MLP 与反向传播公式期末复习讲义

目录

1 多层感知器（MLP）基本概念

1.1 单层网络（线性模型）

考虑输入向量为

$$\boldsymbol{x} \in \mathbb{R}^{I_1},$$

网络参数（权重矩阵）为

$$W \in \mathbb{R}^{I_2 \times I_1},$$

则网络输出向量为

$$\boldsymbol{a} \in \mathbb{R}^{I_2},$$

其中各分量满足

$$a_{i_2} = \sum_{i_1=1}^{I_1} W_{i_2 i_1} X_{i_1}, \quad i_2 = 1, 2, \cdots, I_2.$$

向量形式可写为

$$\boldsymbol{a} = W\boldsymbol{X}.$$

以上构成了一层网络（线性模型），其中

$$\boldsymbol{X} \quad (\text{有时也写为 } \boldsymbol{x})$$

是网络输入，

$$\boldsymbol{a}$$

是网络输出，

$$\boldsymbol{W}$$

是网络参数。

1.2 二层网络

在单层网络的基础上，考虑再增加一层。记第一层的输出为

$$\boldsymbol{a}^{(1)}, \quad \boldsymbol{a}^{(1)} = W^{(1)}\boldsymbol{X}.$$

通过某个非线性激活函数

$$h(\cdot) : \mathbb{R} \to \mathbb{R}$$

获得

$$\boldsymbol{z} = h(\boldsymbol{a}^{(1)}).$$

然后再经过第二层的线性变换，即

$$\boldsymbol{y} = W^{(2)}\boldsymbol{z}.$$

将以上步骤具体展开：

Step1： 第一层线性计算

$$a_{i_2} = \sum_{i_1=1}^{I_1} W^{(1)}_{i_2 i_1} X_{i_1}, \quad i_2 = 1, 2, \cdots, I_2.$$

Step2： 激活函数作用

$$z_{i_2} = h\big(a_{i_2}\big),$$

其中

$$h : \mathbb{R} \to \mathbb{R}$$

是非线性函数。常见的激活函数例如：

$$h(z) = \frac{1}{1+e^{-z}} \quad (\text{Sigmoid 激活函数}),$$

$$h(z) = \max\{z, 0\} \quad (\text{ReLU 激活函数}).$$

Step3： 第二层线性计算

$$y_{i_3} = \sum_{i_2=1}^{I_2} W^{(2)}_{i_3 i_2}\, z_{i_2} = \sum_{i_2=1}^{I_2} W^{(2)}_{i_3 i_2}\, h\Big(\sum_{i_1=1}^{I_1} W^{(1)}_{i_2 i_1}\, X_{i_1}\Big).$$

此时，若

$$\boldsymbol{X} \in \mathbb{R}^{I_1},$$

则网络输出

$$\boldsymbol{y} \in \mathbb{R}^{I_3}.$$

这里 $W^{(1)}_{i_2 i_1}$，$W^{(2)}_{i_3 i_2}$ 为网络参数，h 为激活函数。向量形式可写为

$$\boldsymbol{y} = W^{(2)}\, h\big(W^{(1)} \boldsymbol{X}\big).$$

这就是二层网络（常称 MLP 的一种简单形式），也记作

$$\boldsymbol{y}(\boldsymbol{X}; \boldsymbol{W}).$$

1.3 L-层神经网络

将二层网络扩展至更深的情形，一般可以写成：对于深度为 L 的网络（输入层算作第 1 层，输出层算作第 L 层），记网络为

$$\boldsymbol{y}(\boldsymbol{X}; \boldsymbol{W}) : \mathbb{R}^{I_1} \to \mathbb{R}^{I_L}.$$

其中，

$$y_{i_L}(\boldsymbol{X}, \boldsymbol{W}) = \sum_{i_{L-1}} W^{(L-1)}_{i_L i_{L-1}}\, h^{(L-2)}\Big(\sum_{i_{L-2}} W^{(L-2)}_{i_{L-1} i_{L-2}} \cdots \sum_{i_2} W^{(2)}_{i_3 i_2}\, h^{(1)}\Big(\sum_{i_1} W^{(1)}_{i_2 i_1} X_{i_1}\Big)\Big).$$

为方便，我们引入记号：

$$
\begin{aligned}
&a_{i_1}^{(1)} = X_{i_1}, \\
&a_{i_l}^{(l)} = \sum_{i_{l-1}} W_{i_l i_{l-1}}^{(l-1)} z_{i_{l-1}}^{(l-1)}, \quad l = 2, 3, \cdots, L \text{ (隐藏层计算)}, \\
&z_{i_l}^{(l)} = h^{(l-1)}\big(a_{i_l}^{(l)}\big), \quad l = 1, 2, \cdots, L-1 \text{ (激活层)}, \\
&h^{(0)} = I \ \text{ (恒等算子)}.
\end{aligned}
$$

其中 L 是网络深度，I 通常称为宽度（例如每层的神经元数）。这样我们便能将 L-层网络写得更紧凑。

2 网络训练

2.1 损失函数

给定训练数据集 (样本对)

$$
\{(\boldsymbol{X}_n, \boldsymbol{T}_n)\}_{n=1}^{N},
$$

对于回归任务，常用的均方误差 (MSE) 损失函数为

$$
E(W) = \sum_{n=1}^{N} \sum_{i_L=1}^{I_L} \Big(y_{ni_L}(\boldsymbol{X}_n; W) - T_{ni_L} \Big)^2. \tag{44}
$$

训练的目标就是最小化该损失函数：

$$
W^* = \arg\min_W E(W). \tag{45}
$$

2.2 训练（优化）方法

通常采用基于梯度的优化方法，如**梯度下降**：

$$
W^{t+1} = W^t - \iota \nabla E\big(W^t\big), \tag{46}
$$

其中 ι 是学习率。将当前参数沿着 $\nabla E(W)$ 的反方向进行迭代更新，期望损失函数降低。

在深度学习中更常用的是**随机梯度下降** (SGD) 或其变体：每次迭代不是利用所有样本，而是仅抽取一个批量（batch）或单一样本（mini-batch 为极端是单一样本）来近似计算梯度。遍历所有样本完成一次称为一个 epoch。

2.3 网络推断（预测）

当训练完成后，可得到最优参数 W^*，此时对于新样本 $\boldsymbol{X}^*$，预测值便是

$$\boldsymbol{y}^* = \boldsymbol{y}\big(\boldsymbol{X}^*, W^*\big).$$

3 反向传播算法（Backpropagation）

以下我们将详细推导网络参数关于损失函数的梯度，核心思想即为**反向传播**。

给定损失函数的一般形式（以二范数方差为例），对一个样本 $\boldsymbol{X}$ 而言，

$$E(W) = \frac{1}{2}\sum_{i_L}\Big(y_{i_L}(\boldsymbol{X}; W) - t_{i_L}\Big)^2.$$

为了便于理解，我们先从二层网络的情况展开推导，再推广到 L-层网络。

3.1 二层网络的梯度推导

设二层网络的输出是

$$y_{i_3}(\boldsymbol{X}; W) = \sum_{i_2=1}^{I_2} W^{(2)}_{i_3 i_2}\, z^{(2)}_{i_2}, \quad z^{(2)}_{i_2} = h\Big(\sum_{i_1=1}^{I_1} W^{(1)}_{i_2 i_1}\, X_{i_1}\Big).$$

损失函数为

$$E(W) = \frac{1}{2}\sum_{i_3}\big(y_{i_3}(\boldsymbol{X}; W) - t_{i_3}\big)^2.$$

关于 $\boldsymbol{W}^{(2)}$ 的梯度

$$\frac{\partial E}{\partial W^{(2)}_{i_3 i_2}} = \frac{\partial E}{\partial y_{i_3}} \cdot \frac{\partial y_{i_3}}{\partial W^{(2)}_{i_3 i_2}} = \Big(y_{i_3}(\boldsymbol{X}; W) - t_{i_3}\Big)\, z^{(2)}_{i_2}.$$

关于 $\boldsymbol{W}^{(1)}$ 的梯度

$$\begin{aligned}\frac{\partial E}{\partial W^{(1)}_{i_2 i_1}} &= \sum_{i_3} \frac{\partial E}{\partial y_{i_3}} \cdot \frac{\partial y_{i_3}}{\partial W^{(1)}_{i_2 i_1}} \\ &= \sum_{i_3}\Big(y_{i_3}(\boldsymbol{X}; W) - t_{i_3}\Big)\, \frac{\partial y_{i_3}}{\partial z^{(2)}_{i_2}}\, \frac{\partial z^{(2)}_{i_2}}{\partial a^{(2)}_{i_2}}\, \frac{\partial a^{(2)}_{i_2}}{\partial W^{(1)}_{i_2 i_1}}.\end{aligned}$$

下面逐一分析：

$$\frac{\partial y_{i_3}}{\partial z^{(2)}_{i_2}} = W^{(2)}_{i_3 i_2},$$

$$\frac{\partial z^{(2)}_{i_2}}{\partial a^{(2)}_{i_2}} = h'\big(a^{(2)}_{i_2}\big),$$

$$\frac{\partial a_{i_2}^{(2)}}{\partial W_{i_2 i_1}^{(1)}} = X_{i_1}.$$

于是得到:

$$\frac{\partial E}{\partial W_{i_2 i_1}^{(1)}} = \sum_{i_3} \Big(y_{i_3}(\boldsymbol{X}; W) - t_{i_3} \Big) W_{i_3 i_2}^{(2)} \, h'\big(a_{i_2}^{(2)}\big) \, X_{i_1}.$$

3.2 L-层网络的梯度推导

对于一般的 L-层网络,

$$\boldsymbol{y}(\boldsymbol{X}; \boldsymbol{W}) : \mathbb{R}^{I_1} \to \mathbb{R}^{I_L},$$

其输出第 i_L 个分量为

$$y_{i_L}(\boldsymbol{X}, \boldsymbol{W}) = \sum_{i_{L-1}} W_{i_L i_{L-1}}^{(L-1)} \Big(h^{(L-2)} \big(\sum_{i_{L-2}} W_{i_{L-1} i_{L-2}}^{(L-2)} \cdots \sum_{i_2} W_{i_3 i_2}^{(2)} \Bigg(h^{(1)} \big(\sum_{i_1} W_{i_2 i_1}^{(1)} X_{i_1} \big) \Bigg) \big) \Big),$$
$$i_L = 1, 2, \cdots, I_L.$$

损失函数

$$E(W) = \frac{1}{2} \sum_{i_L} \Big(y_{i_L}(\boldsymbol{X}; W) - t_{i_L} \Big)^2. \tag{47}$$

同样在训练中我们迭代:

$$W^{t+1} = W^t - \iota \nabla E(W^t).$$

3.2.1 常用记号与前向计算

为进行反向传播,先定义一些中间变量:

$$\begin{cases} a_{i_1}^{(1)} = X_{i_1}, \\ a_{i_l}^{(l)} = \sum_{i_{l-1}} W_{i_l i_{l-1}}^{(l-1)} z_{i_{l-1}}^{(l-1)}, \quad l = 2, 3, \cdots, L, \\ z_{i_l}^{(l)} = h^{(l-1)}\big(a_{i_l}^{(l)}\big), \quad l = 1, 2, \cdots, L-1, \\ h^{(0)}(\cdot) = I(\cdot) \quad (\text{恒等算子}). \end{cases}$$

常见的激活函数包括:

$$\mathrm{Relu}(x) = \max(x, 0),$$
$$\text{以及 Softmax: } [h(\boldsymbol{x})]_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (\text{分类任务时常见}).$$

3.2.2 反向传播的核心思想

记

$$\delta_{i_l}^{(l)} = \frac{\partial E}{\partial a_{i_l}^{(l)}}.$$

对最上层（输出层）有

$$\delta_{i_L}^{(L)} = \frac{\partial E}{\partial a_{i_L}^{(L)}} = \frac{\partial E}{\partial y_{i_L}} \cdot \frac{\partial y_{i_L}}{\partial a_{i_L}^{(L)}}.$$

但对于一般的隐藏层

$$\delta_{i_{l-1}}^{(l-1)} = \sum_{i_l} \frac{\partial E}{\partial a_{i_l}^{(l)}} \cdot \frac{\partial a_{i_l}^{(l)}}{\partial a_{i_{l-1}}^{(l-1)}} = \sum_{i_l} \delta_{i_l}^{(l)} \cdot \frac{\partial a_{i_l}^{(l)}}{\partial z_{i_{l-1}}^{(l-1)}} \cdot \frac{\partial z_{i_{l-1}}^{(l-1)}}{\partial a_{i_{l-1}}^{(l-1)}}.$$

具体计算时，

$$a_{i_l}^{(l)} = \sum_{i_{l-1}} W_{i_l i_{l-1}}^{(l-1)} \, z_{i_{l-1}}^{(l-1)},$$

故

$$\frac{\partial a_{i_l}^{(l)}}{\partial z_{i_{l-1}}^{(l-1)}} = W_{i_l i_{l-1}}^{(l-1)}, \quad \frac{\partial z_{i_{l-1}}^{(l-1)}}{\partial a_{i_{l-1}}^{(l-1)}} = h'^{(l-2)}\big(a_{i_{l-1}}^{(l-1)}\big).$$

因此可以将微分“从后往前”层层传递。

3.2.3 参数梯度公式

对于权重矩阵 $W^{(l-1)}$，元素 $W_{i_l i_{l-1}}^{(l-1)}$ 的梯度：

$$\frac{\partial E}{\partial W_{i_l i_{l-1}}^{(l-1)}} = \frac{\partial E}{\partial a_{i_l}^{(l)}} \cdot \frac{\partial a_{i_l}^{(l)}}{\partial W_{i_l i_{l-1}}^{(l-1)}} = \delta_{i_l}^{(l)} \, z_{i_{l-1}}^{(l-1)}.$$

如果继续往更底层（第 $l-1$ 层、第 $l-2$ 层）传播，就会用到类似的链式法则。

3.3 反向传播的总结

反向传播算法的流程通常分为两个主要阶段：

1. **前向传播（Forward Pass）**：从输入层开始，利用设定好的权重矩阵 W 依次计算 $a^{(l)}, z^{(l)}$ 直到输出层 y。

2. **后向传播（Backward Pass）**：先根据输出层误差 (对应损失函数对输出层的梯度) 计算 $\delta^{(L)}$，再一层层地往回计算 $\delta^{(l)}$，最终得到各层参数 $W^{(l)}$ 的梯度 $\frac{\partial E}{\partial W^{(l)}}$，以便更新。

在实际的神经网络训练中，还要结合批量函数（SGD、mini-batch 等），学习率衰减，正则化，动量方法等多种技巧来提高收敛速度和避免过拟合。但最核心的梯度计算公式就是这里所讲的反向传播链式法则。

4 总结

根据上述内容，我们从最简单的一层网络（线性模型）出发，介绍了二层网络（含激活函数），进而推广到 L-层深度网络的一般形式。然后给出了常见的均方误差损失函数，阐述了神经网络训练的目标与常用的梯度下降（及其变形，如 SGD）方法。最后，我们详细推导了反向传播算法，说明了如何通过链式法则求取网络参数关于损失函数的梯度。

有以下三个重点：

- 神经网络（MLP）的数学表达式，尤其是加权求和和激活函数的结构与符号。
- 损失函数（尤其 MSE）及其关于网络参数的求导形式。
- 反向传播的链式法则公式：如何逐层计算 δ，以及如何得到 $\frac{\partial E}{\partial W^{(l)}}$。