3.2 线性回归从零开始重点摘录与练习解答

（1）参数更新

我们将执行以下循环：

- 初始化参数
- 重复以下训练，直到完成

  计算梯度

$$\mathbf{g} \leftarrow \partial_{(\mathbf{w},b)} \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} l(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w}, b)$$

  更新参数

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \eta \mathbf{g}$$

（2）问题解答

1、如果我们将权重初始化为零，会发生什么。算法仍然有效吗？

解：在线性回归中，由于只有一层神经网络，且SGD过程中，梯度求导后结果与参数本身无关，而是取决于输入 $\boldsymbol{X}$ 和 y，因此，可以将权重初始化为0，算法仍然有效，在代码部分有实践。

但是，在多层神经网络中，如果将权重初始化为0，或者其他统一的常量，会导致后面迭代的权重更新相同，并且神经网络中的激活单元的值相同，输出的梯度也相等，导致对称性问题，无法进行独立学习，找到最优解。

4、计算二阶导数时可能会遇到什么问题？这些问题可以如何解决？

解：二阶导数包含了更多关于损失函数曲率的信息，因此在某些情况下，计算二阶导数可能有助于更快地收敛和更准确的更新。

以下是计算二阶导数时可能会遇到的问题，以及可能的解决方法：

1. 计算复杂度高：计算Hessian矩阵需要更多计算资源和时间，尤其是大规模数据和复杂模型

   解决方法：通常可以使用近似方法来估计二阶导数，例如L-BFGS（Limited-memory Broyden-Fletcher-Goldfarb-Shanno）等优化算法。这些方法在一定程度上降低了计算成本，同时仍能提供较好的优化效果。

2. 存储需求大：Hessian矩阵存储需求随着参数数量的增加而增加，可能导致内存不足的问题

   解决方法：使用一些高效的矩阵近似方法，如块对角近似（block-diagonal approximation）或采样Hessian近似，来减少存储需求。

3. 数值不稳定性：在计算Hessian矩阵时，可能会遇到数值不稳定，导致数值误差累积，影响优化结果。

解决方法：使用数值稳定的计算方法，例如通过添加小的正则化项来避免矩阵的奇异性。另外，选择合适的优化算法和学习率调度也可以帮助稳定优化过程。

4. 局部极小值和鞍点：在高维空间中，存在许多局部极小值和鞍点，这可能导致Hessian矩阵的谱值较小，使得计算二阶导数的结果不稳定。

   解决方法：使用正则化技术、随机性优化方法（如随机梯度牛顿法）或基于自适应学习率的算法，可以帮助逃离局部极小值和鞍点。

5、为什么在 squared_loss 函数中需要使用 reshape 函数？

解： 因为 y_hat 和 y 的元素个数相同，但 shape 不一定相同（虽然在本节中二者 shape 一致），为了保证计算时不出错，故使用 reshape 函数将二者的 shape 统一。