COSE474-2024F: DEEP LEARNING HW1

2022320338 데이터과학과 신민서

0.1 Installation

```
!pip install d2l==1.0.3
```

```
Collecting d2l==1.0.3
  Downloading d2l-1.0.3-py3-none-any.whl.metadata (556 bytes)
Collecting jupyter==1.0.0 (from d2l==1.0.3)
  Downloading jupyter-1.0.0-py2.py3-none-any.whl.metadata (995 bytes)
Collecting numpy==1.23.5 (from d2l==1.0.3)
  Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.3 kB)
Collecting matplotlib==3.7.2 (from d2l==1.0.3)
  Downloading matplotlib-3.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.6 kB)
Collecting matplotlib-inline==0.1.6 (from d2l==1.0.3)
  Downloading matplotlib_inline-0.1.6-py3-none-any.whl.metadata (2.8 kB)
Collecting requests==2.31.0 (from d2l==1.0.3)
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting pandas==2.0.3 (from d2l==1.0.3)
  Downloading pandas-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Collecting scipy==1.10.1 (from d2l==1.0.3)
  Downloading scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.9/58.9 kB 4.5 MB/s eta 0:00:00
Requirement already satisfied: notebook in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l==1.0.3)
Collecting qtconsole (from jupyter==1.0.0->d2l==1.0.3)
  Downloading qtconsole-5.6.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l=
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l==1.0.3
Requirement already satisfied: ipykernel in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l==1.0.3
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.10/dist-packages (from jupyter==1.0.0->d2l==1.0.
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2->
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2->d2l=
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2->d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7.2->d2l
Collecting pyparsing<3.1,>=2.3.1 (from matplotlib==3.7.2->d2l==1.0.3)
  Downloading pyparsing-3.0.9-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.7
Requirement already satisfied: traitlets in /usr/local/lib/python3.10/dist-packages (from matplotlib-inline==0.1.6->
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3->d2l==1.0
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas==2.0.3->d2l==1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests==2
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests==2.31.0->d2l==
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests==2.31.0-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests==2.31.0-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matpl
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyter=
Requirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyter==1
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyter==1
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipykernel->jupyter==1.0
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidgets
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from jupyter-console>=jupyter==1
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2l=
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0->d2
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyte
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyt
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupy
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (from nbconvert->jupyter==1.0.0->
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter==1.0
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter==1.0.0
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter=
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter=
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter==
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter=
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.10/dist-packages (from notebook->jupyter==
Collecting qtpy>=2.4.0 (from qtconsole->jupyter==1.0.0->d2l==1.0.3)
  Downloading QtPy-2.4.1-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipy
Collecting jedi>=0.16 (from ipython>=5.0.0->ipykernel->jupyter==1.0.0->d2l==1.0.3)
  Using cached jedi-0.19.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipykernel-
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipykerne
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipykernel->
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython>=5.0.0->ipykerne
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core>=4.7-
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nbcon
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.10/dist-packages (from terminado>=0.8.3->noteboo
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.10/dist-packages (from argon2-cffi->no
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->nbconv
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert->jupy
```

```
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->nbconvert->jupy
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from jedi>=0.16->ipyth
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbforr
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbfo
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.10/dist-packages (from notebook-shim
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from argon2-cffi-bindings->arg
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0.1->argon2-cffi-bi
Requirement already satisfied: anyio<4,>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from jupyter-server<3,>=1
Requirement already satisfied: websocket-client in /usr/local/lib/python3.10/dist-packages (from jupyter-server<3,>=1
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.1.0->jupyte
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<4,>=3.1.0->jupy
Downloading d2l-1.0.3-py3-none-any.whl (111 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.7/111.7 kB 9.1 MB/s eta 0:00:00
Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Downloading matplotlib-3.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 83.0 MB/s eta 0:00:00
Downloading matplotlib_inline-0.1.6-py3-none-any.whl (9.4 kB)
Downloading numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 83.1 MB/s eta 0:00:00
Downloading pandas-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 85.6 MB/s eta 0:00:00
Downloading requests-2.31.0-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.6/62.6 kB 5.4 MB/s eta 0:00:00
Downloading scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34.4/34.4 MB 19.0 MB/s eta 0:00:00
Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.3/98.3 kB 7.7 MB/s eta 0:00:00
Downloading qtconsole-5.6.0-py3-none-any.whl (124 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.7/124.7 kB 11.2 MB/s eta 0:00:00
Downloading QtPy-2.4.1-py3-none-any.whl (93 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.5/93.5 kB 7.7 MB/s eta 0:00:00
Using cached jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
Installing collected packages: requests, qtpy, pyparsing, numpy, matplotlib-inline, jedi, scipy, pandas, matplotlib,
  Attempting uninstall: requests
    Found existing installation: requests 2.32.3
    Uninstalling requests-2.32.3:
      Successfully uninstalled requests-2.32.3
  Attempting uninstall: pyparsing
    Found existing installation: pyparsing 3.1.4
    Uninstalling pyparsing-3.1.4:
      Successfully uninstalled pyparsing-3.1.4
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
  Attempting uninstall: matplotlib-inline
    Found existing installation: matplotlib-inline 0.1.7
    Uninstalling matplotlib-inline-0.1.7:
      Successfully uninstalled matplotlib-inline-0.1.7
  Attempting uninstall: scipy
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
  Attempting uninstall: pandas
    Found existing installation: pandas 2.1.4
    Uninstalling pandas-2.1.4:
      Successfully uninstalled pandas-2.1.4
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
albucore 0.0.16 requires numpy>=1.24, but you have numpy 1.23.5 which is incompatible.
albumentations 1.4.15 requires numpy>=1.24.4, but you have numpy 1.23.5 which is incompatible.
bigframes 1.17.0 requires numpy>=1.24.0, but you have numpy 1.23.5 which is incompatible.
chex 0.1.86 requires numpy>=1.24.1, but you have numpy 1.23.5 which is incompatible.
google-colab 1.0.0 requires pandas==2.1.4, but you have pandas 2.0.3 which is incompatible.
google-colab 1.0.0 requires requests==2.32.3, but you have requests 2.31.0 which is incompatible.
mizani 0.11.4 requires pandas>=2.1.0, but you have pandas 2.0.3 which is incompatible.
pandas-stubs 2.1.4.231227 requires numpy>=1.26.0; python_version < "3.13", but you have numpy 1.23.5 which is incompa
plotnine 0.13.6 requires pandas<3.0.0,>=2.1.0, but you have pandas 2.0.3 which is incompatible.
xarray 2024.9.0 requires numpy>=1.24, but you have numpy 1.23.5 which is incompatible.
xarray 2024.9.0 requires pandas>=2.1, but you have pandas 2.0.3 which is incompatible.
Successfully installed d2l-1.0.3 jedi-0.19.1 jupyter-1.0.0 matplotlib-3.7.2 matplotlib-inline-0.1.6 numpy-1.23.5 pan
WARNING: The following packages were previously imported in this runtime:
  [matplotlib,matplotlib_inline,mpl_toolkits,numpy]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

2.1 Data Manipulation

2.1.1. Getting Started

```
import torch
```

```
x = torch.arange(12, dtype=torch.float32)
#arange function returns 1-dimensinal tensor of specified range
x #x vectors
```

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

```
x.numel() #number of elements
```

```
12
```

```
x.shape #shape of vector
```

```
torch.Size([12])
```

```
X = x.reshape(3, 4)
X #X is a 3x4 matrix
```

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

```
X.shape #shape of matrix
```

```
torch.Size([3, 4])
```

```
torch.zeros((2,3,4)) #zero matrix, tensor
```

```
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.]]])
```

```
torch.ones((2,3,4)) #one matrix, tensor
```

```
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.]]])
```

```
torch.randn(3,4) #randomly distributed from standard Gaussian (normal) distribution
```

```
tensor([[ 1.1764,  0.0816, -0.8972,  0.5053],
        [-0.0864,  0.4782,  0.0743, -0.2758],
        [ 0.3064, -1.9109, -0.5629,  0.0577]])
```

```
torch.tensor([[2,1,4,3],[1,2,3,4],[4,3,2,1]]) #assign exact values
```

```
tensor([[2, 1, 4, 3],
        [1, 2, 3, 4],
        [4, 3, 2, 1]])
```

2.1.2. Indexing and Slicing (Discussion)

X[start:stop] includes start, but not stop

```
X[-1], X[1:3] #last row, second and third row
```

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

```python
X[1,2] = 17
X #assign value
```

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

```python
X[:2, :] = 12 #row 1,2 and all columns
X
```

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

2.1.3. Operations

```python
torch.exp(x)
```

```
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])
```

```python
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 #elementwise computation
```

```
(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.]))
```

```python
#concatenate multiple tensors
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)
#dim=0 concatenate along rows (stack the rows)
#dim=1 concatenate along columns
```

```
(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.]]))
```

```python
X == Y #same number at [i,j] then True
```

```
tensor([[False,  True, False,  True],
        [False, False, False, False],
        [False, False, False, False]])
```

```python
X.sum() #sum all elements
```

```
tensor(66.)
```

2.1.4. Broadcasting

2.1.4. Discussion

- can use broadcasting when shapes differ
- the tensor with fewer dimensions is padded until dimension matches

- a: (3,1), b: (1,2),

- since a has 1 column, it is streched to match b's columns
- a is stretch as
  [0,0],
  [1,1],
  [2,2]

```
a = torch.arange(3).reshape((3,1)) #3x1 shape
b = torch.arange(2).reshape((1,2)) #1x2 shape
a,b
```

```
(tensor([[0],
         [1],
         [2]]),
 tensor([[0, 1]]))
```

```
a+b
```

```
tensor([[0, 1],
        [1, 2],
        [2, 3]])
```

2.1.5. Saving Memory

```
before = id(Y)
Y = Y+X
id(Y) == before #points to different memory address
```

```
False
```

- we can use in-place operation to save memory

```
Z = torch.zeros_like(Y) #creates tensor X with the same shape as Y
print('id(Z):', id(Z)) #prints address of tensor Z
Z[:] = X + Y #slice all the elements of Z, assign X+Y to this slice
print('id(Z):', id(Z)) #updates the new values directly to Z
```

```
id(Z): 133390961148128
id(Z): 133390961148128
```

```
before = id(X)
X += Y
id(X) == before
```

```
True
```

2.1.6. Conversion to other Python Objects (Discussion)

Python libraries for multi-dimensional arrays (tensors)

- NumPy (ndarray)
- PyTorch (torch.Tensor)

```
A = X.numpy()
B = torch.from_numpy(A)
type(A), type(B)
```

```
(numpy.ndarray, torch.Tensor)
```

```
a = torch.tensor([3.5])
a, a.item(), float(a), int(a) #convert to python scalar
```

```
(tensor([3.5000]), 3.5, 3.5, 3)
```

2.2 Data Preprocessing

2.2.1. Reading the Dataset

```
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''')
```

```
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
```

2.2.2. Data Preparation (Discussion)

- supervised learning: trai models to predict a target value given some input values
- NaN are missing values, which we want to handle it via **imputation** or **deletion**
- iloc: integer-location based indexing
- RoofType originally consists of Slate and NaN
- convert this into 2 columns: RoofType_slate and Rooftype_NaN by True/false
- use pd.get_dummies to convert categorical data into numerical data (0,1)

```
inputs, targets = data.iloc[:, 0:2], data.iloc[:, 2] #target selects third column
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
```

```
inputs = inputs.fillna(inputs.mean()) #replace NaN with mean value
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
```

2.2.3. Conversion to the Tensor Format

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

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

2.2 Discussion

- We can now load datasets into tensors and manipulate them with basic mathematical operations

2.3 Linear Algebra

2.3.1. Scalars (Discussion)

$x \in \mathbb{R}$
x is a real-values scalars

```
import torch
```

```
x = torch.tensor(3.0)
y = torch.tensor(2.0)
x+y, x*y, x/y, x**y
```

```
(tensor(5.), tensor(6.), tensor(1.5000), tensor(9.))
```

2.3.2. Vectors (Discussion)

- real world application: e.g. when training a model to predict the risk of a loean defaulting, each applicant can have a vector whose components correspond to quantities like income, length of employment, etc.

```
x = torch.arange(3)
x
```

```
tensor([0, 1, 2])
```

x_2 denotes second element of x (scalar)

```
x[2]
```

```
tensor(2)
```

```
len(x)
```

```
3
```

```
x.shape
```

```
torch.Size([3])
```

2.3.3. Matrices

```
A = torch.arange(6).reshape(3,2)
A
```

```
tensor([[0, 1],
        [2, 3],
        [4, 5]])
```

```
A.T #transpose
```

```
tensor([[0, 2, 4],
        [1, 3, 5]])
```

```
A = torch.tensor([[1,2,3],[2,0,4],[3,4,5]])
A == A.T
```

```
tensor([[True, True, True],
        [True, True, True],
        [True, True, True]])
```

2.3.4. Tensors (Discussion)

- use tensors for nth-order arrays
- e.g. image can be represented in 3rd-order array: height, width, channel

```
torch.arange(24).reshape(2,3,4)
```

```
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]]])
```

2.3.5. Basic Properties of Tensor Arithmetic

```
A = torch.arange(6, dtype=torch.float32).reshape(2,3)
B = A.clone() #copy A to B by allocating new memory
A, A+B
```

```
(tensor([[0., 1., 2.],
         [3., 4., 5.]]),
 tensor([[ 0.,  2.,  4.],
         [ 6.,  8., 10.]]))
```

- Hadamard product of two matrices

```
A * B
```

```
tensor([[ 0.,  1.,  4.],
        [ 9., 16., 25.]])
```

```
a = 2
X = torch.arange(24).reshape(2,3,4)
a + X, (a * X).shape
```

```
(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]))
```

2.3.6. Reduction

```
x = torch.arange(3, dtype=torch.float32)
x, x.sum() #add all the elements in vector x
```

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

```
A.shape, A.sum() #sum of elements of mxn matrix A
```

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

```
A.shape, A.sum(axis=0).shape #sum along the rows (Axis 0)
```

```
(torch.Size([2, 3]), torch.Size([3]))
```

```
A.shape, A.sum(axis=1).shape #sum along the columns (axis 1)
```

```
(torch.Size([2, 3]), torch.Size([2]))
```

```
A.sum(axis=[0,1]) == A.sum() #same
```

```
tensor(True)
```

```
A.mean(), A.sum()/A.numel() #mean
```

```
(tensor(2.5000), tensor(2.5000))
```

```
A.mean(axis=0), A.sum(axis=0)/A.shape[0]
```

```
(tensor([1.5000, 2.5000, 3.5000]), tensor([1.5000, 2.5000, 3.5000]))
```

2.3.7. Non-Reduction Sum

```
sum_A = A.sum(axis=1, keepdims=True)
sum_A, sum_A.shape
```

```
(tensor([[ 3.],
         [12.]]),
 torch.Size([2, 1]))
```

```
A / sum_A #broadcasting
```

```
tensor([[0.0000, 0.3333, 0.6667],
        [0.2500, 0.3333, 0.4167]])
```

```
A.cumsum(axis=0) #cumulative sum of elements A along axis 0
```

```
tensor([[0., 1., 2.],
        [3., 5., 7.]])
```

2.3.8. Dot Products

```
y = torch.ones(3, dtype = torch.float32)
x, y, torch.dot(x,y)
```

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

```
torch.sum(x*y) #equivalent to dot product
```

```
tensor(3.)
```

2.3.9. Matrix-Vector Products (Discussion)

- torch.mv(A,x): matrix-vector multiplication
- A@x is short-hand python operator for mv

```
A.shape, x.shape, torch.mv(A,x), A@x
```

```
(torch.Size([2, 3]), torch.Size([3]), tensor([ 5., 14.]), tensor([ 5., 14.]))
```

2.3.10. Matrix-Matrix Multiplication

```
B = torch.ones(3,4)
torch.mm(A,B), A@B
```

```
(tensor([[ 3.,  3.,  3.,  3.],
         [12., 12., 12., 12.]]),
 tensor([[ 3.,  3.,  3.,  3.],
         [12., 12., 12., 12.]]))
```

2.3.11. Norms (Discussion)

- norm of a vector tells us how big it is

```
u = torch.tensor([3.0, -4.0])
torch.norm(u)
```

```
tensor(5.)
```

```
torch.abs(u).sum()
```

```
tensor(7.)
```

```
torch.norm(torch.ones((4,9)))
```

```
tensor(6.)
```

2.3 Discussion

- we can now use linear algebra to deal with basic mathematical objects like scalars, vectors, matrices, tensors
- Tensors can be sliced or reduced along axis0/1, via indexing or operations
- Hadamard products
- matrix-matrix products

2.5 Automatic Differentiation

2.5.1. A Simple Function (Discussion)

- goal: differentiate $y = 2x^T x$ with respect to column vector x

$$y = 2\sum_{i=1}^{n} x_i^2 \qquad y = 2x^T x$$

$$x^T x = x_1^2 + x_2^2 + \cdots + x_n^2$$

```
x = torch.arange(4.0)
x
```

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

```
x.requires_grad_(True) #need a place to store gradient of y
# x=torch.arange(4.0, requires_grad=True)로도 가능
x.grad #default value is None
```

```
y = 2 * torch.dot(x,x)
y
```

```
tensor(28., grad_fn=<MulBackward0>)
```

```
y.backward() #take the gradient of y with respect to x
x.grad
```

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

```
x.grad == 4*x
```

```
tensor([True, True, True, True])
```

```
x.grad.zero_()
y=x.sum()
y.backward()
x.grad
```

```
tensor([1., 1., 1., 1.])
```

2.5.2. Backward for Non-Scalar Variables

```
x.grad.zero_()
y = x*x
y.backward(gradient=torch.ones(len(y))) #Faster: y.sum().backward()
x.grad
```

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

2.5.3. Detaching Computation

```
x.grad.zero_()
y = x*x
u = y.detach()
z = u*x

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

```
tensor([True, True, True, True])
```

```
x.grad.zero_()
y.sum().backward()
x.grad == 2*x
```

```
tensor([True, True, True, True])
```

2.5.4. Gradients and Python Control Flow

```
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
```

```
a = torch.randn(size=(), requires_grad=True)
d = f(a)
d.backward()
```

```
a.grad == d/a
```

```
tensor(False)
```

2.5 Discussion

- we can now compute differentiation of functions using python tools
- 1. attach gradients to those variables with respect to which we desire derivaties
- 2. record the computation of the target value
- 3. execute the backpropagation function
- 4. access the resulting gradient
- this can come handy when optimizing models

3.1 Linear Regression

3.1.2. Vectorization for Speed

```
%matplotlib inline
import math
import time
import numpy as np
import torch
from d2l import torch as d2l
```

```
n = 10000 #10000-dimensional vectors
a = torch.ones(n)
b = torch.ones(n)
```

```
c = torch.zeros(n)
t = time.time()
```

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

```
'0.13676 sec'
```

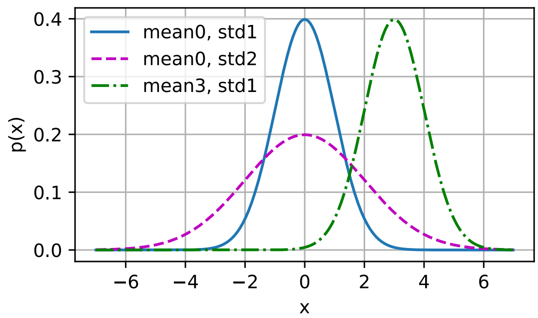
```
t = time.time()
d = a + b
f'{time.time() - t:.5f} sec' #faster method
```

```
'0.00130 sec'
```

3.1.3. The Normal distribution and squared loss

```
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)
```

```
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])
```



- Note that changing the mean corresponds to a shift along the axis, and increasing the variance spreads the distribution out, lowering its peak.

3.1 Discussion

- we now can interpret traditional linear regression, and how linear models could be represented by simple neural networks

3.2 Object-Oriented Design for Implementation

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

3.2.1. Utilities

```
def add_to_class(Class):
  def wrapper(obj):
    setattr(Class, obj.__name__, obj)
  return wrapper
```

```
class A:
  def __init__(self):
    self.b = 1

a = A()
```

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

a.do()
```

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

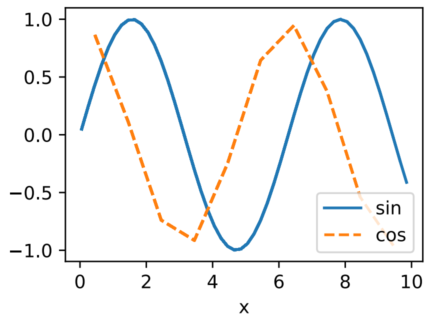
```
class HyperParameters:
  def save_hyperparameters(self, ignore=[]):
    raise NotImplemented
```

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

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

```
self.a = 1 type: <class 'int'>
There is no self.c =  True
```

```
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
```

```
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)
```



3.2.2. Models

```
class Module(nn.Module, d2l.HyperParameters):
    """The base class of models."""
    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):
        """Plot a point in animation."""
        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
```

3.2.3. Data

```
class DataModule(d2l.HyperParameters):
    """The base class of data."""
    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)
```

3.2.4. Training

```
class Trainer(d2l.HyperParameters):
    """The base class for training models with data."""
    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
```

4.1 Softmax Regression (Discussion)

- optimizing over discrete output spaces using probabilistic approach and treating the categories as instances of draws from a probability distribution

4.2 The Image Classification Dataset

```
%matplotlib inline
import time
import torch
import torchvision
from torchvision import transforms
from d2l import torch as d2l

d2l.use_svg_display()
```

4.2.1. Loading the Dataset

```
class FashionMNIST(d2l.DataModule):
    """The Fashion-MNIST dataset."""
    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)
```

```
data = FashionMNIST(resize=(32, 32))
len(data.train), len(data.val)
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz to ../data/Fashion
100%|██████████| 26421880/26421880 [00:07<00:00, 3447323.46it/s]
Extracting ../data/FashionMNIST/raw/train-images-idx3-ubyte.gz to ../data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz to ../data/Fashion
100%|██████████| 29515/29515 [00:00<00:00, 324950.80it/s]
Extracting ../data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to ../data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to ../data/FashionM
100%|██████████| 4422102/4422102 [00:01<00:00, 2909657.39it/s]
Extracting ../data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to ../data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to ../data/FashionM
100%|██████████| 5148/5148 [00:00<00:00, 4871903.65it/s]Extracting ../data/FashionMNIST/raw/t10k-labels-idx1-ubyte.g


(60000, 10000)
```

```
data.train[0][0].shape
```

```
torch.Size([1, 32, 32])
```

```
torch.Size([1, 32, 32])
```

```
torch.Size([1, 32, 32])
```

```
@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]
```

4.2.2. Reading a Minibatch

```
@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)
```

```
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 create
  warnings.warn(_create_warning_msg(
torch.Size([64, 1, 32, 32]) torch.float32 torch.Size([64]) torch.int64
```

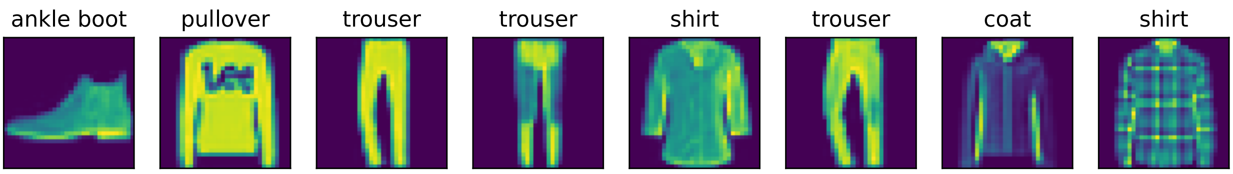
```
tic = time.time()
for X, y in data.train_dataloader():
    continue
f'{time.time() - tic:.2f} sec'
```

```
'15.23 sec'
```

4.2.3. Visualization

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

```
@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)
```



4.2 Discussion

- using the Fashion-MNIST dataset, we have classified into 10 categories
- there is just one channel as the images are grayscale

4.3 The Base Classification Model

4.3.1. The classifier class

```
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)
```

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

4.3.2. Accuracy

```
@d2l.add_to_class(Classifier)
def accuracy(self, Y_hat, Y, averaged=True):
    """Compute the number of correct predictions."""
    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

4.4.1 .The Softmax

```
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)
```

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

```
def softmax(X):
    X_exp = torch.exp(X)
    partition = X_exp.sum(1, keepdims=True)
    return X_exp / partition  # The broadcasting mechanism is applied here
```

```
X = torch.rand((2, 5))
X_prob = softmax(X)
X_prob, X_prob.sum(1)
```

```
(tensor([[0.1647, 0.1440, 0.1487, 0.3321, 0.2106],
         [0.1333, 0.1750, 0.2688, 0.1655, 0.2574]]),
 tensor([1.0000, 1.0000]))
```

4.4.2. The Model

```
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]
```

```
@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)
```

4.4.3. The Cross-Entropy Loss

```
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]
```

```
tensor([0.1000, 0.5000])
```

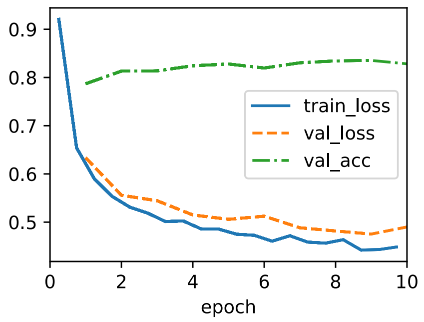
```
def cross_entropy(y_hat, y):
    return -torch.log(y_hat[list(range(len(y_hat))), y]).mean()

cross_entropy(y_hat, y)
```

```
tensor(1.4979)
```

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

4.4.4. training

```
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)
```

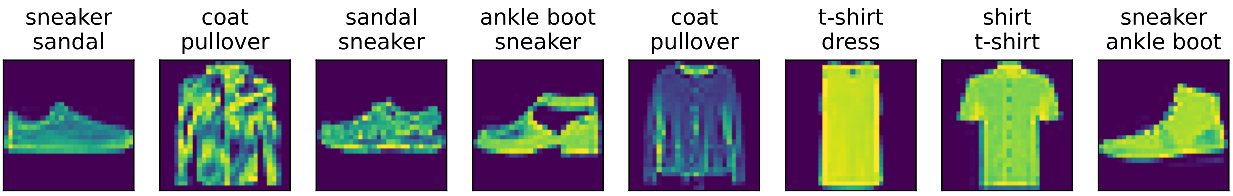


4.4.5. Prediction

```
X, y = next(iter(data.val_dataloader()))
preds = model(X).argmax(axis=1)
preds.shape
```

```
torch.Size([256])
```

```
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)
```



4.4 Discussion

- we can adjust the hyperparamters such as epochs, minibatch size, and learning rate to train the model

5.1 Multilayer Perceptrons

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

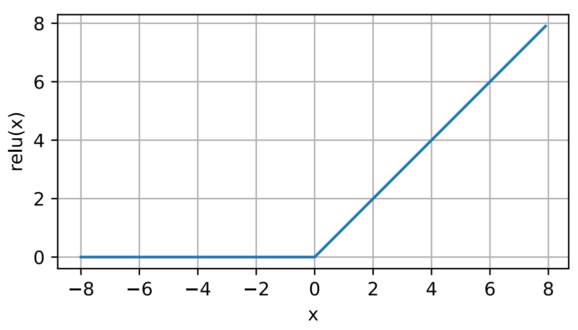
5.1.1.4. Universal Approximators (discussion)

- Even with single-hidden-layer network, given enough nodes (possible absurdly many), and the right set of weights, we can model any function
- kernel methods are way more effective, since they are capable of solving the problem exactly even in **infinite dimesional spaces**(=universal approximator theorem)
- we can approximate many functions much more compactly by using deeper (rather than sider) networks

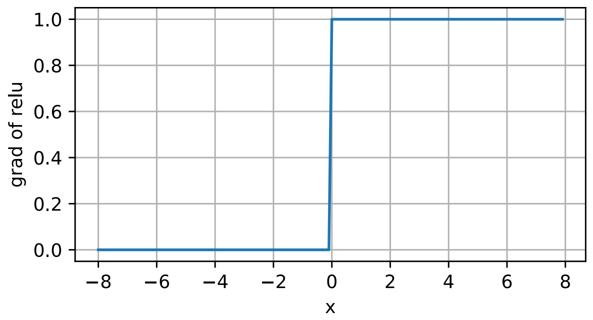
5.1.2.1. ReLU Function (Discussion)

$$\mathrm{ReLU}(x) = \max(x, 0)$$

```python
x = torch.arange(-8.0, 8.0, 0.1, requires_grad=True)
y = torch.relu(x) #output of relu
d2l.plot(x.detach(), y.detach(),'x','relu(x)', figsize=(5,2.5))
```




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

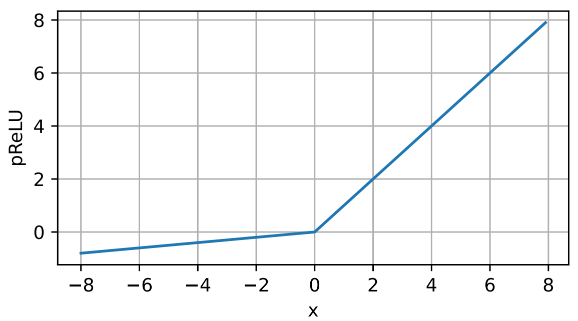


5.1.2.1. Exercise / discussion

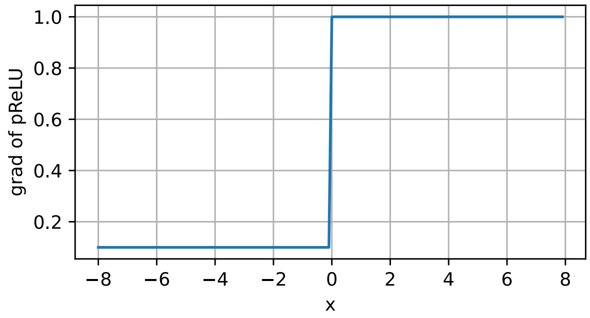
$$\text{pReLU}(x) = \max(0, x) + \alpha \min(0, x)$$

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

```python
y = pReLU(x=x, a=0.1)

d2l.plot(x.detach(), y.detach(), 'x', 'pReLU', figsize=(5, 2.5))
```

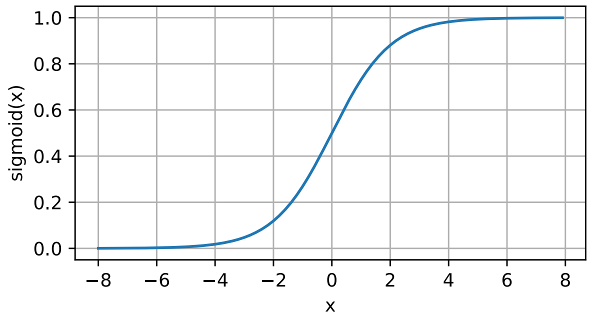



```python
x.grad.data.zero_() #
y.backward(torch.ones_like(x), retain_graph=True)
d2l.plot(x.detach(), x.grad, 'x', 'grad of pReLU', figsize=(5, 2.5))
```

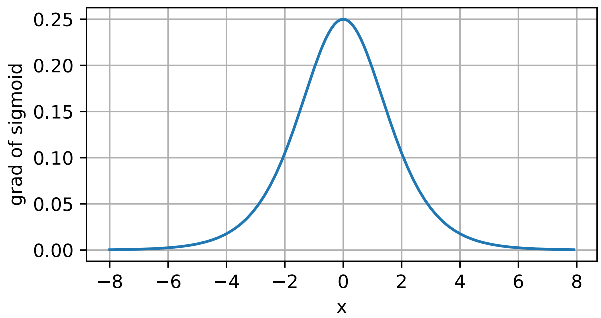


5.1.2.2. Sigmoid Function

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




```
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))
```



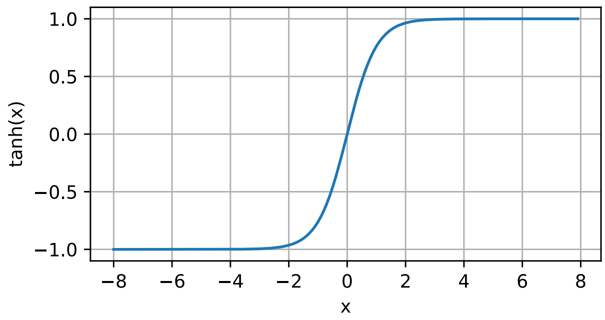
- Gradient Vanishing Problem ~ Backpropagation / Activation functions

5.1.2.3. Tanh Function

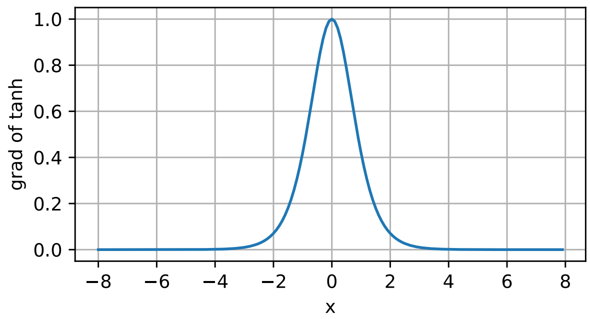
5.1.2.3. Tanh Discussion

- $\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)} = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$

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




```
# Clear out previous gradients
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))
```



5.1 Discussion

- we can incorporate nonlinear models to build more multilayer neural network architectures

5.2 Implementation of Multilayer Perceptrons

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

5.2.1. Implementation from Scratch

5.2.1.1. Initializing Model Parameters (Discussion)

- Fashion-MNIST has 10 classes, each image with 28x28 = 784 grid of grayscale pixel values
- This means: 784 input features & 10 classes
- Implement 1 hidden layer and 256 hidden units

```
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))
```

5.2.1.2. Model (Discussion)

- Compared to softmax regression, this has 2 fully connected layers (one hidden layer, one output layer)

```
def relu(X):
    a = torch.zeros_like(X)
    return torch.max(X, a)
```

```
@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
```

5.2.1.3. Training