Softmax回归

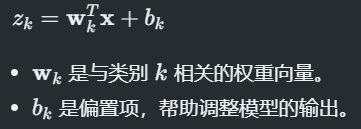
Softmax回归是一种用于多分类问题的机器学习算法。它基于逻辑回归，但与逻辑回归只能处理二分类问题不同，Softmax回归可以处理多分类问题。在Softmax回归中，每个类别都有一个对应的输出，这些输出可以看作是每个类别的“概率”。例如，图像可以被分类为“猫”、“狗”或“鸟”。

具体来说，Softmax回归的输出层是一个全连接层，每个神经元对应一个类别。对于每个输入样本，Softmax回归会计算每个类别的得分（score），然后通过Softmax函数将这些得分转换为概率。

2. 数学原理

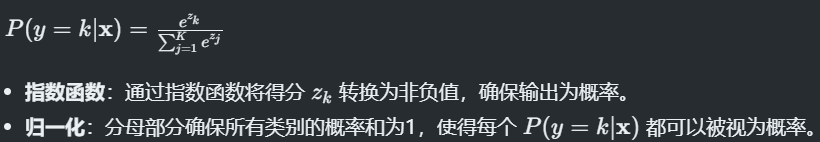
2.1 线性组合

Softmax回归的核心是将输入特征通过线性组合映射到类别的得分。对于每个类别 *k*，我们有：



2.2 Softmax函数

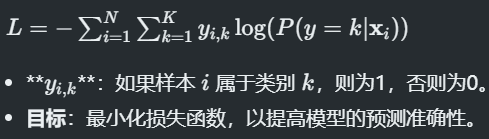
Softmax函数的形式是：



3. 损失函数

3.1 交叉熵损失

交叉熵损失函数用于评估模型的预测与真实标签之间的差异。其形式为：



4. 优化方法

4.1 梯度下降

为了优化模型参数（权重和偏置），通常使用梯度下降算法。步骤如下：

1. **计算梯度**：对损失函数进行求导，得到每个权重的梯度。
2. **更新参数**：使用学习率 η 更新每个权重和偏置：

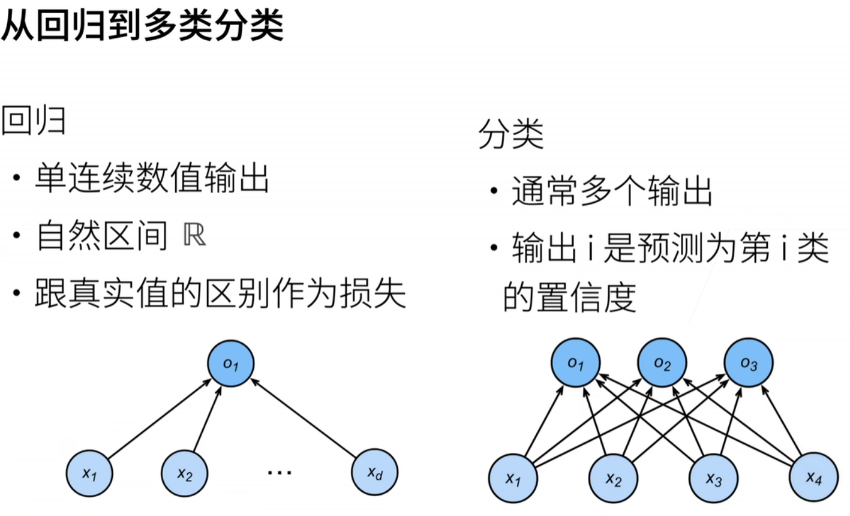


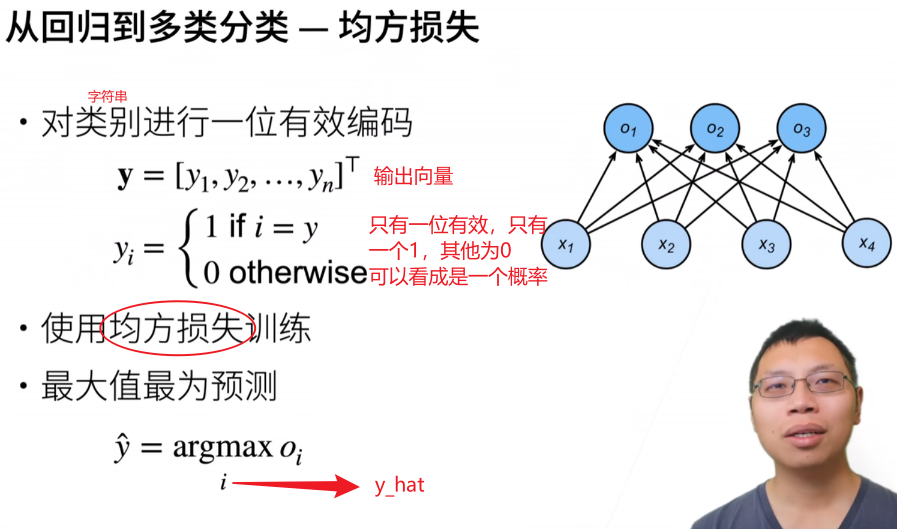
5. 实现细节

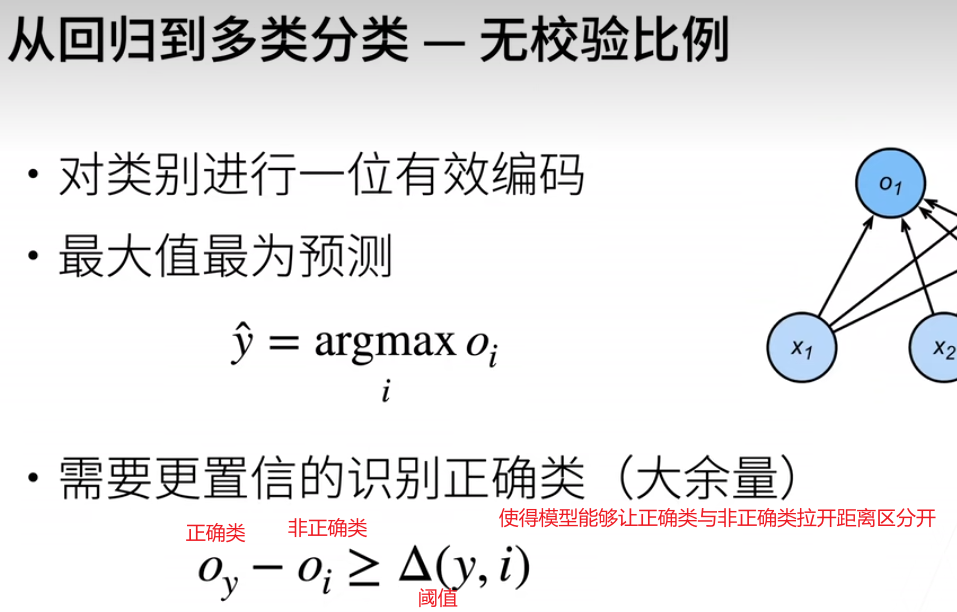
特征预处理：在使用Softmax回归之前，通常需要对输入特征进行预处理，如标准化或归一化，以提高模型的收敛速度和性能。

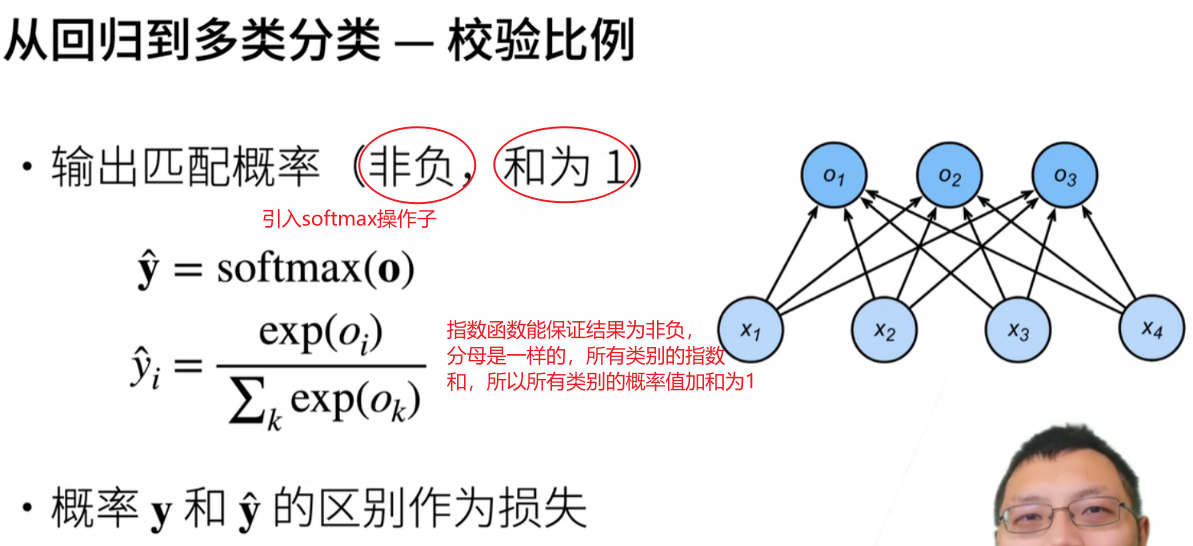
1. Softmax回归–分类问题

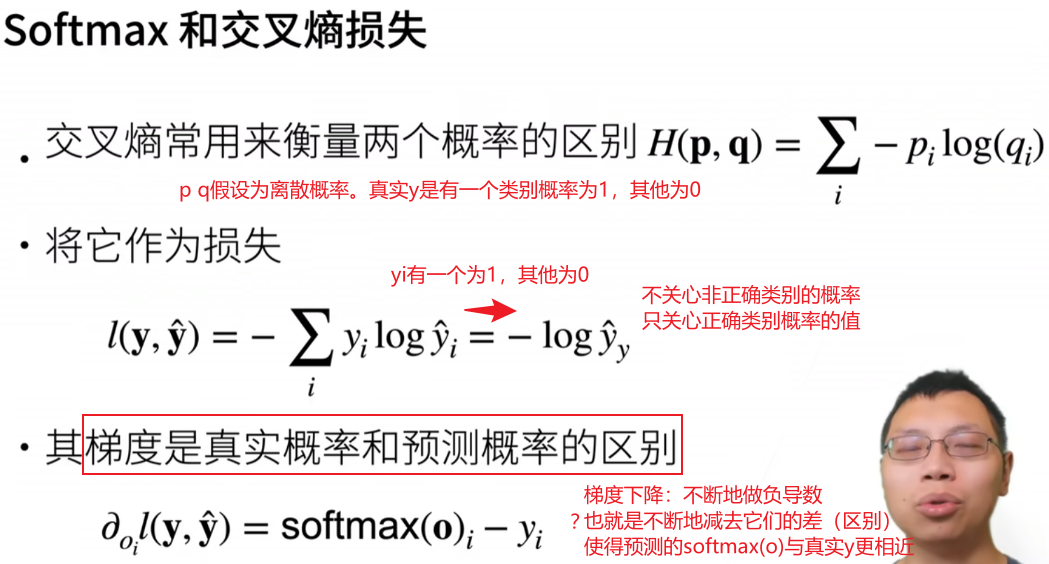


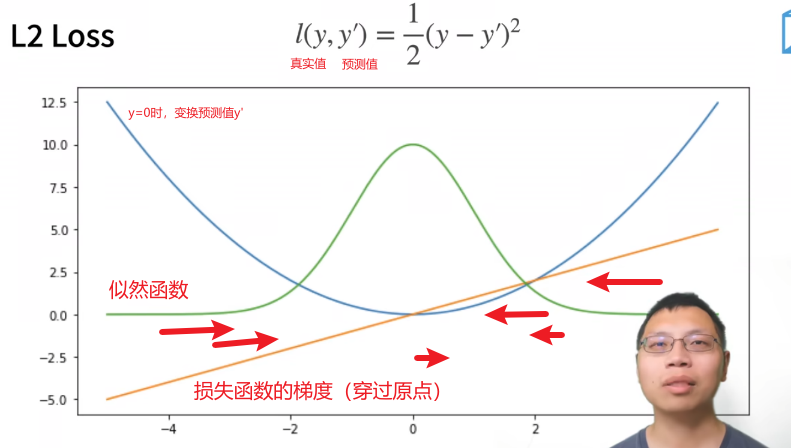








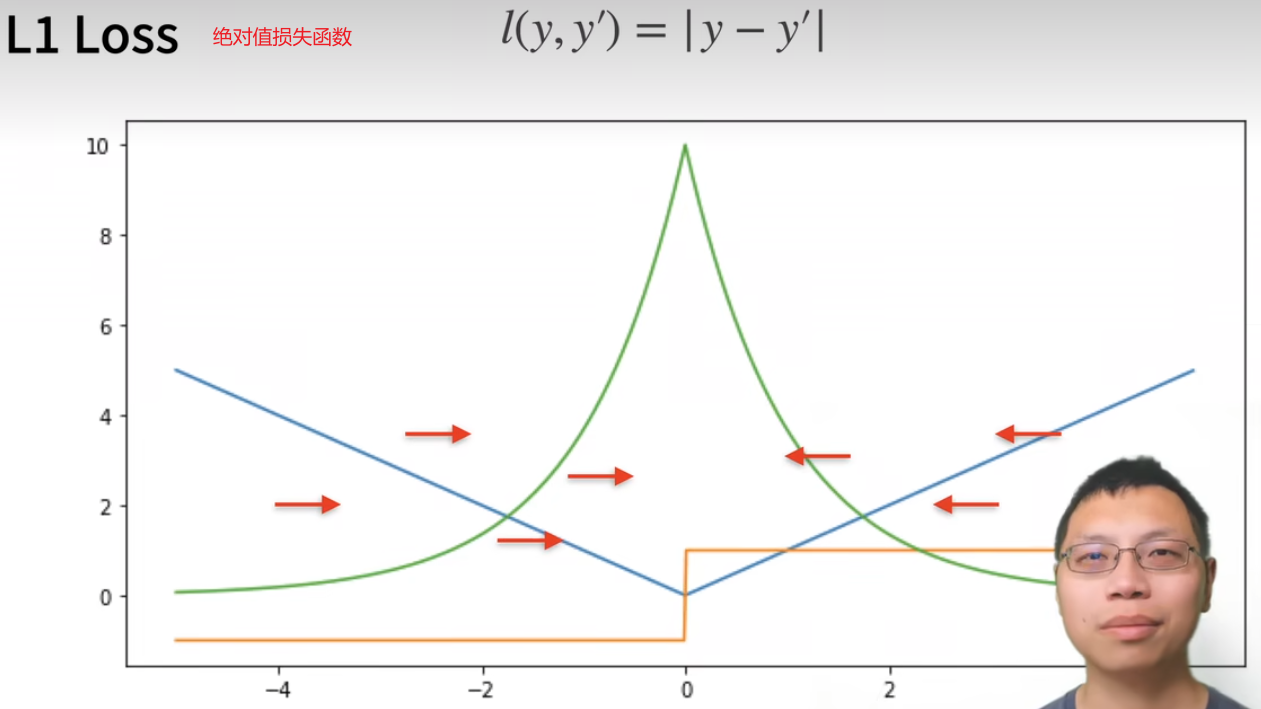




**均方损失函数**：梯度下降时，根据负梯度的方向更新参数，导数决定如何更新参数的。当预测值和真实值相差比较大，梯度比较大，参数变化比较大、多；当预测值和真实值相差比较小（靠近原点）梯度越来越小，参数变化的幅度也越来越小。

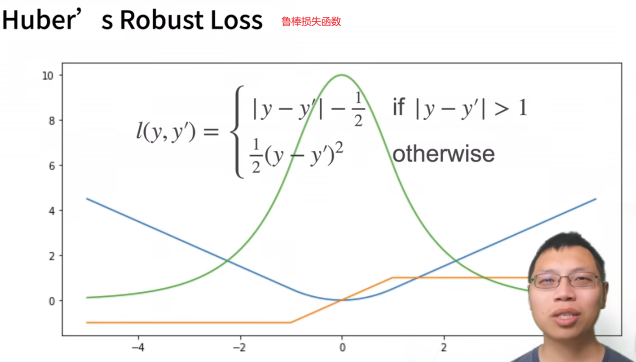
**似然函数**是一种关于统计模型参数的函数。给定输出x时，关于参数θ的似然函数L(θ|x)（在数值上）等于给定参数θ后变量X的概率：L(θ|x)=P(X=x|θ)。

当离原点比较远的时候，不一定想要很大的梯度来更新参数，可以考虑绝对值损失函数。



在大于0，导数为常数1，小于0，导数-1，在0点不可导，导数在（-1,1）之间瞬间变化

当预测值和真实值离得比较远的时候，梯度为常数，权重更新不大，会带来很多稳定性的好处【不管多远，梯度以同样的力度向中间扯】。不好的地方，0点处不可导，从-1到+1的剧烈变化，平滑性差，当预测值和真实值靠的近，优化到了末期，这里可能变得不那么稳定。



**两个结合 HUber’s Robust Loss鲁棒损失**

定义：当预测值和真实值差距较大，绝对值大于1的时候，损失函数是绝对值误差，减去1/2是为了和曲线连接起来。

当预测值和真实值差距较小，绝对值小于等于1的时候，是平方误差。

蓝色曲线：损失函数，在正负一之间是平滑的二次函数，之外是直线

绿色曲线：似然函数，很像高斯分布

橙色曲线：在正负一之间，是渐变的；其他是常数。

当预测值和真实值差距较大，梯度用均匀的力度往回拉。

当预测值和真实值差距较小，到了优化末期，梯度会越来越小，保证优化是平滑的，避免出现数值的问题。

总结：分析损失函数特性是通过函数形状，梯度形状，分析预测值和真实值差别大 差别小的时候的函数特性。

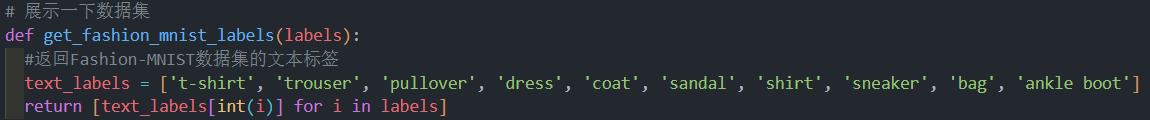
2.图片分类数据集

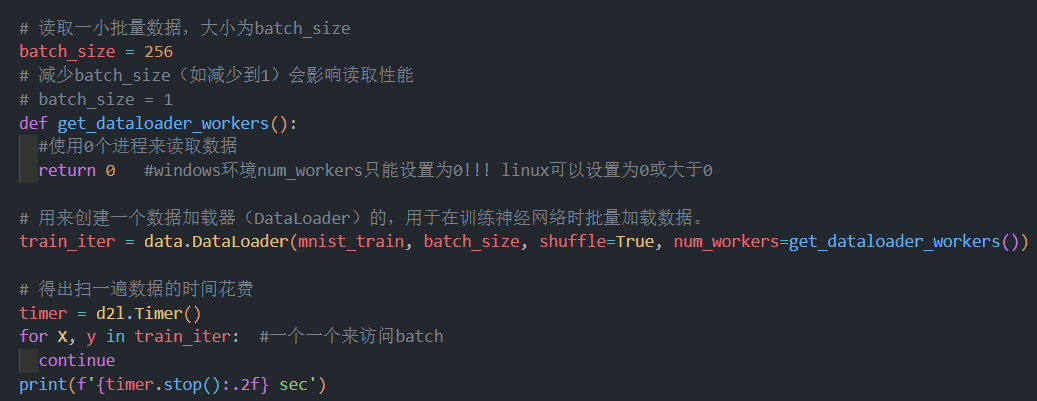


torchvision.datasets：用于加载常见数据集的库，如FashionMNIST

transform=trans：确保数据集中的每个样本都是经过ToTensor变换后的torch.Tensor对象，而不是原始的图像文件。这对于后续的深度学习模型训练和评估非常重要，因为模型需要处理的是张量数据，而不是图像文件。

PIL图像的像素值范围通常是0到255，而深度学习模型通常需要归一化的输入



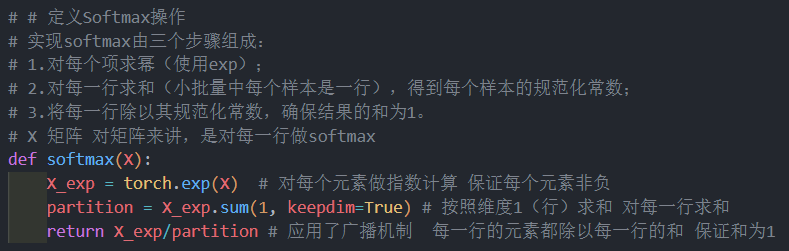


常见性能问题：模型训练速度快，但是数据读取速度慢，读不过来





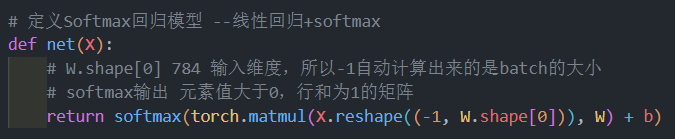
Size：矩阵大小 requires\_grad = True：需要计算梯度



**X**为一个矩阵，对其按每一行做softmax

**keepdim=True**是 PyTorch 中一个常用的参数，主要用于在张量操作（如求和、平均值等）后保持张量的维度，这里是为了使后面广播能够正确进行

**归一化**：**return X\_exp / partition** 将 X\_exp 中的每个元素除以对应的行总和 partition。这一步是为了将每个元素的值归一化到0和1之间，并且保证每一行的总和为1。这样，**X\_exp / partition** 的结果就是一个概率分布，其中每个元素表示对应类别的概率。



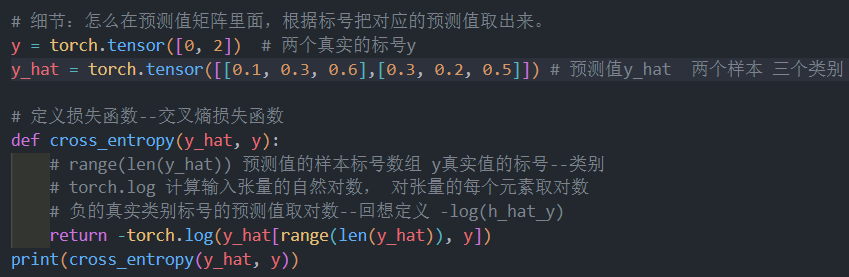
X是输入数据，形状为(batch\_size, input\_dim)。

W是权重矩阵，形状为(input\_dim, output\_dim)。

b是偏置向量，形状为(output\_dim,)。

torch.matmul(X.reshape((-1, W.shape[0])), W)计算输入X与权重W的点积，结果形状为(batch\_size, output\_dim)。

softmax函数将点积结果转换为概率分布，输出形状为(batch\_size, output\_dim)。



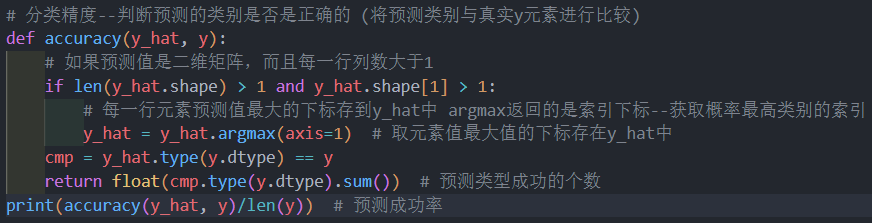
**y**是一个包含两个真实类别标号的张量。

**y\_hat**是一个包含两个样本的预测值矩阵，每个样本有3个类别。

y\_hat[range(len(y\_hat)), y] 使用高级索引从 y\_hat 中提取对应于每个样本真实类别标号的预测值。在深度学习中，**高级索引**主要用于处理张量（tensor）数据，以便于高效的数据操作和模型训练 （range：0—len-1）

Softmax本身不是损失函数，而是一种激活函数，用于将模型的输出转换为概率分布。它通常与交叉熵损失函数结合使用，以处理多分类问题。

在训练多分类模型时，通常先通过Softmax将输出转换为概率，然后使用交叉熵损失来评估模型的性能。因此，Softmax与交叉熵损失函数是常常一起使用的，但它们的功能不同。



每一行元素预测值最大的下标存到y\_hat中，每一行求argmax，argmax返回的是索引下标--获取概率最高类别的索引

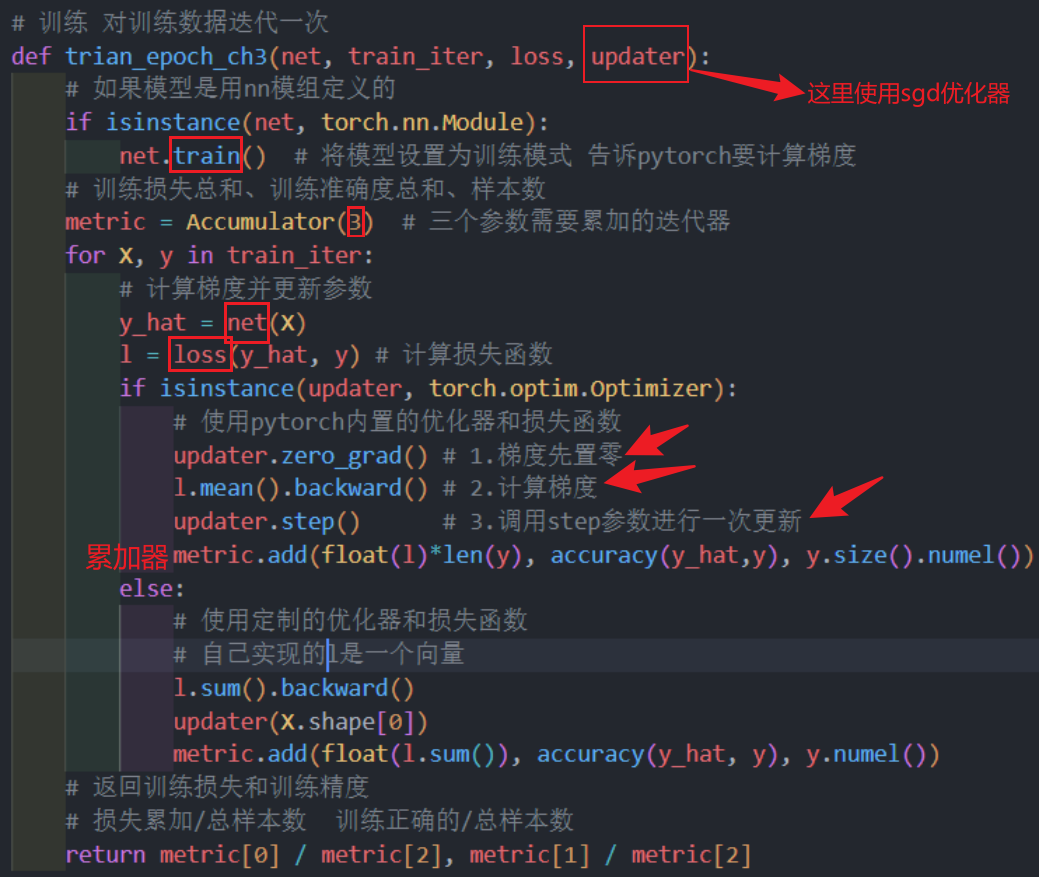
cmp将 y\_hat 的数据类型转换为 y 的数据类型，然后与 y 进行逐元素比较，得到一个布尔张量 cmp，其中 True 表示预测正确，False 表示预测错误。

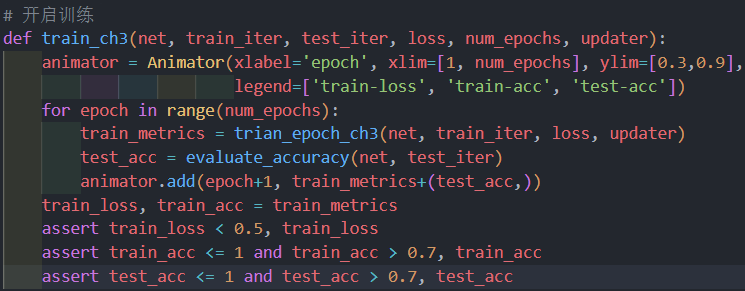
将布尔数组 cmp 转换为 y 的数据类型，并计算 True 的数量，即预测正确的样本数，将这个数量转换为浮点数并返回，然后除以整个真实y的长度，就得出概率。



**Isinstance：**将模型设置为评估模式，不需要计算梯度了，只需要执行forward。

**Accumulator：**累加器，Accumulator 类可以用于在训练机器学习模型时跟踪多个变量的累加值，例如损失函数、梯度等。而且它可以简化代码，避免手动管理多个累加变量

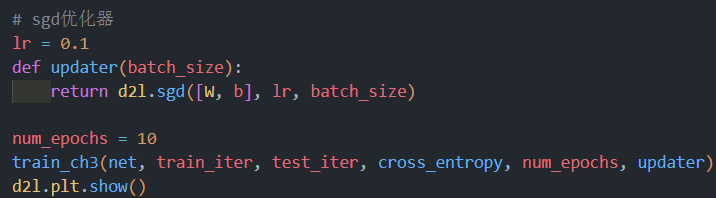




**train\_ch3：net：**神经网络模型。**train\_iter：**训练数据迭代器。

**test\_iter：**测试数据迭代器。**loss：**损失函数。**num\_epochs：**训练的次数。

**updater：**更新模型参数的优化器。这里使用sgd优化器，如下



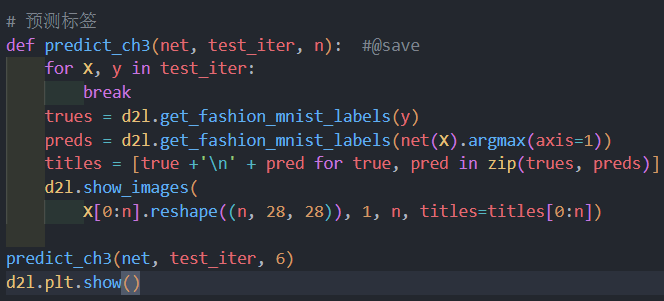
**Animator：**一个用于绘制训练过程的动画类。

* xlabel：x轴标签，这里设置为'epoch'，表示训练次数。
* xlim：x轴范围，从1到num\_epochs\*（10）。
* ylim：y轴范围，从0.3到0.9。
* legend：图例，包括训练损失、训练准确率和测试准确率。

**trian\_epoch\_ch3(net, train\_iter, loss, updater)：**调用 trian\_epoch\_ch3 函数进行一轮训练，返回训练损失和训练准确率，并更新模型

**evaluate\_accuracy(net, test\_iter)：**调用 evaluate\_accuracy 函数在测试数据集上评估精度

**animator.add(epoch+1, train\_metrics+(test\_acc,))：**将训练误差、训练准确率和测试误差、测试准确率添加到动画中



遍历测试数据集

* trues：真实标签，通过d2l.get\_fashion\_mnist\_labels函数将标签转换为对应的服装类别名称。
* preds：预测标签，通过net(X).argmax(axis=1)获取模型对每个样本的预测结果，然后通过d2l.get\_fashion\_mnist\_labels将预测结果转换为对应的服装类别名称。
* titles：每个样本的真实标签和预测标签组合成的标题列表。（显示）

