In [ ]:
```
%load_ext autoreload
%autoreload 2

import numpy as np
from torch import nn
import torch
from d2l import torch as d2l

import mytorch
from mytorch import nn as mynn
```

In [ ]:
```
# load data for testing
data = np.load('testerData.npz')
W, b, X, Y, dLdZ = [data[fname] for fname in data.files]

[N, num_inputs] = X.shape
num_outputs = Y.shape[1]

# converted torch versions
Xt = torch.tensor(X).float()
Wt = torch.tensor(W).float()
bt = torch.tensor(b).float()
Yt = torch.tensor(Y).float()
```

In [ ]:
```
# initialize model and fix weights to true values
my_net = mynn.Linear(num_inputs, num_outputs)
my_net.W = W
my_net.b = b.flatten()

# initialize torch model, loss, optimizer
net = nn.Linear(num_inputs, num_outputs)
net.weight = nn.Parameter(Wt.T)
net.bias = nn.Parameter(bt[:, 0])
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(net.parameters(), lr=0.1, momentum=0.0)
```

In [ ]:
```
# get gradients for both networks
my_out = my_net.forward(X)
my_net.backward(dLdZ)
my_dLdW = my_net.dLdW
my_dLdb = my_net.dLdb

torch_out = net(Xt)
optimizer.zero_grad()
torch_loss_fn = nn.MSELoss()
torch_loss = torch_loss_fn(torch_out, Yt)
torch_loss.backward(retain_graph=True)
```

Compare a single optimization step

In [ ]:
```
# my SGD step
my_optimizer = mytorch.optim.SGD(my_net, lr=0.1)
```

```
my_optimizer.step()
my_Wk = my_net.W
my_bk = my_net.b

# torch SGD step
optimizer.zero_grad()
torch_loss.backward(retain_graph=True)
optimizer.step()
torch_Wk = net.weight.data
torch_bk = net.bias.data

print('MyTorch Wk:\n', my_Wk, '\n')
print('PyTorch Wk:\n', torch_Wk.T, '\n')
print('MyTorch bk:\n', my_bk, '\n')
print('PyTorch bk:\n', torch_bk, '\n')

print('Difference in Wk:', np.linalg.norm(my_Wk - torch_Wk.data.numpy().T))
print('Difference in bk:', np.linalg.norm(my_bk.flatten() - torch_bk.data.numpy()))
```

```
MyTorch Wk:
 [[0.00225624 0.65454369 0.22786553 0.20375799 0.41525752]
 [0.06995093 0.09965917 0.54460619 0.32926168 0.65353759]
 [0.69258879 0.6152178  0.48305614 0.45607477 0.16900205]
 [0.05864521 0.35032098 0.24489474 0.8639804  0.83109932]
 [0.7373715  0.07190249 0.15495887 0.52788239 0.4534458 ]
 [0.14922424 0.40519783 0.92280482 0.5548514  0.28792973]
 [0.09215936 0.45045506 0.78606601 0.93621087 0.64919466]
 [0.28578957 0.23344379 0.51530289 0.79663504 0.54019491]
 [0.22328781 0.61469977 0.93714608 0.01456096 0.10773379]
 [0.06160243 0.89891648 0.83156413 0.43773651 0.74866049]]

PyTorch Wk:
 tensor([[0.0023, 0.6545, 0.2279, 0.2038, 0.4153],
        [0.0700, 0.0997, 0.5446, 0.3293, 0.6535],
        [0.6926, 0.6152, 0.4831, 0.4561, 0.1690],
        [0.0586, 0.3503, 0.2449, 0.8640, 0.8311],
        [0.7374, 0.0719, 0.1550, 0.5279, 0.4534],
        [0.1492, 0.4052, 0.9228, 0.5549, 0.2879],
        [0.0922, 0.4505, 0.7861, 0.9362, 0.6492],
        [0.2858, 0.2334, 0.5153, 0.7966, 0.5402],
        [0.2233, 0.6147, 0.9371, 0.0146, 0.1077],
        [0.0616, 0.8989, 0.8316, 0.4377, 0.7487]])

MyTorch bk:
 [[0.61248868 0.96033306 0.09868591 0.80567634 0.61423769]]

PyTorch bk:
 tensor([0.6125, 0.9603, 0.0987, 0.8057, 0.6142])

Difference in Wk: 1.037580056856067e-07
Difference in bk: 2.613949504884645e-08
```