Softmax regression

Softmax回归虽然它的名字是回归，其实它是一个分类问题。

Difference

Regression

估计一个连续值，只有一个类别输出，输出值和真实值的区别作为损失

Classification

预测一个离散类别，通常多个输出，

Tradiational Softmax Code

```
mnist_train = torchvision.datasets.FashionMNIST(root='D:\\Deeplearning_Li
Mu_Date\\FashionMNIST', train=True, transform=transforms.ToTensor(), download=True)
mnist_test = torchvision.datasets.FashionMNIST(root='D:\\Deeplearning_Li
Mu_Date\\FashionMNIST', train=False, transform=transforms.ToTensor(), download=True)
print('mnist_train_len:',len(mnist_train), 'mnist_test_len:',len(mnist_test))
print('mnist_train_shape:',mnist_train[0][0].shape) # 黑白图片，所以channel为1。

def get_fashion_mnist_labels(labels): # 将数字标签转换为对应的文字标签，方便理解
    text_labels = ['t-shirt', 'trouser', 'pullover', 'dress', 'coat',
                   'sandal','shirt','sneaker', 'bag', 'ankle boot']
    return [text_labels[int(i)] for i in labels]

def show_images(imgs, num_rows, num_cols, titles=None, scale=1.5):
    figsize = (num_cols * scale, num_rows * scale) # 传进来的图像尺寸，scale为放缩比例
因子
    _, axes = d2l.plt.subplots(num_rows,num_cols,figsize=figsize)
    print(_)
    print(axes) # axes 为构建的两行九列的画布
    axes = axes.flatten()
    print(axes) # axes 变成一维数据
    for i,(ax,img) in enumerate(zip(axes,imgs)):
        if(i<1):
            print("i:",i)
            print("ax,img:",ax,img)
        if torch.is_tensor(img):
            # 图片张量
            ax.imshow(img.numpy())
            ax.set_title(titles[i])
        else:
            # PIL图片
            ax.imshow(img)

X, y = next(iter(data.DataLoader(mnist_train,batch_size=18))) # X，y 为仅抽取一次的18
个样本的图片、以及对应的标签值
show_images(X.reshape(18,28,28),2,9,titles=get_fashion_mnist_labels(y))
d2l.plt.show()

batch_size = 256
```

```
def get_dataloader(train_dataset, test_dataset, batch_size):
    return 10
train_iter = data.DataLoader(mnist_train, batch_size=batch_size, shuffle=True)

timer = d2l.Timer()
for X, y in train_iter:
    continue
print(f'{timer.stop():.2f} sec')

def load_data_fashion_mnist(batch_size, resize=None):
    trans = [transforms.ToTensor()]
    if resize:
        trans.insert(0,transforms.Resize(resize)) # 如果有Resize参数传进来，就进行
resize操作
    trans = transforms.Compose(trans)
    mnist_train = torchvision.datasets.FashionMNIST(root='D:\\Deeplearning_Li
Mu_Date\\FashionMNIST',train=True,transform=trans,download=True)
    mnist_test = torchvision.datasets.FashionMNIST(root='D:\\Deeplearning_Li
Mu_Date\\FashionMNIST',train=False,transform=trans,download=True)
    return (data.DataLoader(mnist_train, batch_size, shuffle=True,
num_workers=get_dataloader_workers()),
            data.DataLoader(mnist_train, batch_size, shuffle=True,
num_workers=get_dataloader_workers()))

batch_size = 256
train_iter, test_iter = d2l.load_data_fashion_mnist(batch_size)
num_inputs = 784
num_outputs = 10

w = torch.normal(0, 0.01, size=(num_inputs, num_outputs), requires_grad=True)
b = torch.zeros(num_outputs, requires_grad=True)

X = torch.tensor([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
print(X.sum(0, keepdim=True)) # 沿着第 0 维（行）求和，意味着对每一列的所有元素求和。
print(X.sum(1, keepdim=True)) # 沿着第 1 维（列）求和，意味着对每一行的所有元素求和。

def softmax(X):
    x_exp = torch.exp(X)
    partition = x_exp.sum(1, keepdim=True)
    return x_exp / partition # 这里的除法是逐元素的除法，即除以每一行的和

X = torch.normal(0, 1, size=(2, 5))
print(X)
x_prob = softmax(X)
print(x_prob)
print(x_prob.sum(1)) # 这两行代码的作用是一样的，都是对每一行的元素求和，但是第二种写法更
加简洁。

def net(X):
    return softmax(torch.matmul(X.reshape((-1, w.shape)), w) + b)

y = torch.tensor([0, 2])
y_hat = torch.tensor([[0.1, 0.3, 0.6], [0.3, 0.2, 0.5]])
print(y)
print(y_hat)
print(y_hat[[0,1],y]) # 取出第 0、1 行，第 y 列的元素。即第0行第0列和第1行第2列的元素。

def cross_entropy(y_hat, y):
```

```
    return -torch.log(y_hat[range(len(y_hat)), y])

print('cross_entropy:',cross_entropy(y_hat, y))

def accuracy(y_hat,y):
    if len(y_hat.shape) > 1 and y_hat.shape[1] > 1: # y_hat.shape[1]>1表示不止一个类别，每个类别有各自的概率
        y_hat = y_hat.argmax(axis=1) # y_hat.argmax(axis=1)为求行最大值的索引
    cmp = y_hat.type(y.dtype) == y # 先判断逻辑运算符==，再赋值给cmp，cmp为布尔类型的数据
    return float(cmp.type(y.dtype).sum()) # 获得y.dtype的类型作为传入参数，将cm

print('accuracy:',accuracy(y_hat, y))


class Accumulator:
    def __init__(self,n):
        self.data = [0,0] * n

    def add(self, *args):
        self.data = [a+float(b) for a,b in zip(self.data,args)] # zip函数把两个列表第一个位置元素打包、第二个位置元素打包....

    def reset(self):
        self.data = [0.0] * len(self.data)

    def __getitem__(self,idx):
        return self.data[idx]

def evaluate_accuracy(net, data_iter):
    if isinstance(net, torch.nn.Module):
        net.eval() # 评估模式，关闭dropout
    metric = Accumulator(2) # 正确预测数、预测总数
    for X, y in data_iter:
        metric.add(accuracy(net(X), y), y.numel()) # 将预测的准确率和样本总数相加
    return metric[0] / metric[1] # 返回正确预测数/预测总数

print(evaluate_accuracy(net, test_iter))
```

Pytorch Softmax Code

```
import torch
import torchvision
from torch import nn
from d2l import torch as d2l

batch_size = 256
train_iter, test_iter = d2l.load_data_fashion_mnist(batch_size)
# PyTorch不会隐式地调整输入的形状
# 因此，我们定义了展平层(flatten)在线性层前调整网络输入的形状
net = nn.Sequential(nn.Flatten(), nn.Linear(784, 10))
print(net)
batch_size = 256
# 判断是否是线形层，是则正态分布
def init_weights(m):
    if type(m) == nn.Linear:
        nn.init.normal_(m.weight, std=0.01)
```

```
def train_ch3(net, train_iter, test_iter, loss, num_epochs, updater):
    """Train a model (defined in Chapter 3).

    Defined in :numref:`sec_softmax_scratch`"""
    animator = Animator(xlabel='epoch', xlim=[1, num_epochs], ylim=[0.3, 0.9],
                        legend=['train loss', 'train acc', 'test acc'])
    for epoch in range(num_epochs):
        train_metrics = train_epoch_ch3(net, train_iter, loss, updater)
        test_acc = evaluate_accuracy(net, test_iter)
        animator.add(epoch + 1, train_metrics + (test_acc,))
    train_loss, train_acc = train_metrics
    assert train_loss < 0.5, train_loss
    assert train_acc <= 1 and train_acc > 0.7, train_acc
    assert test_acc <= 1 and test_acc > 0.7, test_acc

net.apply(init_weights)
print(net)
loss = nn.CrossEntropyLoss()
trainer = torch.optim.SGD(net.parameters(), lr=0.1) # 随机梯度
num_epochs = 10
#李沐老师为此课程创建的库，已在上文展示
d2l.train_ch3(net, train_iter, test_iter, loss, num_epochs, trainer)
d2l.plt.show()
```