Softmax回归

初始化模型参数

在Softmax回归模型中，每个图像样本都用一个固定长度的向量表示。例如，原始数据集中的每个图像为 28x28 像素，有batch_size个样本，共同构成一个三维的，**经过展平后变为长度为 784 的向量**。数据集共有 10 个类别，因此需要使用一个大小为 **784 x 10** 的权重矩阵 `W`，并初始化为正态分布。同时，偏置项 `b` 设置为一个形状为 **1 x 10** 的零向量。

Softmax操作实现

Softmax函数的作用是将模型的输出转化为概率分布，使得每个类别的输出值都在 [0, 1] 之间，并且所有类别的输出之和为 1。其数学表达式如下：

$$softmax(X)_{ij} = \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})}$$

其中 **X** 是模型的输出，形状为 `(batch_size, num_classes)`，* $softmax(X)_{ij}$ 表示样本 `i` 对应类别 `j` 的预测概率。

定义模型

Softmax回归使用与线性回归类似的方法，通过将样本 **X**（形状为 `(batch_size, 784)`）展平成一个向量，并与权重 **W**（形状为 `784 x 10`）相乘，之后加上偏置 **b**，得到一个 **10** 维的向量，表示每个样本在 10 个类别上的得分。该操作的数学表达式为：

$$\hat{y} = softmax(XW + b)$$

定义损失函数

在Softmax回归中，采用**交叉熵损失函数**来度量真实标签与预测概率之间的差异。交叉熵损失函数的公式为：

$$\text{loss} = -\sum_i y_i \log(\hat{y}_i)$$

其中 **y_i** 是真实标签的类别（0 或 1，表示正确类别），而 $\hat{y}_i$ 是模型输出的预测概率。

定义分类精度

分类精度是指预测类别与真实标签一致时的比例，即：

$$\text{accracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

对于每个样本，选择 **Softmax** 输出中最大概率对应的类别作为模型预测的类别。为了方便计算精度，定义一个 `Accumulator` 类来累积正确预测的数量和总预测数量，在遍历数据集时实时更新这两个值。

训练模型

模型训练使用**随机批量梯度下降（SGD）**方法。通过定义 `updater` 来更新模型参数 **W** 和 **b**，更新规则依据每个批次的梯度计算。训练过程中，定期计算并评估以下指标：训练损失、训练精度和测试精度。在每个训练周期结束时，记录并输出这些指标，确保模型在训练和测试集上都能获得良好的表现。

总结

Softmax回归通过将每个样本的输入映射到一个概率分布，解决了多分类问题。在训练过程中，利用交叉熵损失和SGD优化算法不断调整权重和偏置，并计算分类精度，最终优化模型的性能。通过这种方式，Softmax回归能够有效地进行多类分类任务。

```python
import torch
from IPython import display
from d2l import torch as d2l

# 定义批量大小并加载数据
batch_size = 256
train_iter, test_iter =
d2l.load_data_fashion_mnist(batch_size)

# 定义网络结构
num_inputs = 784
num_outputs = 10
W = torch.normal(0, 0.01, size=(num_inputs,
num_outputs))
W.requires_grad_()  # 注意这里用了下划线 _，表示原地修改
b = torch.zeros(num_outputs, requires_grad=True)

# 软max函数
def softmax(X):
    X_exp = torch.exp(X)
    partition = X_exp.sum(1, keepdim=True)
    return X_exp / partition

# 网络前向传播
def net(X):
    return softmax(torch.matmul(X.reshape((-1,
W.shape[0])), W) + b)

# 交叉熵损失函数
def cross_entropy(y_hat, y):
    return -torch.log(y_hat[range(len(y_hat)), y])


# 定义优化器
lr = 0.1
```

```
def updater(batch_size):
    return d2l.sgd([W, b], lr, batch_size)

# 执行训练
num_epochs = 10
train_ch3(net, train_iter, test_iter, cross_entropy,
num_epochs, updater)

# 预测结果展示
def predict_ch3(net, test_iter, n=6):
    for X, y in test_iter:
        break
    trues = d2l.get_fashion_mnist_labels(y)
    preds =
d2l.get_fashion_mnist_labels(d2l.argmax(net(X),
axis=1))
    titles = [true + '\n' + pred for true, pred in
zip(trues, preds)]
    d2l.show_images(
        d2l.reshape(X[0:n], (n, 28, 28)), 1, n,
titles=titles[0:n])

predict_ch3(net, test_iter)
```