神经网络训练超详细分解手册

由 AI 生成

2025 年 6 月 21 日

目录

1 第零部分：战场设定

下面是一个极其简化的神经网络，我们将以最详尽、最细致的步骤，手动完成一次完整的“小批量”训练，并进行第二次前向传播来验证学习效果。

1.1 1. 神经网络结构

这是一个 2 输入、1 个隐藏层（含 2 个神经元）、1 个输出的简单网络。

- **输入层**: 2 个神经元 (i_1, i_2)
- **隐藏层**: 2 个神经元 (h_1, h_2)
- **输出层**: 1 个神经元 (o_1)
- **激活函数**: 我们统一使用 Sigmoid 函数, $f(x) = \frac{1}{1+e^{-x}}$
- **损失函数**: 使用均方误差 (MSE), $E = \frac{1}{2}(\text{target} - \text{output})^2$

1.2 2. 初始参数 (权重 Weights & 偏置 Biases)

这些都是我们随机初始化的“出厂设置”。

- **隐藏层权重**: $W^{[1]} = \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix} = \begin{pmatrix} 0.15 & 0.20 \\ 0.25 & 0.30 \end{pmatrix}$
- **隐藏层偏置**: $b_1 = 0.35$ （为简化，两个隐藏神经元共享一个偏置）
- **输出层权重**: $W^{[2]} = \begin{pmatrix} w_5 & w_6 \end{pmatrix} = \begin{pmatrix} 0.40 & 0.45 \end{pmatrix}$
- **输出层偏置**: $b_2 = 0.60$

1.3 3. 训练数据 (小批量, Batch Size = 2)

- **样本 1**: 输入 $X_1 = [0.05, 0.10]$, 真实目标 $y_1 = 0.01$
- **样本 2**: 输入 $X_2 = [0.80, 0.20]$, 真实目标 $y_2 = 0.99$

1.4 4. 超参数

- **学习率** η: 0.5

2 第一部分：首次完整训练周期

2.1 第 1 步：完整前向传播 (Forward Propagation)

我们的目标是让数据流过整个网络，看看在当前的参数设置下，它的预测结果有多离谱。

A. 对样本 1: $X_1 = [0.05, 0.10]$

1. **隐藏层计算:**

$$z_{h1} = w_1 \cdot i_1 + w_2 \cdot i_2 + b_1 = (0.15 \cdot 0.05) + (0.20 \cdot 0.10) + 0.35 = 0.3775$$
$$a_{h1} = \text{Sigmoid}(z_{h1}) = \frac{1}{1+e^{-0.3775}} = \mathbf{0.59326} \quad \text{(神经元 h1 的输出)}$$
$$z_{h2} = w_3 \cdot i_1 + w_4 \cdot i_2 + b_1 = (0.25 \cdot 0.05) + (0.30 \cdot 0.10) + 0.35 = 0.3925$$
$$a_{h2} = \text{Sigmoid}(z_{h2}) = \frac{1}{1+e^{-0.3925}} = \mathbf{0.59688} \quad \text{(神经元 h2 的输出)}$$

2. **输出层计算:**

$$z_{o1} = w_5 \cdot a_{h1} + w_6 \cdot a_{h2} + b_2 = (0.40 \cdot 0.59326) + (0.45 \cdot 0.59688) + 0.60 = 1.10590$$
$$\hat{y}_1 = a_{o1} = \text{Sigmoid}(z_{o1}) = \frac{1}{1+e^{-1.10590}} = \mathbf{0.75136} \quad \text{(样本 1 的预测输出)}$$

3. **样本 1 的误差**:

$$E_1 = \frac{1}{2}(y_1 - \hat{y}_1)^2 = \frac{1}{2}(0.01 - 0.75136)^2 = \mathbf{0.27483}$$

B. 对样本 2: $X_2 = [0.80, 0.20]$

（计算过程与样本 *1* 完全相同，此处从略）

- 最终预测输出: $\hat{y}_2 = \mathbf{0.75798}$
- 样本 2 的误差: $E_2 = \mathbf{0.02692}$

2.2 第 2 步：计算总损失 (体现小批量的作用)

目的: 我们不能只根据一个样本的表现就去调整网络，那太片面了。我们需要综合这个批次里所有样本的表现，得出一个“平均表现分”，以此为依据进行调整。

$$E_{total} = \frac{E_1 + E_2}{2} = \frac{0.27483 + 0.02692}{2} = \mathbf{0.150875}$$

3 第二部分：超详细反向传播 (Backward Propagation)

我们的目标是，根据总误差，像侦探一样，从后往前，一环一环地追究每一个参数的“责任”，这个“责任”就是梯度。

3.1 A. 追究输出层参数 (w_5, w_6, b_2) 的责任

3.1.1 第 A.1 步：计算输出神经元 o_1 的“总责任分数” (δ_{o1})

目的: 在追究权重 w_5, w_6 之前，我们先要确定它们连接的下游神经元 o_1 本身，应该为最终的错误负多大的责任。这个责任分数由两部分决定：一是最终错误有多大，二是它自己当时有多“敏感”。

对于样本 1:

1. **计算“最终错误程度”**：$\frac{\partial E_1}{\partial a_{o1}} = a_{o1} - y_1 = 0.75136 - 0.01 = 0.74136$
   含义：预测值偏离真实值有多远。
2. **计算“神经元敏感度”**：$\frac{\partial a_{o1}}{\partial z_{o1}} = a_{o1}(1 - a_{o1}) = 0.75136(1 - 0.75136) = 0.18681$
   含义：神经元当时是否处于容易被影响的“激活区”。
3. **计算“总责任分数”**：$\delta_{o1,\text{样本 1}} = 0.74136 \times 0.18681 = \mathbf{0.13849}$
   含义：综合了错误程度和敏感度的最终责任分数。

对于样本 2:

1. **计算“最终错误程度”**：$a_{o1} - y_2 = 0.75798 - 0.99 = -0.23202$
2. **计算“神经元敏感度”**：$a_{o1}(1 - a_{o1}) = 0.75798(1 - 0.75798) = 0.18345$
3. **计算“总责任分数”**：$\delta_{o1,\text{样本 2}} = -0.23202 \times 0.18345 = \mathbf{-0.04256}$

3.1.2 第 A.2 步：计算权重 w_5 的最终平均梯度

目的: 将神经元 o_1 的责任，根据连接它的上游信号的“嗓门大小”，分摊给权重 w_5。最后再综合两个样本的分析，求出平均责任。

1. **计算样本 1 对 w_5 的梯度**: $\frac{\partial E_1}{\partial w_5} = \delta_{o1,\text{样本 1}} \cdot a_{h1,\text{样本 1}} = 0.13849 \cdot 0.59326 = 0.08217$
   含义：在案件 *1* 中，w_5 的责任 $= o_1$ 的责任分数 $\times$ 当时 h_1 的输出信号强度。
2. **计算样本 2 对 w_5 的梯度**: $\frac{\partial E_2}{\partial w_5} = \delta_{o1,\text{样本 2}} \cdot a_{h1,\text{样本 2}} = -0.04256 \cdot 0.62483 = -0.02659$
   含义：在案件 *2* 中，w_5 的责任。
3. **计算最终平均梯度**: $\frac{\partial E_{total}}{\partial w_5} = \frac{0.08217 + (-0.02659)}{2} = \mathbf{0.02779}$
   含义：综合两起案件，得出 w_5 最终的“判决书”，也就是它需要调整的方向和幅度。

3.1.3 第 A.3 步：计算 w_6 和 b_2 的最终平均梯度

（计算逻辑与 w_5 完全相同）

- w_6 **的最终平均梯度**: $\frac{\partial E_{total}}{\partial w_6} = \mathbf{0.02757}$
- b_2 **的最终平均梯度**: $\frac{\partial E_{total}}{\partial b_2} = \mathbf{0.04797}$

3.2 B. 追究隐藏层参数 $(w_1, w_2, w_3, w_4, b_1)$ 的责任

3.2.1 第 B.1 步：计算隐藏神经元 h_1, h_2 的“总责任分数”(δ_h)

目的: 我们要顺藤摸瓜，把输出层 o_1 的责任，通过连接权重 w_5, w_6，“甩锅”给更上游的隐藏层神经元 h_1, h_2。

计算 h_1 的责任分数 δ_{h1} (对样本 1):

1. **计算“继承来的锅”**：$\delta_{o1,\text{样本 1}} \cdot w_5 = 0.13849 \cdot 0.40 = 0.05540$
   含义：从下游 o_1 沿着权重 w_5 这条路反向传回来的责任有多大。
2. **计算“自身敏感度”**：$a_{h1}(1 - a_{h1}) = 0.59326(1 - 0.59326) = 0.24128$
3. **计算“总责任分数”**：$\delta_{h1,\text{样本 1}} = 0.05540 \times 0.24128 = \mathbf{0.01337}$

（用同样的方法，我们可以计算出所有隐藏神经元在两个样本中的责任分数）

- $\delta_{h1,\text{样本 2}} = \mathbf{-0.00401}$
- $\delta_{h2,\text{样本 1}} = \mathbf{0.01499}$
- $\delta_{h2,\text{样本 2}} = \mathbf{-0.00440}$

3.2.2 第 B.2 步：计算权重 w_1 的最终平均梯度

目的: 现在我们有了隐藏神经元 h_1 的责任分数，就可以用同样的方法，把它分摊给最源头的权重 w_1 了。

1. **计算样本 1 对 w_1 的梯度**: $\frac{\partial E_1}{\partial w_1} = \delta_{h1,\text{样本 1}} \cdot i_1 = 0.01337 \cdot 0.05 = 0.00067$
   含义：在案件 *1* 中，w_1 的责任 $= h_1$ 的责任分数 $\times$ 当时输入信号 i_1 的强度。
2. **计算样本 2 对 w_1 的梯度**: $\frac{\partial E_2}{\partial w_1} = \delta_{h1,\text{样本 2}} \cdot i_1 = -0.00401 \cdot 0.80 = -0.00321$
3. **计算最终平均梯度**: $\frac{\partial E_{total}}{\partial w_1} = \frac{0.00067 + (-0.00321)}{2} = \mathbf{-0.00127}$

3.2.3 第 B.3 步：计算其余隐藏层参数的最终平均梯度

（计算逻辑与 w_1 完全相同）

- $\frac{\partial E_{total}}{\partial w_2} = \mathbf{0.000265}$
- $\frac{\partial E_{total}}{\partial w_3} = \mathbf{-0.00138}$
- $\frac{\partial E_{total}}{\partial w_4} = \mathbf{0.00031}$
- $\frac{\partial E_{total}}{\partial b_1} = \mathbf{0.00498}$

4 第三部分：参数更新与验证

4.1 第 4 步：更新所有参数

目的: 我们已经拿到了所有参数的“判决书”(最终平均梯度)。现在，我们根据这个判决书和学习率（每次调整的力度），来对所有参数进行最终的调整。

$$\text{公式: } W_{new} = W_{old} - \eta \cdot \text{梯度}$$

$$w_{1,new} = 0.15 - 0.5 \cdot (-0.00127) = \mathbf{0.150635}$$

$$w_{2,new} = 0.20 - 0.5 \cdot (0.000265) = \mathbf{0.1998675}$$

$$w_{3,new} = 0.25 - 0.5 \cdot (-0.00138) = \mathbf{0.25069}$$

$$w_{4,new} = 0.30 - 0.5 \cdot (0.00031) = \mathbf{0.299845}$$

$$w_{5,new} = 0.40 - 0.5 \cdot (0.02779) = \mathbf{0.386105}$$

$$w_{6,new} = 0.45 - 0.5 \cdot (0.02757) = \mathbf{0.436215}$$

$$b_{1,new} = 0.35 - 0.5 \cdot (0.00498) = \mathbf{0.34751}$$

$$b_{2,new} = 0.60 - 0.5 \cdot (0.04797) = \mathbf{0.576015}$$

4.2 第 5 步：再次前向传播 (验证学习效果)

目的: 用调整后的新参数，再来处理一下样本 *1*，看看它的“错误分数”是不是降低了。
使用新参数计算:

- 新预测 $\hat{y}_{1,new} = \mathbf{0.74363}$
- 新误差 $E_{1,new} = \frac{1}{2}(0.01 - 0.74363)^2 = \mathbf{0.26909}$

5 最终结论

	训练前	**训练后**
样本 1 的误差	**0.27483**	**0.26909**

表 1: 训练前后误差对比

误差减小了！这清晰地证明了我们的网络通过这一次极其详尽的“责任追究”过程，成功地学习到了如何微调自己，并朝着正确的方向前进了一小步。