1.3.1 损失函数

有时候，我们会把一些神经网络的输出值设计为连续范围的值。例如，一个预测温度的网络会输出 0 ～ 100℃的任何值。

也有时候，我们会把网络设计成输出 true/false 或 1/0。例如，我们要判断一幅图像是不是猫，输出值应该尽量接近 0.0 或 1.0，而不是介于两者之间。

如果我们针对不同情况设计损失函数，会发现均方误差只适用于第一种情况。有的读者可能知道这是一个回归（regression）任务，不过不知道也没关系。

对于第二种情况，也就是一个分类（classification）任务，更适合使用其他损失函数。一种常用的损失函数是二元交叉熵损失（binary cross entropy loss），它同时惩罚置信度（confidence）高的错误输出和置信值低的正确输出。PyTorch 将其定义为 nn.BCELoss()。

我们的网络对 MNIST 图像进行分类，属于第二种类型。在理想情况下，输出节点中应该只有一个接近 1.0，其他全部接近 0.0。

复制之前的笔记本，将损失函数从 MSELoss() 更改为 BCELoss()。

```
self.loss_function = nn.BCELoss()
```

让我们像以前一样训练网络 3 个周期。测试数据集的性能得分（准确率）从 87% 提高到了 91%。

再来看看损失图表。