第 4 单元　神经网络

支撑的课程目标

1. 能够基于智能信息处理的基本理论和技术，识别和理解数据处理与分析等问题的相关特性.
2. 能够运用智能信息处理的相关原理和专业知识，设计实验方案，为解决数据处理与分析等问题提供支持.

基本要求

1. 能够详述单层感知器和多层感知器的基本概念，理解感知器模型的特点.
2. 应用 BP 神经网络的基本原理，解决数据分析领域的分类问题.

教学重点与难点

重点：　多层感知器；前向传播；反向传播.

难点：　反向传播算法.

教学过程设计

新课导入、知识讲授、教学目标达成考核、总结.

教学过程设计

本单元教学通过“互动、开放”的课堂形式，采用探究式学习、问题导入的教学方法，激发学生的学习兴趣，促成课程目标的达成.

教学学时

6 学时.

一、导入新课（5 分钟）

人工智能的研究者为了模拟人类的认知，提出了不同的模型。人工神经网络是人工智能中非常重要的一个学派——连接主义最为广泛使用的模型。

在传统上，基于规则的符号主义学派认为，人类的认知是基于信息中的模式；而这些模式可以被表示成为符号，并可以通过操作这些符号，显式地使用逻辑规则进行计算与推理。

而基于统计的连接主义的模型将认知所需的功能属性结合到模型中来，通过模拟生物神经网络的信息处理方式来构建具有认知功能的模型。类似于生物神经元与神经网络，这类模型具有三个特点：

拥有处理信号的基础单元

处理单元之间以并行方式连接

处理单元之间的连接是有权重的

这一类模型被称为人工神经网络，多层感知器是最为简单的一种。

二、知识讲授（240 分钟）

本单元要点：

* 基础概念

* 单层感知器

* 多层感知器

* BP 神经网络

1 基础概念

神经网络：神经网络是一个有向图，以神经元为顶点，神经元的输入为顶点的入边，神经元的输出为顶点的出边。因此神经网络实际上是一个计算图，直观地展示了一系列对数据进行计算操作的过程。神经网络是一个端到端的系统，这个系统接受一定形式的数据作为输入，经过系统内的一系列计算操作后，给出一定形式的数据作为输出；系统内的运算可以被视为一个黑箱子，这与人类的认知在一定程度上具有相似性。通常地，为了直观起见，人们对神经网络中的各顶点进行了层次划分。

输入层：接受来自网络外部的数据的顶点，组成输入层。

输出层：向网络外部输出数据的顶点，组成输出层。

隐藏层：除了输入层和输出层以外的其他层，均为隐藏层。

2 单层感知器

感知器的概念由 Rosenblatt Frank 在 1957 提出，是一种监督训练的二元分类器。

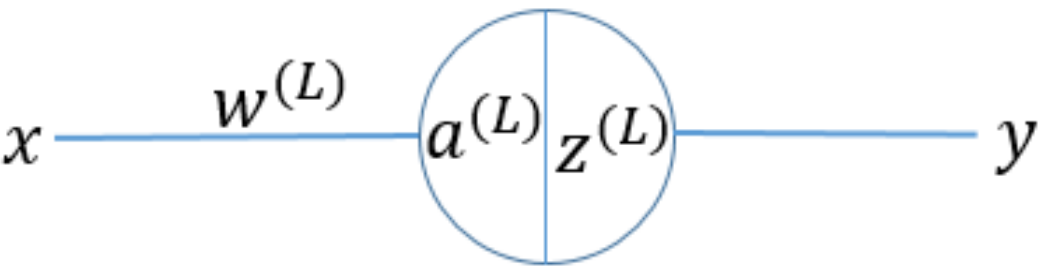

图 1: 单个神经元的网络示意图

图1考虑只有一个神经元的神经网络。其中各参数的含义如下：

* L-层数
* w-权重
* a- 预激活函数
* z- 激活函数
* y-输出

预激活值 a 可以写为：

$$a^{(L)} = w^{(L)}x + w_0^{(L)} \tag{1}$$

a 的值由激活函数激活，这里使用 sigmoid 激活函数，则有：

$$\begin{aligned} a^{(L)} &= w^{(L)}x + w_0^{(L)} \\ z^{(L)} &= \sigma(a^{(L)}) \\ \hat{y} &= z^{(L)} \end{aligned} \tag{2}$$

这个网络的输出是 $\hat{y}$。用字母 J 表示损失函数，现在计算反向传播，损失函数的梯度可以表示为：

$$\nabla J = \frac{\partial J}{\partial w} \tag{3}$$

J 是 w 的复合函数，我们使用链式法则来计算梯度。我们使用链式法则是因为误差不受权重的直接影响，权重影响预激活函数，进而影响激活函数，进而影响输出，最后影响损失。下面的树显示了每个术语如何依赖于上面网络中的另一个术语。

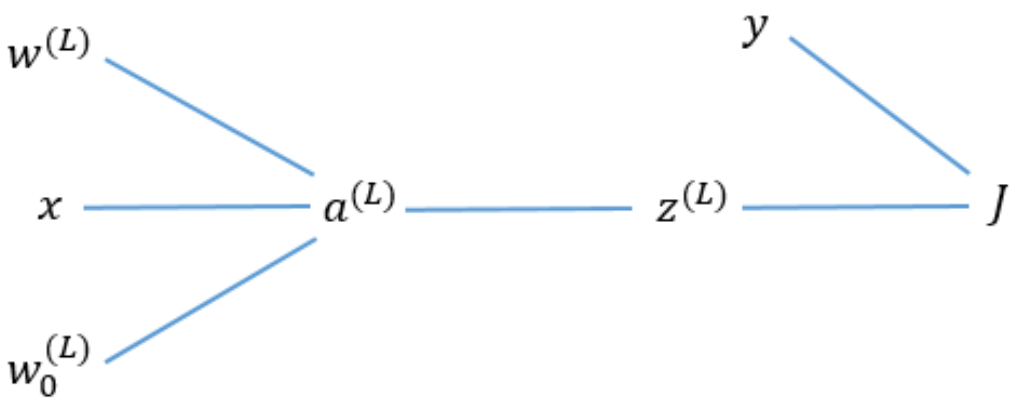

图 2: 各个变量之间的关联图

预激活函数取决于输入、权重和偏差、激活函数依赖于预激活函数、损失取决于激活函数。图2右上角的 y 是与预测输出进行比较并计算损失的真实值。

所以当我们应用链式法则时，我们得到：

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial z^{(L)}} \cdot \frac{\partial z^{(L)}}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial w^{(L)}} \tag{4}$$

我们有另一个词来指代这个梯度，即损失相对于权重的瞬时变化率。

3 多层感知器

下面我们将上述从单个神经元网络的梯度计算中获得的知识扩展到具有两层的真正神经网络：一个输入层、一个隐藏层和一个输出层。图3给出了一个 2

层的全连接网络结构的示意图。其中：输入层（第 0 层）有两个神经元。隐藏层（第一层）有两个神经元。输出层（第二层）有两个神经元。

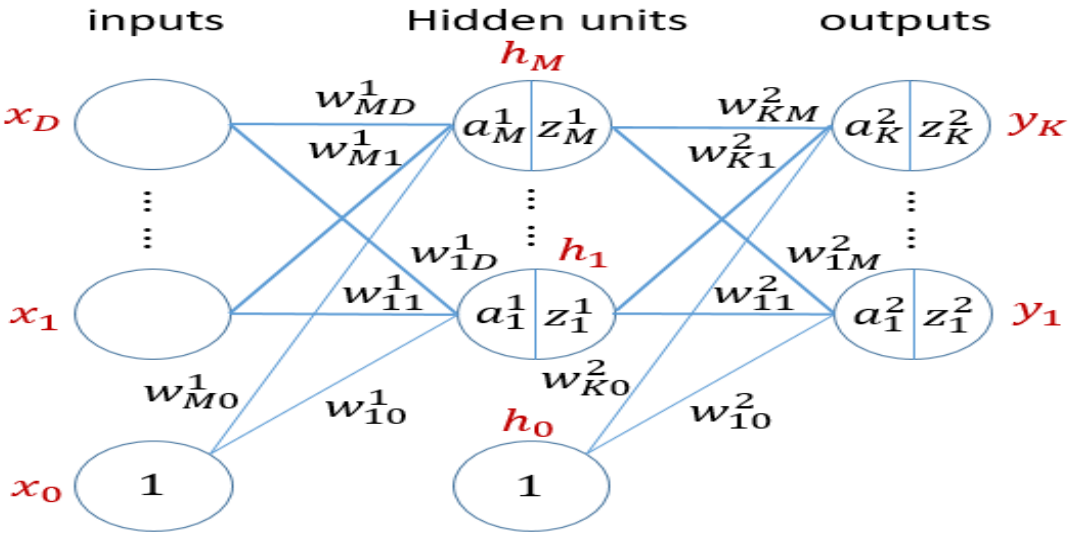

图 3: 全连接网络示意图

3.1 神经网络的前向传播

下面构建 2 层的神经网络。首先构建输入 $x_1, \cdots, x_D$ 的 M 个线性组合：

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{5}$$

其中 $j = 1, \cdots, M$，上标 (1) 表示相应的参数是网络第一层的参数。我们也称参数 $w_{ji}^{(1)}$ 为权值，参数 $w_{j0}^{(1)}$ 为偏置，$a_j^{(1)}$ 称为激活值。使用非线性的可微分的激活函数 $h(\cdot)$ 变换 $a_j^{(1)}$ 得到：

$$z_j^{(1)} = h(a_j^{(1)}) \tag{6}$$

在神经网络中，$z_j^{(1)}$ 被称为隐单元 h_j 的输出，非线性函数 $h(\cdot)$ 通常被选为 sigmoidal 函数，例如 logistic sigmoid 或 tanh 函数。

这些值 $z_j^{(1)}$ 被线性加强后获得输出单元的激活值 $a_k^{(2)}$:

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j^{(1)} + w_{k0}^{(2)} \tag{7}$$

其中 $k = 1, \cdots, K$，上标 (2) 表示相应的参数是网络第二层的参数。我们也称参数 $w_{kj}^{(2)}$ 为权值，参数 $w_{k0}^{(2)}$ 为偏置。$a_k^{(2)}$ 称为激活值。然后，使用合适的激活函数将输出单元的预激活值 $a_k^{(2)}$ 变换为一组神经网络的输出值 y_k:

$$y_k = h(a_k^{(2)}) \tag{8}$$

对标准的回归问题，激活函数为 $h()$ 恒等函数，因此 $y_k = a_k^{(2)}$。

对多二分类问题，激活函数 $h()$ 选用 logistic sigmoid 函数，因此 $y_k = \sigma(a_k^{(2)})$ 其中

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{9}$$

我们组合这以上两个阶段给出整个网络函数

$$y_k(\mathbf{x}, \mathbf{w}) = h(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)}) \tag{10}$$

其中所有权重和偏置参数可以被组合成一个向量，记作 $\mathbf{w}$。因此，从这个角度而言，神经网络模型只是一个非线性函数（从一组输入变量 $\{x_i\}$ 到一组输出变量 $\{y_k\}$），受可调节参数 $\mathbf{w}$ 控制。上式估计 $y_k(\mathbf{x}, \mathbf{w})$ 的过程可以被解释为信息通过网络的前向传播。

通过定义辅助的输入变量 x_0（其值被限制为 $x_0 = 1$），式 (5) 中的偏置参数可以被吸收到权值参数中，式 (5) 可以重写为

$$a_j^{(1)} = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \tag{11}$$

我们同样将第二层的偏置参数吸收到第二层的权值中

$$\begin{aligned} a_k^{(2)} &= \sum_{j=0}^{M} w_{kj}^{(2)} z_j^{(1)} \\ &= \sum_{j=0}^{M} w_{kj}^{(2)} h(a_j^{(1)}) \\ &= \sum_{j=0}^{M} w_{kj}^{(2)} h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i) \end{aligned} \tag{12}$$

因此，整个网络函数可以写为

$$y_k(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_{kj}^{(2)} h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i) \tag{13}$$

从图3可以看出，该神经网络模型包括两个处理阶段，每个阶段都类似于感知器模型，因为这个原因，该神经网络模型被称为多层感知器。

3.2 神经网络的后向传播

在多层感知器被引入的同时，也引入了一个新的问题：由于隐藏层的预期输出并没有在训练样例中给出，隐藏层结点的误差无法像单层感知器那样直接计算得到。为了解决这个问题，后向传播算法被引入，其核心思想是将误差由输出层向前层后向传播，利用后一层的误差来估计前一层的误差。后向传播算法由 Henry J. Kelley 在 1960 和 Arthur E. Bryson 在 1961 分别提出。使用后向传播算法训练的网络称为 BP 神经网络。

但是，与感知器的关键不同之处在于，在神经网络中隐层单元使用的是连续的 sigmoidal 非线性函数，而感知器中使用的是阶跃函数的非线性。这意味着神经网络函数对网络参数是可微分的，这个性质在神经网络训练中发挥核心作用。因此，可以利用微分的性质进行网络训练。

到目前为止，我们将神经网络视为带参数 **w** 的非线性函数，从输入变量 **x** 到输出变量 **y** 。一种确定网络参数问题的简单方法是梯度下降算法

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) \tag{14}$$

在梯度下降算法中，待解决的关键问题就是求误差函数 E 对于参数 $\mathbf{w}$ 的梯度。下面讲解梯度的求解过程。

给定一个训练集 $\{\mathbf{x}_n\}$, 其中 $n = 1, \cdots, N$，及一组标签 $\mathbf{t} = \{t_1, \cdots, t_N\}$。我们得到如下的误差函数：

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) \tag{15}$$

我们的目标就是计算 $\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \nabla E_n(\mathbf{w})$。由上式可知，误差函数包括 N 项之和。下面我们将考虑误差函数中的一项的梯度 $\nabla E_n = \frac{\partial E_n}{\partial w_{ji}}$ 的计算问题。

3.2.1 一般的前向传播与后向传播

(1) 前向传播

在一般的前馈网络中，每个神经元要计算一组输入数据的加权之和：

$$a_j^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)} \tag{16}$$

其中 $z_i^{(l-1)}$ 表示向第 l 层神经元 j 发送连接的第 $l-1$ 层神经元的激活值 (即前一层神经元的激活值)。利用非线性激活函数函数 $h(\cdot)$ 对 $a_j^{(l)}$ 做变换得到 l 层神经元 j 的激活值 $z_j^{(l)}$

$$z_j^{(l)} = h(a_j^{(l)}) \tag{17}$$

注意：式 (16) 中的变量 $z_i^{(l-1)}$ 可能是输入数据（第一层），式 (17) 中的 l 层神经元 j 可能是一个输出神经元 (即 l 层有可能是输出层)。

对于训练集中的每个输入数据（假设我们提供了神经网络的输入向量），连续依次使用式 (16) 和式 (17) 计算隐层和输出层神经元的激活值。这一过程就是前向传播。

(2) 反向传播

下面考虑计算 E_n 关于权重 $w_{ji}^{(l)}$ 的导数。首先注意到 E_n 仅通过 l 层神经元 j 的预激活值 $a_j^{(l)}$ 关联到权重 $w_{ji}^{(l)}$。因此，我们通过偏导数的链式法则得到

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} \tag{18}$$

现在引入一个有用的符号

$$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial a_j^{(l)}} \tag{19}$$

其中，$\delta_j^{(l)}$ 也被称为误差。利用式 (16) 对 $w_{ji}^{(l)}$ 求导得到

$$\frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)} \tag{20}$$

将 (19) 和 (20) 代入 (18)，得到

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{21}$$

从式 (21) 可以看出，$w_{ji}^{(l)}$ 的导数等于 $\delta_j^{(l)}$ 的值（通过权值 w_{ji} 的输出端神经元得到的，即第 l 层神经元）乘以 $z_i^{(l-1)}$ 的值（通过权值 $w_{ji}^{(l)}$ 的输入端神经元得到的，即第 $l-1$ 层神经元）。因此，为了计算导数，我们仅需要在网络中对每个隐神经元和输出神经元计算 δ 值，然后利用式 (21) 得到导数。

下面讲解 δ 值的求法。对于输出神经元和隐层神经元，δ 值的计算公式是不同的。假设此处的误差函数为平方误差函数，则对输出神经元，得到

$$\delta_k^{(l)} = (y_k - t_k)h'(a_k^{(l)}) \tag{22}$$

计算过程如下：

$$
\begin{aligned}
E_n &= \sum_k E_n^k = \frac{1}{2}\sum_k (y_k - t_k)^2 \\
y_k &= h(a_k^{(l)}) \\
\delta_k^{(l)} &= \frac{\partial E_n}{\partial a_k^{(l)}} \\
&= \frac{\partial \sum_k E_n^k}{\partial a_k^{(l)}} \\
&= \frac{\partial E_n^k}{\partial a_k^{(l)}} \\
&= \frac{\partial E_n^k}{\partial y_k}\frac{\partial y_k}{\partial a_k^{(l)}} \\
&= \frac{\partial E_n^k}{\partial y_k}\frac{\partial y_k}{\partial a_k^{(l)}} \\
&= (y_k - t_k)h'(a_k^{(l)})
\end{aligned}
\tag{23}
$$

对隐层的神经元，我们的目标是寻找隐层神经元的误差与输出层神经元误

差之间的关系。我们使用偏导数的链式法则，得到

$$
\begin{aligned}
E_n &= \sum_k E_n^k = \frac{1}{2}\sum_k (y_k - t_k)^2 \\
y_k &= h(a_k^{(l)}) \\
a_k^{(l)} &= \sum_j w_{kj}^{(l)} z_j^{(l-1)} \\
z_j^{(l-1)} &= h(a_j^{(l-1)}) \\
\delta_j^{(l-1)} &\equiv \frac{\partial E_n}{\partial a_j^{(l-1)}} \\
&= \frac{\partial \sum_k E_n^k}{\partial a_j^{(l-1)}} \\
&= \sum_k \frac{\partial E_n^k}{\partial a_j^{(l-1)}} \\
&= \sum_k \frac{\partial E_n^k}{\partial a_k^{(l)}} \frac{\partial a_k^{(l)}}{\partial a_j^{(l-1)}} \\
&= \sum_k \frac{\partial E_n^k}{\partial a_k^{(l)}} \frac{\partial a_k^{(l)}}{\partial z_j^{(l-1)}} \frac{\partial z_j^{(l-1)}}{\partial a_j^{(l-1)}} \\
&= \sum_k \delta_k^{(l)} w_{kj}^{(l)} h'(a_j^{(l-1)}) \\
&= h'(a_j^{(l-1)}) \sum_k w_{kj}^{(l)} \delta_k^{(l)}
\end{aligned}
\tag{24}
$$

其中求和要对所有的从第 $l-1$ 层神经元 j 出发连接到的第 l 层神经元 k 进行。注意：标记为 k 的神经元可能是其它隐层的神经元的或输出神经元。上式指出某个隐层神经元的 δ 值可以通过更高层神经元的 δ 值反向传播计算出。

注意：求和的下标是 $w_{kj}^{(l)}$ 的第一个下标（对应于网络的信息的反向传播），而在前向传播中，求和的下标是第二个下标。因为我们已经知道了输出神经元的 $\delta_k^{(l)}$ 值，通过递归的使用上式，可以估计出所有隐层单元的 $\delta_j^{(l-1)}$ 值。

反向传播过程可以总结如下：

（1）给定网络的输入向量，利用 $a_j = \sum_i w_{ji} z_i$ 和 $z_j = h(a_j)$ 进行网络的前向传播找到所有隐层神经元和输出神经元的激活值。

（2）利用 $\delta_k^{(l)} = (y_k - t_k) h'(a_k^{(l)})$ 计算所有输出神经元的 $\delta_k^{(l)}$ 值。

（3）使用 $\delta_j^{(l-1)} = h'(a_j^{(l-1)}) \sum_k w_{kj}^{(l)} \delta_k^{(l)}$ 反向传播 $\delta_k^{(l)}$ 得到每个隐层神经元的 $\delta_j^{(l-1)}$。

（4）使用 $\frac{\partial E_n}{\partial w_{ji}^{(l-1)}} = \delta_j^{(l-1)} z_i^{(l-2)}$ 计算参数 $w_{ji}^{(l-1)}$ 的导数。

3.2.2 简单的例子：

图4给出了一个 2 层的全连接网络结构的示意图。其中：输入层（第 0 层）有两个神经元。隐藏层（第一层）有两个神经元。输出层（第二层）有两个神经元。

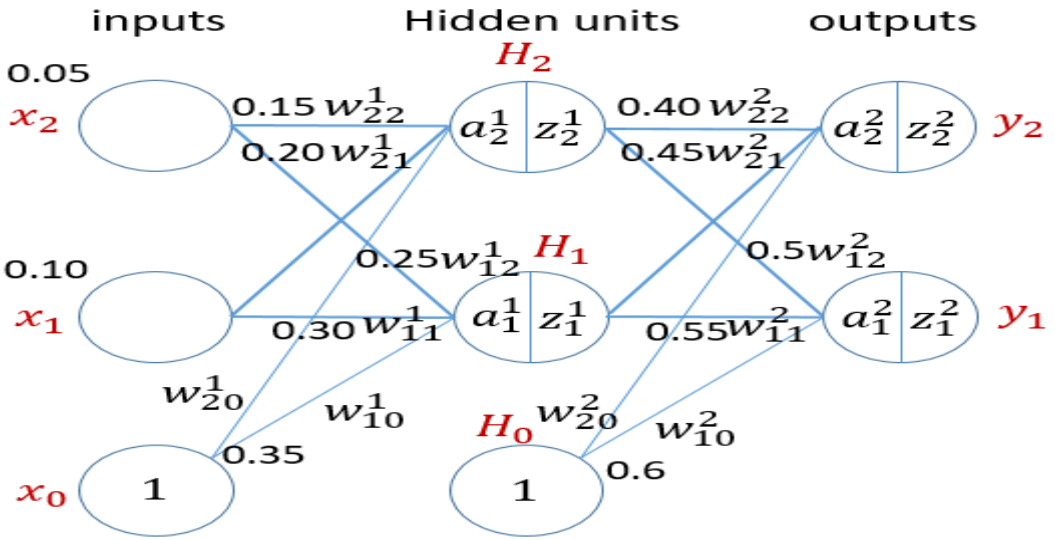

图 4: 全连接网络示意图

我们考虑一个标准的平方误差函数，对每个样本，误差有下式给出

$$E_n = \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2 \tag{25}$$

对训练集中每个样本，前向传播过程如下：

$$a_j^{(1)} = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \tag{26}$$

$$z_j^{(1)} = h(a_j^{(1)}) \tag{27}$$

$$a_k^{(2)} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j^{(1)} \tag{28}$$

$$y_k = z_k^{(2)} = h(a_k^{(2)}) \tag{29}$$

$$\tag{30}$$

后向传播过程如下：

$$\delta_k^{(2)} = (y_k - t_k) h'(a_k^{(2)}) \tag{31}$$

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k^{(2)} z_j^{(1)} \tag{32}$$

$$\delta_j^{(1)} = h'(a_j^{(1)}) \sum_k w_{kj}^{(2)} \delta_k^{(2)} \tag{33}$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} x_i \tag{34}$$

假设初始值设定如下：

$$x_0 = 1, x_1 = 0.10, x_2 = 0.05; z_0 = 1 \tag{35}$$

$$w_{20}^{(1)} = 0.35, w_{21}^{(1)} = 0.20, w_{22}^{(1)} = 0.15; w_{10}^{(1)} = 0.35, w_{11}^{(1)} = 0.3, w_{12}^{(1)} = 0.25 \tag{36}$$

$$w_{20}^{(2)} = 0.60, w_{21}^{(2)} = 0.45, w_{22}^{(2)} = 0.40; w_{10}^{(2)} = 0.60, w_{11}^{(2)} = 0.55, w_{12}^{(2)} = 0.5 \tag{37}$$

计算前向传播值

根据第一层的初始参数值和输入值，可得到隐层的输出：

$$a_2^{(1)} = \sum_{i=0}^{2} w_{2i}^{(1)} x_i = w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 \tag{38}$$

$$= 0.35 * 1 + 0.2 * 0.1 + 0.15 * 0.05 = 0.3775 \tag{39}$$

$$z_2^{(1)} = h(a_2^{(1)}) = \frac{1}{1 + exp(-a_2^{(1)})} = 0.593269992 \tag{40}$$

$$a_1^{(1)} = \sum_{i=0}^{2} w_{1i}^{(1)} x_i = w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 \tag{41}$$

$$= 0.35 * 1 + 0.3 * 0.1 + 0.25 * 0.05 = 0.3925 \tag{42}$$

$$z_1^{(1)} = h(a_1^{(1)}) = \frac{1}{1 + exp(-a_1^{(1)})} = 0.596884378 \tag{43}$$

由此，可知 $z_2^{(1)} = 0.593269992, z_1^{(1)} = 0.596884378, z_0^{(1)} = 1$。将 $z_0^{(1)}, z_1^{(1)}, z_2^{(1)}$ 作为输出层的输入，得到输出层的输出：

$$a_2^{(2)} = \sum_{j=0}^{2} w_{2j}^{(2)} z_j^{(1)} = w_{20}^{(2)} z_0^{(1)} + w_{21}^{(2)} z_1^{(1)} + w_{22}^{(2)} z_2^{(1)} \tag{44}$$

$$= 0.6 * 1 + 0.45 * 0.596884378 + 0.4 * 0.593269992 = 1.105905967 \tag{45}$$

$$y_2 = z_2^{(2)} = h(a_2^{(2)}) = \frac{1}{1 + exp(-a_2^{(2)})} = 0.751365070 \tag{46}$$

$$a_1^{(2)} = \sum_{j=0}^{2} w_{1j}^{(2)} z_j^{(1)} = w_{10}^{(2)} z_0^{(1)} + w_{11}^{(2)} z_1^{(1)} + w_{12}^{(2)} z_2^{(1)} \tag{47}$$

$$= 0.6 * 1 + 0.55 * 0.596884378 + 0.5 * 0.593269992 = 1.224921404 \tag{48}$$

$$y_1 = z_1^{(2)} = h(a_1^{(2)}) = \frac{1}{1 + exp(-a_1^{(2)})} = 0.772928465 \tag{49}$$

计算总误差为：$E=\frac{1}{2}\sum_{i=1}^{2}(target-output)^2$ 假设输出层的实际输出值为 $t_1=0.99, t_2=0.01$，则网络的误差可以计算为

$$E_{y_2}=\frac{1}{2}(t_2-y_2)^2=\frac{1}{2}(0.01-0.751365070)^2=0.274811083$$

$$E_{y_1}=\frac{1}{2}(t_1-y_1)^2=\frac{1}{2}(0.99-0.772928465)^2=0.023560026$$

网络的总误差计算为：

$$E=E_{y_1}+E_{y_2}=0.274811083+0.023560026=0.298371109$$

后向传播过程如下：

利用 $\delta_k^{(2)}=(y_k-t_k)h'(a_k^{(2)})$ 计算所有输出神经元 y_1, y_2 的 $\delta_1^{(2)}, \delta_2^{(2)}$ 值。由于 $h(a_k)=\frac{1}{1+exp(-a_k)}$，得到 $h'(a_k)=h(a_k)(1-h(a_k))$。使用 $\frac{\partial E_n}{\partial w_{kj}^{(2)}}=\delta_k^{(2)}z_j^{(1)}$ 计算 $w_{kj}^{(2)}$ 的导数，设 $\eta=0.5$。

根据输出神经元 2，可以计算：

$$\delta_2^{(2)}=(y_2-t_2)h'(a_2^{(2)})=(y_2-t_2)h(a_2^{(2)})(1-h(a_2^{(2)})) \tag{50}$$

$$=(0.751365070-0.01)*0.751365070*(1-0.751365070)=0.138498562 \tag{51}$$

$$\frac{\partial E_n}{\partial w_{22}^{(2)}}=\delta_2^{(2)}z_2^{(1)}=0.138498562*0.593269992=0.082167041 \tag{52}$$

$$\frac{\partial E_n}{\partial w_{21}^{(2)}}=\delta_2^{(2)}z_1^{(1)}=0.138498562*0.596884378=0.082667628 \tag{53}$$

$$w_{22}^{(2)}(t+1)=w_{22}^{(2)}(t)-\eta*\frac{\partial E_n}{\partial w_{22}^{(2)}}=0.4-0.5*0.082167041=0.35891648 \tag{54}$$

$$w_{21}^{(2)}(t+1)=w_{21}^{(2)}(t)-\eta*\frac{\partial E_n}{\partial w_{21}^{(2)}}=0.45-0.5*0.082667628=0.408666186$$

(55)

根据输出神经元 1，可以计算：

$$\delta_1^{(2)} = (y_1 - t_1)h'(a_1^{(2)}) = (y_1 - t_1)h(a_1^{(2)})(1 - h(a_1^{(2)})) \tag{56}$$

$$= (0.772928465 - 0.99) * 0.772928465 * (1 - 0.772928465) = -0.038098237 \tag{57}$$

$$\frac{\partial E_n}{\partial w_{12}^{(2)}} = \delta_1^{(2)} z_2^{(1)} = -0.038098237 * 0.593269992 = -0.022602541 \tag{58}$$

$$\frac{\partial E_n}{\partial w_{11}^{(2)}} = \delta_1^{(2)} z_1^{(1)} = -0.038098237 * 0.596884378 = -0.022740242 \tag{59}$$

$$w_{12}^{(2)}(t+1) = w_{12}^{(2)}(t) - \eta * \frac{\partial E_n}{\partial w_{12}^{(2)}} = 0.5 - 0.5 * (-0.022602541) = 0.511301271 \tag{60}$$

$$w_{11}^{(2)}(t+1) = w_{11}^{(2)}(t) - \eta * \frac{\partial E_n}{\partial w_{11}^{(2)}} = 0.55 - 0.5 * (-0.022740242) = 0.561370121 \tag{61}$$

使用 $\delta_j^{(1)} = h'(a_j^{(1)}) \sum_k w_{kj}^{(2)} \delta_k^{(2)}$ 计算隐层所有神经元的 $\delta_2^{(1)}, \delta_1^{(1)}$。使用 $\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} x_i$ 计算 $w_{ji}^{(1)}$ 的导数。

根据隐层的神经元 2，可以计算

$$\delta_2^{(1)} = h'(a_2^{(1)}) \sum_{k=1}^{2} w_{k2}^{(2)} \delta_k^{(2)} = z_2^{(1)}(1 - z_2^{(1)})(w_{12}^{(2)} \delta_1^{(2)} + w_{22}^{(2)} \delta_2^{(2)}) = 0.593269992* \tag{62}$$

$$(1 - 0.593269992) * (0.5 * (-0.038098237) + 0.4 * 0.138498562) = 0.008771355 \tag{63}$$

$$\frac{\partial E_n}{\partial w_{22}^{(1)}} = \delta_2^{(1)} * x_2 = 0.008771355 * 0.05 = 0.000438568 \tag{64}$$

$$\frac{\partial E_n}{\partial w_{21}^{(1)}} = \delta_2^{(1)} * x_1 = 0.008771355 * 0.1 = 0.000877136 \tag{65}$$

$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} x_i$

$$w_{22}^{(1)}(t+1) = w_{22}^{(1)}(t) - \eta * \frac{\partial E_n}{\partial w_{22}^{(1)}} = 0.15 - 0.5 * 0.000438568 = 0.149780716 \tag{66}$$

$$w_{21}^{(1)}(t+1) = w_{21}^{(1)}(t) - \eta * \frac{\partial E_n}{\partial w_{21}^{(1)}} = 0.20 - 0.5 * 0.000877136 = 0.199561432 \tag{67}$$

根据隐层的神经元 1，可以计算

$$\delta_1^{(1)} = h'(a_1^{(1)}) \sum_{k=1}^{2} w_{k1}^{(2)} \delta_k^{(2)} = z_1^{(1)}(1 - z_1^{(1)})(w_{11}^{(2)} \delta_1^{(2)} + w_{21}^{(2)} \delta_2^{(2)}) = 0.596884378* \tag{68}$$

$$(1 - 0.596884378) * (0.55 * (-0.038098237) + 0.45 * 0.138498562) = 0.009954255 \tag{69}$$

$$\frac{\partial E_n}{\partial w_{12}^{(1)}} = \delta_1^{(1)} * x_2 = 0.009954255 * 0.05 = 0.000497713 \tag{70}$$

$$\frac{\partial E_n}{\partial w_{11}^{(1)}} = \delta_1^{(1)} * x_1 = 0.009954255 * 0.1 = 0.000995426 \tag{71}$$

$$w_{12}^{(1)}(t+1) = w_{12}^{(1)}(t) - \eta * \frac{\partial E_n}{\partial w_{12}^{(1)}} = 0.25 - 0.5 * 0.000497713 = 0.249751144 \tag{72}$$

$$w_{11}^{(1)}(t+1) = w_{11}^{(1)}(t) - \eta * \frac{\partial E_n}{\partial w_{11}^{(1)}} = 0.30 - 0.5 * 0.000995426 = 0.299502287 \tag{73}$$

最后，我们更新了所有的权重！当我们用 0.05 和 0.1 作为前向传播的输入时，网络误差为 0.298371109。在第一轮反向传播之后，总误差下降为 0.291027774。它可能看起来不多，但是在重复此过程 10,000 次之后，总误差会下降到 0.0000351019。

此时，当我们用 0.05 和 0.1 作为输入时，两个输出神经元的值为 0.015912196（vs 0.01 目标）和 0.984065734（vs 0.99 目标）。

三、教学目标考核（50 分钟）

讨论：

1. 一元感知器有什么特点？为什么需要多层感知器？
2. 多层感知器是如何构建的？
3. 多层感知器是如何实现自我学习的？

四、总结（5 分钟）

感知器模型可以算得上是深度学习的基石。最初的单层感知器模型就是为了模拟人脑神经元提出的，但是就连异或运算都无法模拟。经过多年的研究，人们终于提出了多层感知器模型，用于拟合任意函数。结合高效的 BP 算法，神经网络终于诞生。尽管目前看来，BP 神经网络已经无法胜任许多工作，但是从发展的角度来看，BP 神经网络仍是学习深度学习不可不知的重要部分。