Softmax Regression in Gluon

In [1]:
```
%matplotlib inline
import d2l
from mxnet import gluon, init
from mxnet.gluon import nn
```

In [2]:
```
batch_size = 256
train_iter, test_iter = d2l.load_data_fashion_mnist(batch_size)
```

Initialize Model Parameters

The output layer of softmax regression is a fully connected layer. We initialize the weights at random with zero mean and standard deviation 0.01.

In [3]:
```
net = nn.Sequential()
net.add(nn.Dense(10))
net.initialize(init.Normal(sigma=0.01))
```

The Softmax

$$\log\left(\hat{y}_j\right) = \log\left(\frac{e^{z_j}}{\sum_{i=1}^{n} e^{z_i}}\right) = z_j - \log\left(\sum_{i=1}^{n} e^{z_i}\right)$$

We make our life easier by using the log of the softmax as follows.

```
In [4]: loss = gluon.loss.SoftmaxCrossEntropyLoss()
```

Optimization Algorithm

We use SGD with a learning rate of 0.1, just as in linear regression.

In [5]:
```
trainer = gluon.Trainer(net.collect_params(), 'sgd', {'learning_rate': 0.1})
```

Training

Next, we use the training functions defined in the last section to train a model.

```
In [6]: num_epochs = 5
        d2l.train_ch3(net, train_iter, test_iter, loss, num_epochs, batch_size, None,
                      None, trainer)
```

```
epoch 1, loss 0.7891, train acc 0.746, test acc 0.805
epoch 2, loss 0.5731, train acc 0.811, test acc 0.826
epoch 3, loss 0.5297, train acc 0.824, test acc 0.824
epoch 4, loss 0.5047, train acc 0.830, test acc 0.832
epoch 5, loss 0.4896, train acc 0.834, test acc 0.843
```

```
In [ ]:
```