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

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

import mytorch
from mytorch import nn as mynn
from models import MLP0, MLP1
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [ ]:
```
# set up synthetic data
N = 10
num_inputs = 7
num_outputs = 2

# numpy/our versions
W = np.random.rand(num_inputs, num_outputs)
b = np.random.rand(num_outputs)
X = np.random.randn(N, num_inputs)
Y = X @ W + np.outer(np.ones(N), b) + 0.5 * np.random.randn(N, num_outputs)

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

MLP0

Test `forward()`

In [ ]:
```
# initialize model and fix weights to true values
mlp0 = MLP0(num_inputs, num_outputs)
mlp0.layers[0].W = W
mlp0.layers[0].b = b

# initialize torch model, loss, optimizer
net = nn.Sequential(nn.Linear(num_inputs, num_outputs))
net[0].weight = nn.Parameter(Wt.T)
net[0].bias = nn.Parameter(bt)
optimizer = torch.optim.SGD(net.parameters(), lr=1, momentum=0.0)

my_out = mlp0.forward(X)
torch_out = net(Xt)

print('MyTorch:\n', my_out, '\n')
```

```
print('PyTorch:\n', torch_out.data, '\n')
print('Difference:', np.linalg.norm(my_out - torch_out.data.numpy()))
```

```
MyTorch:
 [[ 3.86002782  2.99438936]
 [-0.79696783 -0.2794177 ]
 [-0.16700786  0.08508616]
 [-1.68222523 -0.6309244 ]
 [-0.82226997 -0.4766995 ]
 [ 3.64463101  1.33491015]
 [-2.59587695 -0.72561625]
 [-1.96711607 -1.04104274]
 [-1.24519469 -1.12006223]
 [ 0.83728668  1.70837926]]

PyTorch:
 tensor([[ 3.8600,  2.9944],
        [-0.7970, -0.2794],
        [-0.1670,  0.0851],
        [-1.6822, -0.6309],
        [-0.8223, -0.4767],
        [ 3.6446,  1.3349],
        [-2.5959, -0.7256],
        [-1.9671, -1.0410],
        [-1.2452, -1.1201],
        [ 0.8373,  1.7084]])

Difference: 3.7775427099892897e-07
```

Test `backward()`

In [ ]:
```
my_mse_fn = mynn.MSELoss()
my_mse = my_mse_fn.forward(my_out, Y)
dLdZ = my_mse_fn.backward()
mlp0.backward(dLdZ)
my_dLdW = mlp0.layers[0].dLdW
my_dLdb = mlp0.layers[0].dLdb

optimizer.zero_grad()
torch_loss_fn = nn.MSELoss()
torch_loss = torch_loss_fn(torch_out, Yt)
torch_loss.backward(retain_graph=True)
torch_dLdW = net[0].weight.grad.data
torch_dLdb = net[0].bias.grad.data

print('MyTorch dLdW:\n', my_dLdW, '\n')
print('PyTorch dLdW:\n', torch_dLdW.T, '\n')
print('MyTorch dLdb:\n', my_dLdb, '\n')
print('PyTorch dLdb:\n', torch_dLdb, '\n')

print('Difference in dLdW:', np.linalg.norm(my_dLdW.T - torch_dLdW.numpy()))
print('Difference in dLdb:', np.linalg.norm(my_dLdb.flatten() - torch_dLdb.n
```

```
MyTorch dLdW:
 [[-0.21218814  0.05111207]
 [-0.08868613  0.02084339]
 [ 0.078828    0.17021322]
 [ 0.06374441 -0.16389866]
 [ 0.17499219 -0.0323467 ]
 [ 0.21739307 -0.07383894]
 [-0.08884962 -0.10607492]]

PyTorch dLdW:
 tensor([[-0.2122,  0.0511],
        [-0.0887,  0.0208],
        [ 0.0788,  0.1702],
        [ 0.0637, -0.1639],
        [ 0.1750, -0.0323],
        [ 0.2174, -0.0738],
        [-0.0888, -0.1061]])

MyTorch dLdb:
 [-0.34788953  0.21548112]

PyTorch dLdb:
 tensor([-0.3479,  0.2155])

Difference in dLdW: 1.5759888413168285e-07
Difference in dLdb: 1.6025198587496917e-08
```

Test a single optimization step

In [ ]:

```
# my SGD step
my_optimizer = mytorch.optim.SGD(mlp0, lr=1)
my_optimizer.step()
my_Wk = mlp0.layers[0].W
my_bk = mlp0.layers[0].b

# torch SGD step
optimizer.zero_grad()
torch_loss.backward(retain_graph=True)
optimizer.step()
torch_Wk = net[0].weight.data
torch_bk = net[0].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)

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

```
MyTorch Wk:
 [[0.64096616 0.2778433 ]
 [0.61522442 0.40386958]
 [0.29952571 0.16312084]
 [0.88400252 0.59914513]
 [0.28828544 0.71377332]
 [0.58806773 0.79375926]
 [0.69368298 0.18085436]]

PyTorch Wk:
 tensor([[0.6410, 0.2778],
        [0.6152, 0.4039],
        [0.2995, 0.1631],
        [0.8840, 0.5991],
        [0.2883, 0.7138],
        [0.5881, 0.7938],
        [0.6937, 0.1809]])

MyTorch bk:
 [[0.69026393 0.39344826]]

PyTorch bk:
 tensor([0.6903, 0.3934])
Difference in Wk: 1.8491735404934512e-07
Difference in bk: 6.96111754757384e-08
```

MLP1

Test `forward()`

In [ ]:
```
num_hiddens=3

# initialize torch model, loss, optimizer
net = nn.Sequential(nn.Linear(num_inputs, num_hiddens),
                    nn.ReLU(),
                    nn.Linear(num_hiddens, num_outputs),
                    nn.Identity())
optimizer = torch.optim.SGD(net.parameters(), lr=0.1, momentum=0.0)

# initialize my network using torch W, b for each layer
W0 = net[0].weight.detach().numpy().T
b0 = net[0].bias.detach().numpy().T
W1 = net[2].weight.detach().numpy().T
b1 = net[2].bias.detach().numpy().T

mlp1 = MLP1(num_inputs, num_outputs, num_hiddens)
mlp1.layers[0].W = W0
mlp1.layers[0].b = b0
mlp1.layers[1].W = W1
mlp1.layers[1].b = b1

my_out = mlp1.forward(X)
```

```
torch_out = net(Xt)

print('MyTorch:\n', my_out, '\n')
print('PyTorch:\n', torch_out.data, '\n')
print('Difference:', np.linalg.norm(my_out - torch_out.data.numpy()))
```

```
MyTorch:
 [[-0.43635382  0.38668833]
 [-0.54950023  0.3628647 ]
 [-0.54950023  0.3628647 ]
 [-0.51382037  0.21551731]
 [-0.50974625  0.37123513]
 [-0.54950023  0.3628647 ]
 [-0.38876143  0.29533438]
 [-0.56818172  0.27561022]
 [-0.52001079  0.3601708 ]
 [-0.61013367  0.07966791]]

PyTorch:
 tensor([[-0.4364,  0.3867],
        [-0.5495,  0.3629],
        [-0.5495,  0.3629],
        [-0.5138,  0.2155],
        [-0.5097,  0.3712],
        [-0.5495,  0.3629],
        [-0.3888,  0.2953],
        [-0.5682,  0.2756],
        [-0.5200,  0.3602],
        [-0.6101,  0.0797]])

Difference: 7.751443391709816e-08
```

Test `backward()`

In [ ]:

```
my_mse_fn = mynn.MSELoss()
my_mse = my_mse_fn.forward(my_out, Y)
dLdZ = my_mse_fn.backward()
mlp1.backward(dLdZ)
my_dLdW0 = mlp1.layers[0].dLdW.T
my_dLdb0 = mlp1.layers[0].dLdb
my_dLdW1 = mlp1.layers[1].dLdW.T
my_dLdb1 = mlp1.layers[1].dLdb

optimizer.zero_grad()
torch_loss_fn = nn.MSELoss()
torch_loss = torch_loss_fn(torch_out, Yt)
torch_loss.backward(retain_graph=True)
torch_dLdW0 = net[0].weight.grad.data
torch_dLdb0 = net[0].bias.grad.data
torch_dLdW1 = net[2].weight.grad.data
torch_dLdb1 = net[2].bias.grad.data

print('Difference in dLdW0:', np.linalg.norm(my_dLdW0 - torch_dLdW0.data.num
print('Difference in dLdb0:', np.linalg.norm(my_dLdb0.flatten() - torch_dLdb
```

```
print('Difference in dLdW1:', np.linalg.norm(my_dLdW1 - torch_dLdW1.data.num
print('Difference in dLdb1:', np.linalg.norm(my_dLdb1.flatten() - torch_dLdb
```

```
Difference in dLdW0: 5.073911425469867e-08
Difference in dLdb0: 4.622361428754544e-09
Difference in dLdW1: 3.821583479869518e-08
Difference in dLdb1: 2.3672265170171044e-08
```

Test a single optimization step

In [ ]:
```
# my SGD step
my_optimizer = mytorch.optim.SGD(mlp1, lr=1)
my_optimizer.step()
my_Wk0 = mlp1.layers[0].W
my_bk0 = mlp1.layers[0].b
my_Wk1 = mlp1.layers[1].W
my_bk1 = mlp1.layers[1].b

# torch SGD step
optimizer.zero_grad()
torch_loss.backward(retain_graph=True)
optimizer.step()
torch_Wk0 = net[0].weight.data
torch_bk0 = net[0].bias.data
torch_Wk1 = net[2].weight.data
torch_bk1 = net[2].bias.data

print('Difference in Wk0:', np.linalg.norm(my_Wk0 - torch_Wk0.numpy().T))
print('Difference in bk0:', np.linalg.norm(my_bk0.flatten() - torch_bk0.nump
print('Difference in Wk1:', np.linalg.norm(my_Wk1 - torch_Wk1.numpy().T))
print('Difference in bk1:', np.linalg.norm(my_bk1.flatten() - torch_bk1.nump
```

```
Difference in Wk0: 0.4484992422899716
Difference in bk0: 0.11698040828986507
Difference in Wk1: 0.0828025291773284
Difference in bk1: 0.7600495461767015
```