COSE474-2024F: Deep Learing HW1

2.1 Data Manipulation

In [146…
```
import torch
```

In [147…
```
x = torch.arange(12, dtype=torch.float32)
x
```

Out[147…
```
tensor([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11.])
```

In [148…
```
x.numel()
```

Out[148…
```
12
```

In [149…
```
x.shape
```

Out[149…
```
torch.Size([12])
```

In [150…
```
X = x.reshape(3, 4)
X
```

Out[150…
```
tensor([[ 0.,  1.,  2.,  3.],
        [ 4.,  5.,  6.,  7.],
        [ 8.,  9., 10., 11.]])
```

n = h×w 이므로 n과 h를 알면 w는 몰라도 알 수 있음.

따라서 reshape(h, -1) = reshape(h, w)와 같음. (-1차원은 존재하지 않으므로 -1을 사용.)

In [151…
```
torch.zeros((2, 3, 4))
```

Out[151…
```
tensor([[[0., 0., 0., 0.],
         [0., 0., 0., 0.],
         [0., 0., 0., 0.]],

        [[0., 0., 0., 0.],
         [0., 0., 0., 0.],
         [0., 0., 0., 0.]]])
```

In [152…
```
torch.ones((2, 3, 4))
```

Out[152…
```
tensor([[[1., 1., 1., 1.],
         [1., 1., 1., 1.],
         [1., 1., 1., 1.]],

        [[1., 1., 1., 1.],
         [1., 1., 1., 1.],
         [1., 1., 1., 1.]]])
```

In [153…
```
torch.randn(3, 4)
```

Out[153…
```
tensor([[ 1.0614,  1.0812,  1.9287, -0.4680],
        [-0.5326,  0.1604,  0.3012, -0.9876],
        [-0.9359, -0.6125,  1.4360, -0.4317]])
```

In [154…
```
torch.tensor([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])
```

Out[154…
```
tensor([[2, 1, 4, 3],
        [1, 2, 3, 4],
        [4, 3, 2, 1]])
```

In [155…
```
X[-1], X[1:3]
```

Out[155…
```
(tensor([ 8.,  9., 10., 11.]),
 tensor([[ 4.,  5.,  6.,  7.],
         [ 8.,  9., 10., 11.]]))
```

In [156…
```
X[1, 2]=17
X
```

Out[156…
```
tensor([[ 0.,  1.,  2.,  3.],
        [ 4.,  5., 17.,  7.],
        [ 8.,  9., 10., 11.]])
```

In [157…
```
X[:2, :] = 12
X
```

```
Out[157… tensor([[12., 12., 12., 12.],
                 [12., 12., 12., 12.],
                 [ 8.,  9., 10., 11.]])
```

```
In [158… torch.exp(x)
```

```
Out[158… tensor([162754.7969, 162754.7969, 162754.7969, 162754.7969, 162754.7969,
                 162754.7969, 162754.7969, 162754.7969,   2980.9580,   8103.0840,
                  22026.4648,  59874.1406])
```

```
In [159… x=torch.tensor([1.0, 2, 4, 8])
         y = torch.tensor([2, 2, 2, 2])
         x+y, x-y, x*y, x/y, x**y
```

```
Out[159… (tensor([ 3.,  4.,  6., 10.]),
          tensor([-1.,  0.,  2.,  6.]),
          tensor([ 2.,  4.,  8., 16.]),
          tensor([0.5000, 1.0000, 2.0000, 4.0000]),
          tensor([ 1.,  4., 16., 64.]))
```

```
In [160… X = torch.arange(12, dtype=torch.float32).reshape((3, 4))
         Y = torch.tensor([[2.0, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])
         torch.cat((X, Y), dim=0), torch.cat((X, Y), dim=1)
```

```
Out[160… (tensor([[ 0.,  1.,  2.,  3.],
                  [ 4.,  5.,  6.,  7.],
                  [ 8.,  9., 10., 11.],
                  [ 2.,  1.,  4.,  3.],
                  [ 1.,  2.,  3.,  4.],
                  [ 4.,  3.,  2.,  1.]]),
          tensor([[ 0.,  1.,  2.,  3.,  2.,  1.,  4.,  3.],
                  [ 4.,  5.,  6.,  7.,  1.,  2.,  3.,  4.],
                  [ 8.,  9., 10., 11.,  4.,  3.,  2.,  1.]]))
```

```
In [161… X == Y
```

```
Out[161… tensor([[False,  True, False,  True],
                 [False, False, False, False],
                 [False, False, False, False]])
```

```
In [162… X.sum()
```

```
Out[162… tensor(66.)
```

```
In [163… a = torch.arange(3).reshape((3, 1))
         b = torch.arange(2).reshape((1, 2))
         a, b
```

```
Out[163… (tensor([[0],
                  [1],
                  [2]]),
          tensor([[0, 1]]))
```

```
In [164… a+b
```

```
Out[164… tensor([[0, 1],
                 [1, 2],
                 [2, 3]])
```

```
In [165… before = id(Y)
         Y = Y + X
         id(Y) == before
```

```
Out[165… False
```

```
In [166… Z = torch.zeros_like(Y)
         print('id(Z):', id(Z))
         Z[:] = X + Y
         print('id(Z):', id(Z))
```

```
id(Z): 131957874357040
id(Z): 131957874357040
```

```
In [167… before = id(X)
         X += Y
         id(X) == before
```

```
Out[167… True
```

```
In [168… A = X.numpy()
         B = torch.from_numpy(A)
         type(A), type(B)
```

Out[168… (numpy.ndarray, torch.Tensor)

In [169…
```
a = torch.tensor([3.5])
a, a.item(), float(a), int(a)
```

Out[169…
```
(tensor([3.5000]), 3.5, 3.5, 3)
```

2.2 Data Preprocessing

In [170…
```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (
2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.16.0)
```

In [171…
```
import os

os.makedirs(os.path.join('..', 'data'), exist_ok=True)
data_file = os.path.join('..', 'data', 'house_tiny.csv')
with open(data_file, 'w') as f:
    f.write('''NumRooms,RoofType,Price
NA,NA,127500
2,NA,106000
4,Slate,178100
NA,NA,140000''')
```

In [172…
```
import pandas as pd

data = pd.read_csv(data_file)
print(data)
```

```
   NumRooms RoofType   Price
0       NaN      NaN  127500
1       2.0      NaN  106000
2       4.0    Slate  178100
3       NaN      NaN  140000
```

In [173…
```
inputs, targets = data.iloc[:, 0:2], data.iloc[:, 2]
inputs = pd.get_dummies(inputs, dummy_na=True)
print(inputs)
```

```
   NumRooms  RoofType_Slate  RoofType_nan
0       NaN           False          True
1       2.0           False          True
2       4.0            True         False
3       NaN           False          True
```

In [174…
```
inputs = inputs.fillna(inputs.mean())
print(inputs)
```

```
   NumRooms  RoofType_Slate  RoofType_nan
0       3.0           False          True
1       2.0           False          True
2       4.0            True         False
3       3.0           False          True
```

In [175…
```
import torch

X= torch.tensor(inputs.to_numpy(dtype=float))
y=torch.tensor(targets.to_numpy(dtype=float))
X, y
```

Out[175…
```
(tensor([[3., 0., 1.],
         [2., 0., 1.],
         [4., 1., 0.],
         [3., 0., 1.]], dtype=torch.float64),
 tensor([127500., 106000., 178100., 140000.], dtype=torch.float64))
```

2.3 Linear Algebra

In [176…
```
import torch
```

In [177…
```
x=torch.tensor(3.0)
y=torch.tensor(2.0)

x+y, x*y, x/y, x**y
```

Out[177… (tensor(5.), tensor(6.), tensor(1.5000), tensor(9.))

In [178…
```
x=torch.arange(3)
x
```
Out[178…
```
tensor([0, 1, 2])
```
In [179…
```
x[2]
```
Out[179…
```
tensor(2)
```
In [180…
```
len(x)
```
Out[180…
```
3
```
In [181…
```
x.shape
```
Out[181…
```
torch.Size([3])
```
In [182…
```
A=torch.arange(6).reshape(3, 2)
A
```
Out[182…
```
tensor([[0, 1],
        [2, 3],
        [4, 5]])
```
In [183…
```
A.T
```
Out[183…
```
tensor([[0, 2, 4],
        [1, 3, 5]])
```
In [184…
```
A=torch.tensor([[1,2,3],[2,0,4],[3,4,5]])
A==A.T
```
Out[184…
```
tensor([[True, True, True],
        [True, True, True],
        [True, True, True]])
```
In [185…
```
torch.arange(24).reshape(2, 3, 4)
```
Out[185…
```
tensor([[[ 0,  1,  2,  3],
         [ 4,  5,  6,  7],
         [ 8,  9, 10, 11]],

        [[12, 13, 14, 15],
         [16, 17, 18, 19],
         [20, 21, 22, 23]]])
```
In [186…
```
A=torch.arange(6, dtype=torch.float32).reshape(2, 3)
B=A.clone()
A, A+B
```
Out[186…
```
(tensor([[0., 1., 2.],
         [3., 4., 5.]]),
 tensor([[ 0.,  2.,  4.],
         [ 6.,  8., 10.]]))
```
In [187…
```
A*B
```
Out[187…
```
tensor([[ 0.,  1.,  4.],
        [ 9., 16., 25.]])
```
In [188…
```
a=2
X=torch.arange(24).reshape(2,3,4)
a+X,(a*X).shape
```
Out[188…
```
(tensor([[[ 2,  3,  4,  5],
          [ 6,  7,  8,  9],
          [10, 11, 12, 13]],

         [[14, 15, 16, 17],
          [18, 19, 20, 21],
          [22, 23, 24, 25]]]),
 torch.Size([2, 3, 4]))
```
In [189…
```
x=torch.arange(3, dtype=torch.float32)
x, x.sum()
```
Out[189…
```
(tensor([0., 1., 2.]), tensor(3.))
```
In [190…
```
A.shape, A.sum()
```

```
Out[190… (torch.Size([2, 3]), tensor(15.))

In [191… A.shape, A.sum(axis=0).shape

Out[191… (torch.Size([2, 3]), torch.Size([3]))

In [192… A.shape, A.sum(axis=1).shape

Out[192… (torch.Size([2, 3]), torch.Size([2]))

In [193… A.sum(axis=[0,1]) == A.sum()

Out[193… tensor(True)

In [194… A.mean(), A.sum()/A.numel()

Out[194… (tensor(2.5000), tensor(2.5000))

In [195… A.mean(axis=0), A.sum(axis=0)/A.shape[0]

Out[195… (tensor([1.5000, 2.5000, 3.5000]), tensor([1.5000, 2.5000, 3.5000]))

In [196… sum_A=A.sum(axis=1, keepdims=True)
         sum_A, sum_A.shape

Out[196… (tensor([[ 3.],
                  [12.]]),
          torch.Size([2, 1]))

In [197… A / sum_A

Out[197… tensor([[0.0000, 0.3333, 0.6667],
                 [0.2500, 0.3333, 0.4167]])

In [198… A.cumsum(axis=0)

Out[198… tensor([[0., 1., 2.],
                 [3., 5., 7.]])

In [199… y=torch.ones(3, dtype=torch.float32)
         x, y, torch.dot(x, y)

Out[199… (tensor([0., 1., 2.]), tensor([1., 1., 1.]), tensor(3.))

In [200… torch.sum(x*y)

Out[200… tensor(3.)

In [201… A.shape, x.shape, torch.mv(A, x), A@x

Out[201… (torch.Size([2, 3]), torch.Size([3]), tensor([ 5., 14.]), tensor([ 5., 14.]))

In [202… B = torch.ones(3, 4)
         torch.mm(A, B), A@B

Out[202… (tensor([[ 3.,  3.,  3.,  3.],
                  [12., 12., 12., 12.]]),
          tensor([[ 3.,  3.,  3.,  3.],
                  [12., 12., 12., 12.]]))

In [203… u = torch.tensor([3.0, -4.0])
         torch.norm(u)

Out[203… tensor(5.)

In [204… torch.abs(u).sum()

Out[204… tensor(7.)

In [205… torch.norm(torch.ones((4, 9)))

Out[205… tensor(6.)
```

2.5 Automatic Differentiation

```
In [206… import torch

In [207… x=torch.arange(4.0)
```

```
x
```

Out[207…
```
tensor([0., 1., 2., 3.])
```

In [208…
```
x.requires_grad_(True)
x.grad
```

In [209…
```
y=2*torch.dot(x, x)
y
```

Out[209…
```
tensor(28., grad_fn=<MulBackward0>)
```

In [210…
```
y.backward()
x.grad
```

Out[210…
```
tensor([ 0.,  4.,  8., 12.])
```

In [211…
```
x.grad == 4*x
```

Out[211…
```
tensor([True, True, True, True])
```

In [212…
```
x.grad.zero_()
y=x.sum()
y.backward()
x.grad
```

Out[212…
```
tensor([1., 1., 1., 1.])
```

In [213…
```
x.grad.zero_()
y=x*x
y.backward(gradient=torch.ones(len(y)))
x.grad
```

Out[213…
```
tensor([0., 2., 4., 6.])
```

In [214…
```
x.grad.zero_()
y=x*x
u=y.detach()
z=u*x

z.sum().backward()
x.grad == u
```

Out[214…
```
tensor([True, True, True, True])
```

In [215…
```
x.grad.zero_()
y.sum().backward()
x.grad == 2*x
```

Out[215…
```
tensor([True, True, True, True])
```

In [216…
```
def f(a):
    b=a*2
    while b.norm()<1000:
        b=b*2
    if b.sum()>0:
        c=b
    else:
        c=100*b
    return c
```

In [217…
```
a=torch.randn(size=(), requires_grad=True)
d=f(a)
d.backward()
```

In [218…
```
a.grad == d/a
```

Out[218…
```
tensor(True)
```

3.1 Linear Regression

In [219…
```
pip install d2l
```

```
Requirement already satisfied: d2l in /usr/local/lib/python3.10/dist-packages (1.0.3)
Requirement already satisfied: jupyter==1.0.0 in /usr/local/lib/python3.10/dist-packages (from d2l) (1.0.0)
Requirement already satisfied: numpy==1.23.5 in /usr/local/lib/python3.10/dist-packages (from d2l) (1.23.5)
Requirement already satisfied: matplotlib==3.7.2 in /usr/local/lib/python3.10/dist-packages (from d2l) (3.7.2)
Requirement already satisfied: matplotlib-inline==0.1.6 in /usr/local/lib/python3.10/dist-packages (from d2l) (0
.1.6)
```

```
Requirement already satisfied: requests==2.31.0 in /usr/local/lib/python3.10/dist-packages (from d2l) (2.31.0)
Requirement already satisfied: pandas==2.0.3 in /usr/local/lib/python3.10/dist-packages (from d2l) (2.0.3)
Requirement already satisfied: scipy==1.10.1 in /usr/local/lib/python3.10/dist-packages (from d2l) (1.10.1)
Requirement already satisfied: notebook in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l) (6
.5.5)
Requirement already satisfied: qtconsole in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l) (
5.6.0)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->
d2l) (6.1.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l) (
6.5.4)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l) (
5.5.6)
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l)
(7.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7
.2->d2l) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2->
d2l) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.
7.2->d2l) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.
7.2->d2l) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.
2->d2l) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2-
>d2l) (10.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib
==3.7.2->d2l) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib=
=3.7.2->d2l) (2.8.2)
Requirement already satisfied: traitlets in /usr/local/lib/python3.10/dist-packages (from matplotlib-inline==0.1
.6->d2l) (5.7.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3->d2l)
(2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3->d2
l) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
s==2.31.0->d2l) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests==2.31.0->d
2l) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests==2.3
1.0->d2l) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests==2.3
1.0->d2l) (2024.8.30)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->m
atplotlib==3.7.2->d2l) (1.16.0)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupy
ter==1.0.0->d2l) (0.2.0)
Requirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyte
r==1.0.0->d2l) (7.34.0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyte
r==1.0.0->d2l) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyter=
=1.0.0->d2l) (6.3.3)
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.10/dist-packages (from ipywid
gets->jupyter==1.0.0->d2l) (3.6.9)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywid
gets->jupyter==1.0.0->d2l) (3.0.13)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from jupyter-console->jupyter==1.0.0->d2l) (3.0.47)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from jupyter-console->jupyte
r==1.0.0->d2l) (2.18.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0->
d2l) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyte
r==1.0.0->d2l) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0
->d2l) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1
.0.0->d2l) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from nbconvert->ju
pyter==1.0.0->d2l) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==
1.0.0->d2l) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jup
yter==1.0.0->d2l) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert->j
upyter==1.0.0->d2l) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyt
er==1.0.0->d2l) (2.1.5)
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jup
yter==1.0.0->d2l) (0.8.4)
```

```
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyt
er==1.0.0->d2l) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter
==1.0.0->d2l) (5.10.4)
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->
jupyter==1.0.0->d2l) (1.5.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0
.0->d2l) (1.3.0)
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter=
=1.0.0->d2l) (24.0.1)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter==1
.0.0->d2l) (23.1.0)
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python3.10/dist-packages (from notebook->jupy
ter==1.0.0->d2l) (1.6.0)
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from notebook->jupy
ter==1.0.0->d2l) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyt
er==1.0.0->d2l) (0.18.1)
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.10/dist-packages (from notebook->jupy
ter==1.0.0->d2l) (0.20.0)
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyt
er==1.0.0->d2l) (1.1.0)
Requirement already satisfied: qtpy>=2.4.0 in /usr/local/lib/python3.10/dist-packages (from qtconsole->jupyter==
1.0.0->d2l) (2.4.1)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0-
>ipykernel->jupyter==1.0.0->d2l) (71.0.4)
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipyke
rnel->jupyter==1.0.0->d2l) (0.19.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipyker
nel->jupyter==1.0.0->d2l) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipyk
ernel->jupyter==1.0.0->d2l) (0.7.5)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipykern
el->jupyter==1.0.0->d2l) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipyk
ernel->jupyter==1.0.0->d2l) (4.9.0)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core>=
4.7->nbconvert->jupyter==1.0.0->d2l) (4.3.6)
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=
0.4.7->notebook->jupyter==1.0.0->d2l) (0.2.4)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5
.1->nbconvert->jupyter==1.0.0->d2l) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->n
bconvert->jupyter==1.0.0->d2l) (4.23.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!=
3.0.1,<3.1.0,>=2.0.0->jupyter-console->jupyter==1.0.0->d2l) (0.2.13)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.10/dist-packages (from terminado>=0.8.3->not
ebook->jupyter==1.0.0->d2l) (0.7.0)
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.10/dist-packages (from argon2-cffi
->notebook->jupyter==1.0.0->d2l) (21.2.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->nb
convert->jupyter==1.0.0->d2l) (2.6)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert->
jupyter==1.0.0->d2l) (0.5.1)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from jedi>=0.16->
ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l) (0.8.4)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->n
bformat>=5.1->nbconvert->jupyter==1.0.0->d2l) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (
from jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter==1.0.0->d2l) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=
2.6->nbformat>=5.1->nbconvert->jupyter==1.0.0->d2l) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->
nbformat>=5.1->nbconvert->jupyter==1.0.0->d2l) (0.20.0)
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.10/dist-packages (from notebook-
shim>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.24.0)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from argon2-cffi-bindings
->argon2-cffi->notebook->jupyter==1.0.0->d2l) (1.17.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0.1->argon2-cf
fi-bindings->argon2-cffi->notebook->jupyter==1.0.0->d2l) (2.22)
Requirement already satisfied: anyio<4,>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from jupyter-server<3
,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (3.7.1)
Requirement already satisfied: websocket-client in /usr/local/lib/python3.10/dist-packages (from jupyter-server<
3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.8.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.1.0->ju
pyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.1.0->
jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook->jupyter==1.0.0->d2l) (1.2.2)
```

In [220…

```python
%matplotlib inline
import random
import torch
from d2l import torch as d2l
```

In [221…
```
n = 10000
a = torch.ones(n)
b = torch.ones(n)
```

In [222…
```
import time
c = torch.zeros(n)
t = time.time()
for i in range(n):
    c[i] = a[i] + b[i]
f'{time.time() - t:.5f} sec'
```

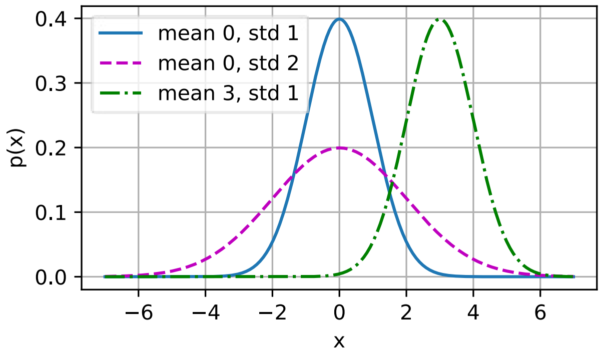
Out[222…
```
'0.24468 sec'
```

In [223…
```
t=time.time()
d=a+b
f'{time.time()-t:.5f} sec'
```

Out[223…
```
'0.00206 sec'
```

In [224…
```
import math
import numpy as np
def normal(x, mu, sigma):
    p = 1 / math.sqrt(2 * math.pi * sigma**2)
    return p * np.exp(-0.5 * (x - mu)**2 / sigma**2)
```

In [225…
```
from typing_extensions import ParamSpec
x = np.arange(-7, 7, 0.01)

params = [(0, 1), (0, 2), (3, 1)]
d2l.plot(x, [normal(x, mu, sigma) for mu, sigma in params], xlabel='x',
         ylabel='p(x)', figsize=(4.5, 2.5),
         legend=[f'mean {mu}, std {sigma}' for mu, sigma in params])
```



3.2 Object-Oriented Design for Implementation

In [226…
```
import time
import numpy as np
import torch
from torch import nn
from d2l import torch as d2l
```

In [227…
```
def add_to_class(Class):
    def wrapper(obj):
        setattr(Class, obj.__name__, obj)
    return wrapper
```

In [228…
```
class A:
    def __init__(self):
        self.b = 1

a = A()
```

In [229…
```
@add_to_class(A)
def do(self):
    print('Class attribute "b" is', self.b)

a.do()
```

```
Class attribute "b" is 1
```

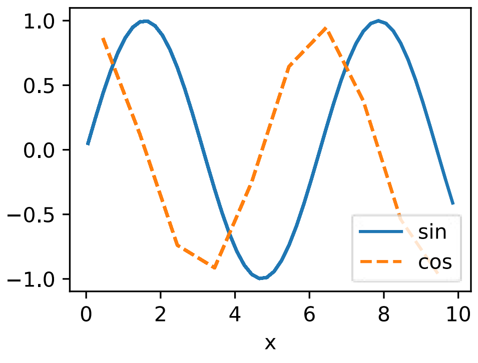
In [230…
```
class HyperParameters:
    def save_hyperparameters(self, ignore=[]):
        raise NotImplemented
```

In [231…
```python
class B(d2l.HyperParameters):
    def __init__(self, a, b, c):
        self.save_hyperparameters(ignore=['c'])
        print('self.a =', self.a, 'self.b =', self.b)
        print('There is no self.c =', not hasattr(self, 'c'))

b = B(a=1, b=2, c=3)
```

```
self.a = 1 self.b = 2
There is no self.c = True
```

In [232…
```python
class ProgressBoard(d2l.HyperParameters):
    def __init__(self, xlabel=None, ylabel=None, xlim=None,
                 ylim=None, xscale='linear', yscale='linear',
                 ls=['-', '--', '-.', ':'], colors=['C0', 'C1', 'C2', 'C3'],
                 fig=None, axes=None, figsize=(3.5, 2.5), display=True):
        self.save_hyperparameters()

    def draw(self, x, y, label, every_n=1):
        raise NotImplemented
```

In [233…
```python
board = d2l.ProgressBoard('x')
for x in np.arange(0, 10, 0.1):
    board.draw(x, np.sin(x), 'sin', every_n=2)
    board.draw(x, np.cos(x), 'cos', every_n=10)
```



In [234…
```python
class Module(nn.Module, d2l.HyperParameters):
    def __init__(self, plot_train_per_epoch=2, plot_valid_per_epoch=1):
        super().__init__()
        self.save_hyperparameters()
        self.board = ProgressBoard()

    def loss(self, y_hat, y):
        raise NotImplementedError

    def forward(self, X):
        assert hasattr(self, 'net'), 'Neural network is defined'
        return self.net(X)

    def plot(self, key, value, train):
        assert hasattr(self, 'trainer'), 'Trainer is not inited'
        self.board.xlabel = 'epoch'
        if train:
            x = self.trainer.train_batch_idx / \
                self.trainer.num_train_batches
            n = self.trainer.num_train_batches / \
                self.plot_train_per_epoch
        else:
            x = self.trainer.epoch + 1
            n = self.trainer.num_val_batches / \
                self.plot_valid_per_epoch
        self.board.draw(x, value.to(d2l.cpu()).detach().numpy(),
                        ('train_' if train else 'val_') + key,
                        every_n=int(n))

    def training_step(self, batch):
        l = self.loss(self(*batch[:-1]), batch[-1])
        self.plot('loss', l, train=True)
        return l

    def validation_step(self, batch):
        l = self.loss(self(*batch[:-1]), batch[-1])
        self.plot('loss', l, train=False)

    def configure_optimizers(self):
        raise NotImplementedError
```

In [235…
```python
class DataModule(d2l.HyperParameters):
    def __init__(self, root='../data', num_workers=4):
        self.save_hyperparameters()

    def get_dataloader(self, train):
        raise NotImplementedError

    def train_dataloader(self):
        return self.get_dataloader(train=True)

    def val_dataloader(self):
        return self.get_dataloader(train=False)
```

In [236…

```
class Trainer(d2l.HyperParameters):
    def __init__(self, max_epochs, num_gpus=0, gradient_clip_val=0):
        self.save_hyperparameters()
        assert num_gpus == 0, 'No GPU support yet'

    def prepare_data(self, data):
        self.train_dataloader = data.train_dataloader()
        self.val_dataloader = data.val_dataloader()
        self.num_train_batches = len(self.train_dataloader)
        self.num_val_batches = (len(self.val_dataloader)
                                if self.val_dataloader is not None else 0)

    def prepare_model(self, model):
        model.trainer = self
        model.board.xlim = [0, self.max_epochs]
        self.model = model

    def fit(self, model, data):
        self.prepare_data(data)
        self.prepare_model(model)
        self.optim = model.configure_optimizers()
        self.epoch = 0
        self.train_batch_idx = 0
        self.val_batch_idx = 0
        for self.epoch in range(self.max_epochs):
            self.fit_epoch()

    def fit_epoch(self):
        raise NotImplementedError
```

3.4 Linear Regression Implementation from Scratch

In [237…

```
%matplotlib inline
import torch
from d2l import torch as d2l
```

In [238…

```
class LinearRegressionScratch(d2l.Module):
    def __init__(self, num_inputs, lr, sigma=0.01):
        super().__init__()
        self.save_hyperparameters()
        self.w = torch.normal(0, sigma, (num_inputs, 1), requires_grad=True)
        self.b = torch.zeros(1, requires_grad=True)
```

In [239…

```
@d2l.add_to_class(LinearRegressionScratch)
def forward(self, X):
    return torch.matmul(X, self.w) + self.b
```

In [240…

```
@d2l.add_to_class(LinearRegressionScratch)
def loss(self, y_hat, y):
    l = (y_hat - y) ** 2 / 2
    return l.mean()
```

In [241…

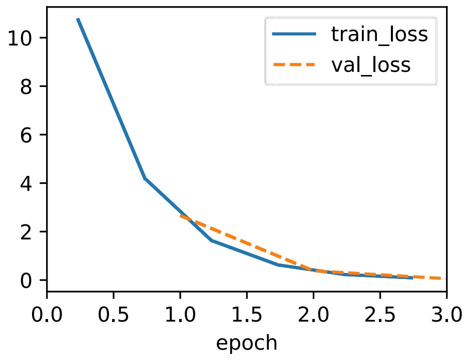
```
class SGD(d2l.HyperParameters):
    def __init__(self, params, lr):
        self.save_hyperparameters()

    def step(self):
        for param in self.params:
            param -= self.lr * param.grad

    def zero_grad(self):
        for param in self.params:
            if param.grad is not None:
                param.grad.zero_()
```

In [242…

```
@d2l.add_to_class(LinearRegressionScratch)
def configure_optimizers(self):
    return SGD([self.w, self.b], self.lr)
```

In [243…
```
@d2l.add_to_class(d2l.Trainer)
def prepare_batch(self, batch):
    return batch

@d2l.add_to_class(d2l.Trainer)
def fit_epoch(self):
    self.model.train()
    for batch in self.train_dataloader:
        loss = self.model.training_step(self.prepare_batch(batch))
        self.optim.zero_grad()
        with torch.no_grad():
            loss.backward()
            if self.gradient_clip_val > 0:
                self.clip_gradients(self.gradient_clip_val, self.model)
            self.optim.step()
        self.train_batch_idx += 1
    if self.val_dataloader is None:
        return
    self.model.eval()
    for batch in self.val_dataloader:
        with torch.no_grad():
            self.model.validation_step(self.prepare_batch(batch))
        self.val_batch_idx += 1
```

In [244…
```
model = LinearRegressionScratch(2, lr=0.03)
data = d2l.SyntheticRegressionData(w=torch.tensor([2, -3.4]), b=4.2)
trainer = d2l.Trainer(max_epochs=3)
trainer.fit(model, data)
```



In [245…
```
with torch.no_grad():
    print(f'error in estimating w: {data.w - model.w.reshape(data.w.shape)}')
    print(f'error in estimating b: {data.b - model.b}')
```

```
error in estimating w: tensor([ 0.1250, -0.1876])
error in estimating b: tensor([0.2367])
```

4.1 Softmax Regression

4.2 The Image Classification Dataset

In [246…
```
%matplotlib inline
import time
import torch
import torchvision
from torchvision import transforms
from d2l import torch as d2l

d2l.use_svg_display()
```

In [247…
```
class FashionMNIST(d2l.DataModule):
    def __init__(self, batch_size=64, resize=(28, 28)):
        super().__init__()
        self.save_hyperparameters()
        trans = transforms.Compose([transforms.Resize(resize),
                                    transforms.ToTensor()])
        self.train = torchvision.datasets.FashionMNIST(
            root=self.root, train=True, transform=trans, download=True)
        self.val = torchvision.datasets.FashionMNIST(
            root=self.root, train=False, transform=trans, download=True)
```

In [248…
```
data = FashionMNIST(resize=(32, 32))
len(data.train), len(data.val)
```

Out[248…
```
(60000, 10000)
```

In [249…
```
data.train[0][0].shape
```

Out[249…
```
torch.Size([1, 32, 32])
```

In [250…
```
@d2l.add_to_class(FashionMNIST)
def text_labels(self, indices):
    """Return text labels."""
    labels = ['t-shirt', 'trouser', 'pullover', 'dress', 'coat',
              'sandal', 'shirt', 'sneaker', 'bag', 'ankle boot']
    return [labels[int(i)] for i in indices]
```

In [251…
```
@d2l.add_to_class(FashionMNIST)
def get_dataloader(self, train):
    data = self.train if train else self.val
    return torch.utils.data.DataLoader(data, self.batch_size, shuffle=train,
                                       num_workers=self.num_workers)
```

In [252…
```
X, y = next(iter(data.train_dataloader()))
print(X.shape, X.dtype, y.shape, y.dtype)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will cr
eate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
torch.Size([64, 1, 32, 32]) torch.float32 torch.Size([64]) torch.int64
```

In [253…
```
tic = time.time()
for X, y in data.train_dataloader():
    continue
f'{time.time() - tic:.2f} sec'
```

Out[253…
```
'13.29 sec'
```

In [254…
```
def show_images(imgs, num_rows, num_cols, titles=None, scale=1.5):
    """Plot a list of images."""
    raise NotImplementedError
```

In [255…
```
@d2l.add_to_class(FashionMNIST)
def visualize(self, batch, nrows=1, ncols=8, labels=[]):
    X, y = batch
    if not labels:
        labels = self.text_labels(y)
    d2l.show_images(X.squeeze(1), nrows, ncols, titles=labels)
batch = next(iter(data.val_dataloader()))
data.visualize(batch)
```

4.3 The Base Classification Model

In [256…
```
import torch
from d2l import torch as d2l
```

In [257…
```
class Classifier(d2l.Module):
    """The base class of classification models."""
    def validation_step(self, batch):
        Y_hat = self(*batch[:-1])
        self.plot('loss', self.loss(Y_hat, batch[-1]), train=False)
        self.plot('acc', self.accuracy(Y_hat, batch[-1]), train=False)
```

In [258…
```
@d2l.add_to_class(d2l.Module)
def configure_optimizers(self):
    return torch.optim.SGD(self.parameters(), lr=self.lr)
```

In [259…
```
@d2l.add_to_class(Classifier)
def accuracy(self, Y_hat, Y, averaged=True):
    Y_hat = Y_hat.reshape((-1, Y_hat.shape[-1]))
    preds = Y_hat.argmax(axis=1).type(Y.dtype)
    compare = (preds == Y.reshape(-1)).type(torch.float32)
    return compare.mean() if averaged else compare
```

4.4 Softmax Regression Implementation from Scratch

In [260…
```
import torch
from d2l import torch as d2l
```

In [261…
```
X = torch.tensor([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
X.sum(0, keepdims=True), X.sum(1, keepdims=True)
```

Out[261…
```
(tensor([[5., 7., 9.]]),
 tensor([[ 6.],
         [15.]]))
```

In [262…
```
def softmax(X):
    X_exp = torch.exp(X)
    partition = X_exp.sum(1, keepdims=True)
    return X_exp / partition
```

In [263…
```
X = torch.rand((2, 5))
X_prob = softmax(X)
X_prob, X_prob.sum(1)
```

Out[263…
```
(tensor([[0.2748, 0.1371, 0.2134, 0.2214, 0.1534],
         [0.1712, 0.1546, 0.2515, 0.1910, 0.2317]]),
 tensor([1.0000, 1.0000]))
```

In [264…
```
class SoftmaxRegressionScratch(d2l.Classifier):
    def __init__(self, num_inputs, num_outputs, lr, sigma=0.01):
        super().__init__()
        self.save_hyperparameters()
        self.W = torch.normal(0, sigma, size=(num_inputs, num_outputs),
                              requires_grad=True)
        self.b = torch.zeros(num_outputs, requires_grad=True)

    def parameters(self):
        return [self.W, self.b]
```

In [265…
```
@d2l.add_to_class(SoftmaxRegressionScratch)
def forward(self, X):
    X = X.reshape((-1, self.W.shape[0]))
    return softmax(torch.matmul(X, self.W) + self.b)
```

In [266…
```
y = torch.tensor([0, 2])
y_hat = torch.tensor([[0.1, 0.3, 0.6], [0.3, 0.2, 0.5]])
y_hat[[0, 1], y]
```

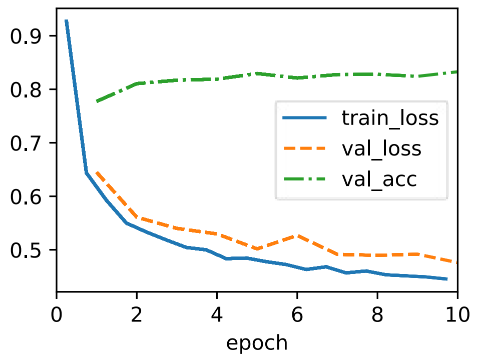
Out[266…
```
tensor([0.1000, 0.5000])
```

In [267…
```
def cross_entropy(y_hat, y):
    return -torch.log(y_hat[list(range(len(y_hat))), y]).mean()

cross_entropy(y_hat, y)
```

Out[267…
```
tensor(1.4979)
```

In [268…
```
@d2l.add_to_class(SoftmaxRegressionScratch)
def loss(self, y_hat, y):
    return cross_entropy(y_hat, y)
```

In [269…
```
data = d2l.FashionMNIST(batch_size=256)
model = SoftmaxRegressionScratch(num_inputs=784, num_outputs=10, lr=0.1)
trainer = d2l.Trainer(max_epochs=10)
trainer.fit(model, data)
```

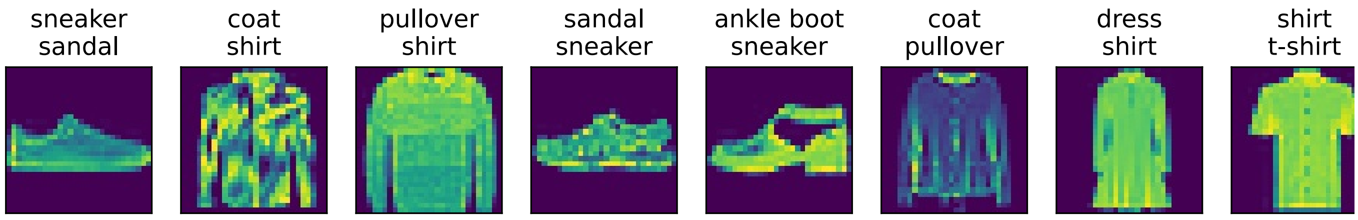


In [270…
```
X, y = next(iter(data.val_dataloader()))
```

```
preds = model(X).argmax(axis=1)
preds.shape
```

Out[270… torch.Size([256])

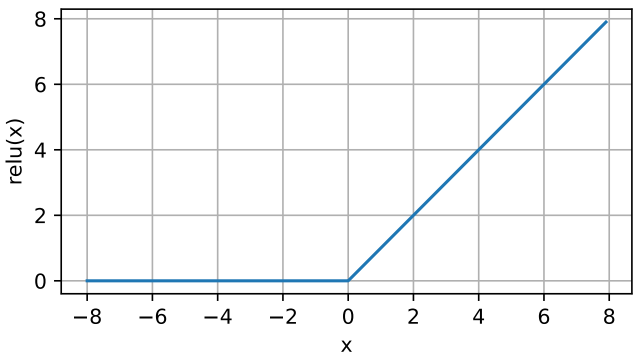
In [271…
```
wrong = preds.type(y.dtype) != y
X, y, preds = X[wrong], y[wrong], preds[wrong]
labels = [a+'\n'+b for a, b in zip(
    data.text_labels(y), data.text_labels(preds))]
data.visualize([X, y], labels=labels)
```



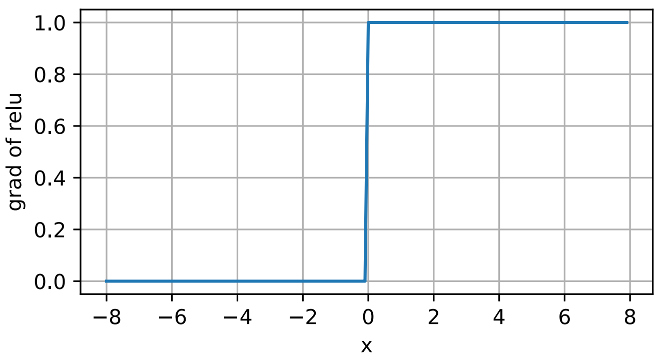
5.1 Multilayer Perceptrons

In [272…
```
%matplotlib inline
import torch
from d2l import torch as d2l
```

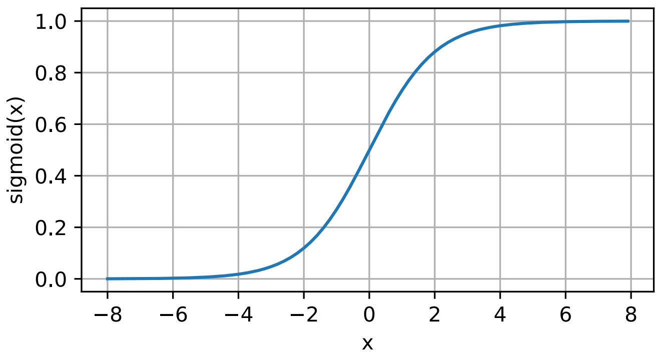
In [273…
```
x = torch.arange(-8.0, 8.0, 0.1, requires_grad=True)
y = torch.relu(x)
d2l.plot(x.detach(), y.detach(), 'x', 'relu(x)', figsize=(5, 2.5))
```



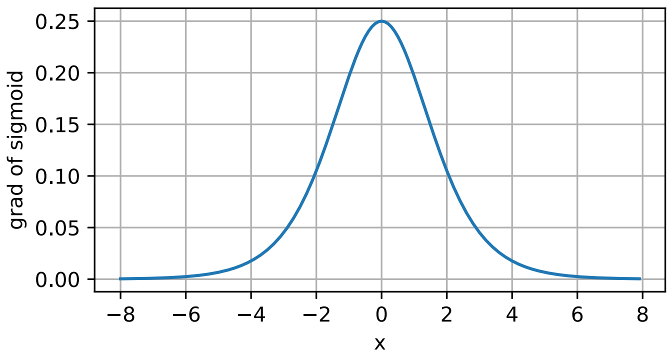
In [274…
```
y.backward(torch.ones_like(x), retain_graph=True)
d2l.plot(x.detach(), x.grad, 'x', 'grad of relu', figsize=(5, 2.5))
```



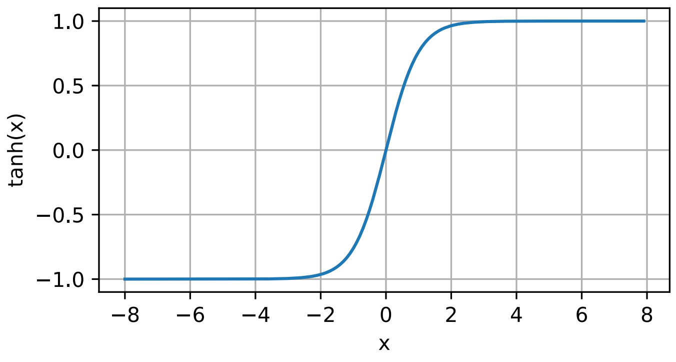
In [275…
```
y = torch.sigmoid(x)
d2l.plot(x.detach(), y.detach(), 'x', 'sigmoid(x)', figsize=(5, 2.5))
```



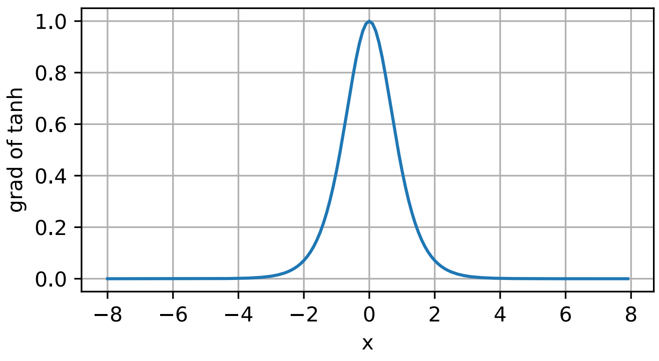
In [276…
```
x.grad.data.zero_()
y.backward(torch.ones_like(x),retain_graph=True)
d2l.plot(x.detach(), x.grad, 'x', 'grad of sigmoid', figsize=(5, 2.5))
```



In [277…
```
y = torch.tanh(x)
d2l.plot(x.detach(), y.detach(), 'x', 'tanh(x)', figsize=(5, 2.5))
```



In [278…
```
x.grad.data.zero_()
y.backward(torch.ones_like(x),retain_graph=True)
d2l.plot(x.detach(), x.grad, 'x', 'grad of tanh', figsize=(5, 2.5))
```



5.2 Implementation of Multilayer Perceptrons

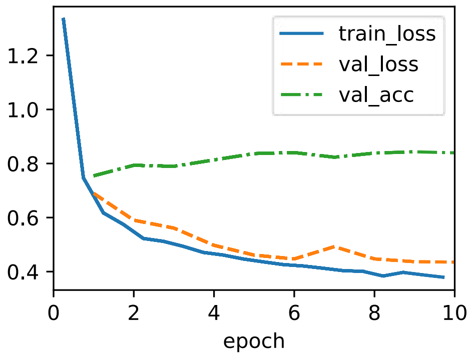
In [279…
```
import torch
from torch import nn
from d2l import torch as d2l
```

In [280…
```
class MLPScratch(d2l.Classifier):
    def __init__(self, num_inputs, num_outputs, num_hiddens, lr, sigma=0.01):
        super().__init__()
        self.save_hyperparameters()
        self.W1 = nn.Parameter(torch.randn(num_inputs, num_hiddens) * sigma)
        self.b1 = nn.Parameter(torch.zeros(num_hiddens))
        self.W2 = nn.Parameter(torch.randn(num_hiddens, num_outputs) * sigma)
        self.b2 = nn.Parameter(torch.zeros(num_outputs))
```

In [281…
```
def relu(X):
    a = torch.zeros_like(X)
    return torch.max(X, a)
```

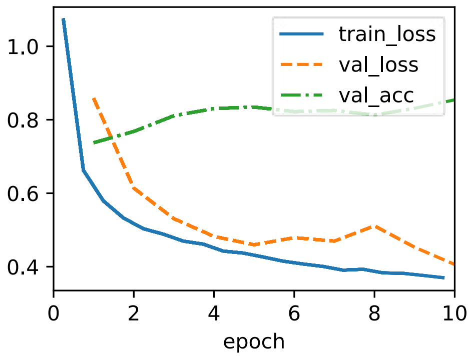
In [282…
```
@d2l.add_to_class(MLPScratch)
def forward(self, X):
    X = X.reshape((-1, self.num_inputs))
    H = relu(torch.matmul(X, self.W1) + self.b1)
    return torch.matmul(H, self.W2) + self.b2
```

In [283…
```
model = MLPScratch(num_inputs=784, num_outputs=10, num_hiddens=256, lr=0.1)
data = d2l.FashionMNIST(batch_size=256)
trainer = d2l.Trainer(max_epochs=10)
trainer.fit(model, data)
```



In [284…
```
class MLP(d2l.Classifier):
    def __init__(self, num_outputs, num_hiddens, lr):
        super().__init__()
        self.save_hyperparameters()
        self.net = nn.Sequential(nn.Flatten(), nn.LazyLinear(num_hiddens),
                                 nn.ReLU(), nn.LazyLinear(num_outputs))
```

In [285…
```
model = MLP(num_outputs=10, num_hiddens=256, lr=0.1)
trainer.fit(model, data)
```



5.3 Forward Propagation, Backward Propagation, and Computational Graphs

Discussions & Exercises

2챕터 discussion

- 딥러닝 분야에서 단순한 n차원 배열이 아니라 텐서를 사용함으로써 얻는 장점에는 무엇이 있을까?
- 브로드 캐스팅은 평소 내가 알던 행렬의 계산에 위배되는 계산인데, 왜 이런 편리한 연산을 텐서가 아닌 일반적인 행렬의 계산에서는 사용하지 않는 걸까?
- 결측값을 처리해주는 방식들은 추후 모델 학습에 있어 굉장히 중요하기 때문에 알아둘 필요가 있는 것 같다. 각 상황마다 선택할 수 있는 적절한 결측값 처리법이 매뉴얼화 되어 있는지 궁금하다. 예를 들어 어떤 상황에선 결측값 제거가 낫고, 어떤 상황에선 평균치로 채우는 게 낫다 하는 기준이 있을까?
- autograd를 통해 복잡한 미분 계산을 간단히 처리할 수 있게 되었구나. 순전파는 어떻게 이루어지는지 알아봐야겠다.

3챕터 discussion

- 적절한 학습률과 데이터셋을 준비하는 능력이 곧 머신러닝 능력을 결정할 것 같다.
- 경사 하강법 외의 다른 최적화 알고리즘에는 무엇이 있을까?
- 찾은 최적값이 국소 최적인지 전역 최적인지는 어떻게 알 수 있는걸까?
- 실무에서는 이미 짜여진 프로그램을 쓰겠지만 간략하게 모델과 최적화를 위한 함수들을 정의함으로써 딥러닝 과정이 어떻게 이루어지는지에 대한 이해가 한층 용이했다.

4챕터 discussion

- 물론 시그모이드 같은 함수도 있지만 어쨌든 복잡해보이는 분류의 기반이 되는 소프트맥스 함수가 간단하다는 점이 놀랍다.
- 텐서, 뉴런, 엔트로피 등을 보니 머신러닝에서 수학 외의 다른 학문의 개념(물리학, 생명과학 등)을 어느정도 알고 있는 것이 얼마나 중요한지 느껴진다.
- 소프트맥스 함수가 대두되게 된 배경(비음수성과 정규화)을 알게됐다.

5챕터 discussion

- 은닉층이 늘어남과 뉴런이 많아짐은 각각 학습에 어떤 영향을 어떻게 끼치는 건지에 대해, 계층과 계층 간의 입출력에 대한 설명(순전파, 역전파)을 읽고 직접 코드를 작성하며 대략적으로나마 알 수 있었다.
- 은닉층과 뉴런의 적절한 수는 어떻게 판단할 수 있을까?
- 역전파는 gradient를 계산하기에 아주 훌륭한 도구이지만 메모리 사용량이 크다는 문제가 있다고 한다. 이를 극복할만한 대안은 없을까?

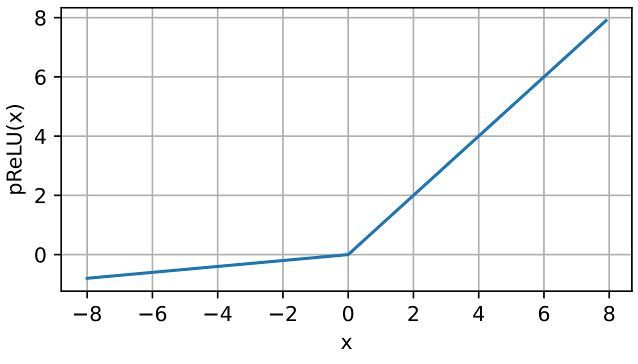
Own Exercise

- pReLU

$$\text{pReLU}(x)=\max(0, x) + \alpha\min(0, x)$$

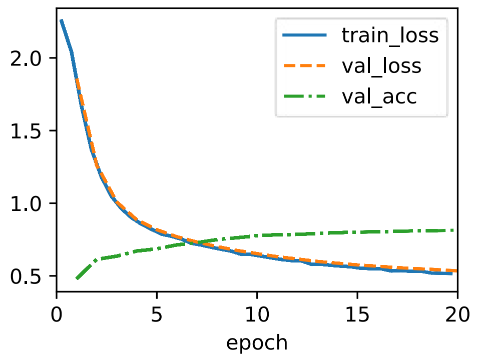
In [286…
```python
pReLU = lambda x, a: torch.max(torch.tensor(0), x) + a * torch.min(torch.tensor(0), x)
```

In [287…
```python
y = pReLU(x, 0.1)
d2l.plot(x.detach(), y.detach(), 'x', 'pReLU(x)', figsize=(5, 2.5))
```



- 훈련 모델 수치 조정(에폭, 학습률)

In [288…
```python
model = MLPScratch(num_inputs=784, num_outputs=10, num_hiddens=256, lr=0.01)
data = d2l.FashionMNIST(batch_size=256)
trainer = d2l.Trainer(max_epochs=20)
trainer.fit(model, data)
```



- 에폭을 늘리고(20) 학습률을 낮췄더니(0.01), 과적합의 위험이 있는 것으로 보인다.