3.7 softmax回归的简洁实现重点摘录与练习解答

（1） softmax 函数的改进

在前面的交叉熵计算中，从数学上讲，这是合理的。然而，从数值计算的角度来看，指数可能会造成数值稳定性问题。

根据前文定义，softmax函数 $\hat{y}_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)}$，其中 $\hat{y}_j$ 是预测的概率分布，o_j 是未规范化的预测 $\boldsymbol{o}$ 的第 j 个元素。如果 o_k 中的一些数值非常大，那么 $\exp(o_k)$ 可能大于数据类型容许的最大数字，即上溢（overflow）。这将使分母或分子变为 inf（无穷大），最后得到的是 0、inf 或 nan（不是数字）的 $\hat{y}_j$。在这些情况下，我们无法得到一个明确定义的交叉熵值。

解决这个问题的一个技巧是：在继续 softmax 计算之前，先从所有 o_k 中减去 $\max(o_k)$。这里可以看到每个 o_k 按常数进行的移动不会改变 softmax 的返回值：

$$
\begin{aligned}
\hat{y}_j &= \frac{\exp(o_j - \max(o_k))\exp(\max(o_k))}{\sum_k \exp(o_k - \max(o_k))\exp(\max(o_k))} \\
&= \frac{\exp(o_j - \max(o_k))}{\sum_k \exp(o_k - \max(o_k))}.
\end{aligned}
$$

在减法和规范化步骤之后，可能有些 $o_j - \max(o_k)$ 具有较大的负值。由于精度受限，$\exp(o_j - \max(o_k))$ 将有接近零的值，即下溢（underflow）。这些值可能会四舍五入为零，使 $\hat{y}_j$ 为零，并且使得 $\log(\hat{y}_j)$ 的值为 -inf。反向传播几步后，可能会出现可怕的 nan 结果。

尽管我们要计算指数函数，但我们最终在计算交叉熵损失时会取它们的对数。通过将 softmax 和交叉熵结合在一起，可以避免反向传播过程中可能会困扰我们的数值稳定性问题。如下面的等式所示，我们避免计算 $\exp(o_j - \max(o_k))$，而可以直接使用 $o_j - \max(o_k)$，因为 $\log(\exp(\cdot))$ 被抵消了。

$$
\begin{aligned}
\log(\hat{y}_j) &= \log\left(\frac{\exp(o_j - \max(o_k))}{\sum_k \exp(o_k - \max(o_k))}\right) \\
&= \log\left(\exp(o_j - \max(o_k))\right) - \log\left(\sum_k \exp(o_k - \max(o_k))\right) \\
&= o_j - \max(o_k) - \log\left(\sum_k \exp(o_k - \max(o_k))\right).
\end{aligned}
$$

我们也希望保留传统的 softmax 函数，以备我们需要评估通过模型输出的概率。但是，我们没有将 softmax 概率传递到损失函数中，而是在交叉熵损失函数中传递未规范化的预测，并同时计算 softmax 及其对数。

（**2**）问题解答

2、增加迭代周期的数量。为什么测试精度会在一段时间后降低？我们怎么解决这个问题？

解：因为样本复杂度小于模型复杂度，出现过拟合导致的。

随着迭代周期的数量增加，模型不断去接近样本规律，但到了某一迭代次数后，模型表达能力过剩，会去学习一些只能满足训练样本的非共性特征，即过拟合，从而降低模型的泛化能力，导致测试精度降低。

可以通过以下方法解决：

1）数据增强（Data Augmentation）：通过对训练数据进行各种随机变换（如旋转、平移、缩放、翻转等），扩增训练数据的多样性，可以降低过拟合风险。

2）正则化（Regularization）：通过在损失函数中引入正则化项（如L1正则化、L2正则化），限制模型参数的大小，防止模型过于复杂而出现过拟合。

3）早停（Early Stopping）：在训练过程中监控验证集的性能，当验证集性能不再提升时，停止训练，避免过拟合。

4）Dropout：通过在训练过程中随机将一部分神经元的输出置为0，来减少神经元之间的依赖关系，降低过拟合。

5）模型复杂度调整：减少模型的复杂度，可以通过减少网络层数、减少神经元个数等方式，降低过拟合风险。

6）数据集分割：合理划分训练集、验证集和测试集，用于模型的训练、调参和评估，以确保模型在未知数据上的泛化能力