行模型训练成为可能，同时学习表示函数和在其之上的线性分类器。这是深度学习和神经网络背后的主要想法。事实上，式(3.1)描述了一个非常常用的神经网络结构，称为*多层感知器*(Multi-Layer Perceptron，MLP)。有了此动机，我们现在来更详细地描述多层神经网络。