

Lab1. PyTorch Basics

TA: 李臻欣

23210240025@m.fudan.edu.cn

Outline

- Background: What is PyTorch?
- Tensors
- Training & Testing a Classification Model in Pytorch
 - Dataset & Dataloader
 - torch.nn: Models & Loss Functions
 - torch.optim: Optimization Process

Goals of this Lab

- Learn the basic concepts of PyTorch
- Tensors
 - Get proficient at handling PyTorch Tensors
- Training & Testing Models in PyTorch
 - Know how a Logistic Regression model is trained and evaluated
- Work with Kaggle Notebooks

Prerequisites

- We assume you are already familiar with...

1. Python3

- if-else, loop, function, file IO, class, ...
- refs: [link1](#), [link2](#), [link3](#)

2. Linear Classifiers

- Logistic Regression
- Loss / Optimization



Some knowledge of **NumPy** will also be useful!

What is PyTorch?

- An **machine learning framework** in Python.
- Two main features:
 - N-dimensional **Tensor** computation (like NumPy) on **GPUs**
 - **Automatic differentiation** for training deep neural networks

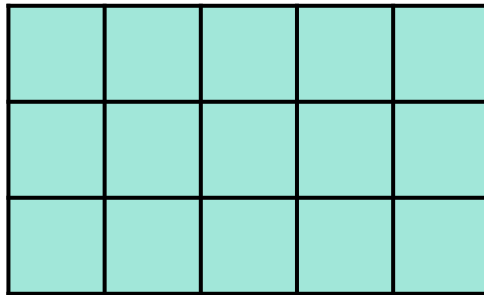


Tensors

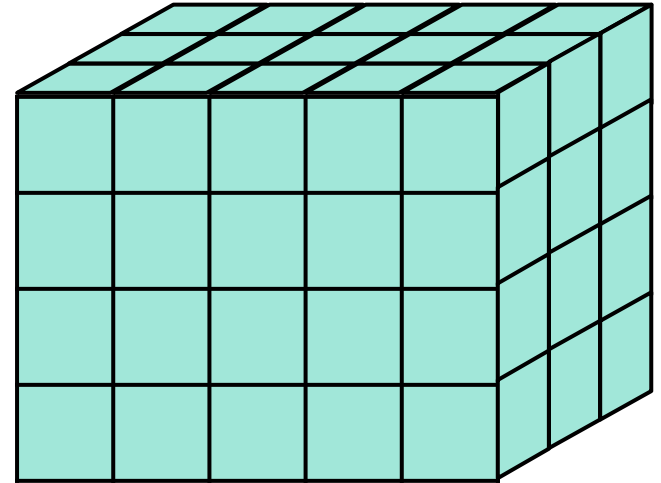
- High-dimensional matrices (arrays)



1-D tensor
e.g. audio



2-D tensor
e.g. black&white
images



3-D tensor
e.g. RGB images

Tensors – Shape of Tensors



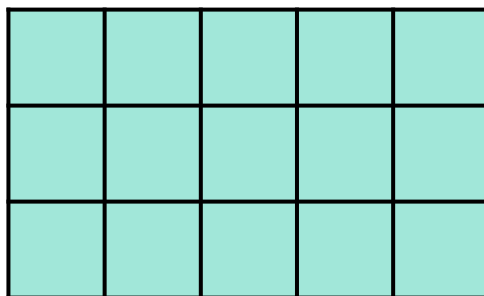
5

(5,)



dim 0

3



5

(3, 5)

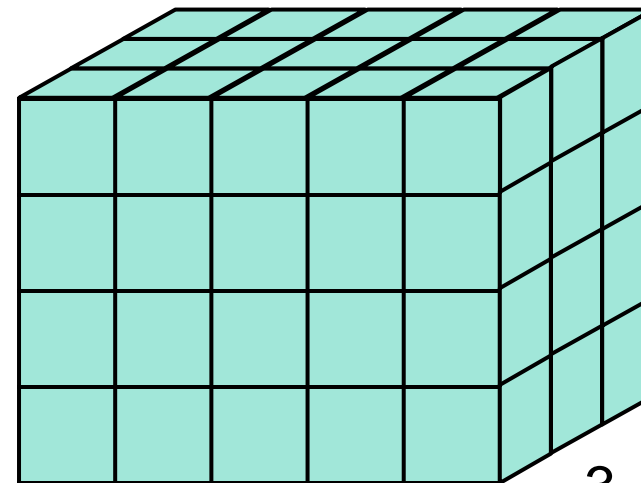


dim 0



dim 1

4



3

5

(4, 5, 3)



dim 0



dim 1



dim 2

Note: **dim** in PyTorch == **axis** in NumPy

Tensors – Creating Tensors

- Directly from data (list or numpy.ndarray)

```
x = torch.tensor([[1, -1], [-1, 1]])
```

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

```
x = torch.from_numpy(np.array([[1, -1], [-1, 1]]))
```

- Tensor of constant zeros & ones

```
x = torch.zeros([2, 2])
```

```
x = torch.ones([1, 2, 5])
```

shape



```
tensor([[0., 0.],  
        [0., 0.]])
```

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


Tensors – Common Operations

Common arithmetic functions are supported, such as:

- Addition

$$z = x + y$$

- Subtraction

$$z = x - y$$

- Power

$$y = x.\text{pow}(2)$$

- Summation

$$y = x.\text{sum}()$$

- Mean

$$y = x.\text{mean}()$$

Tensors – Common Operations

- **Transpose:** transpose two specified dimensions

```
>>> x = torch.zeros([2, 3])
```

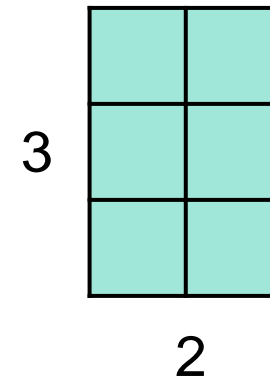
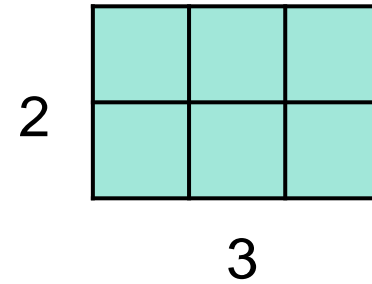
```
>>> x.shape
```

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

```
>>> x = x.transpose(0, 1)
```

```
>>> x.shape
```

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



Tensors – Common Operations

- **Squeeze:** remove the specified dimension with length = 1

```
>>> x = torch.zeros([1, 2, 3])
```

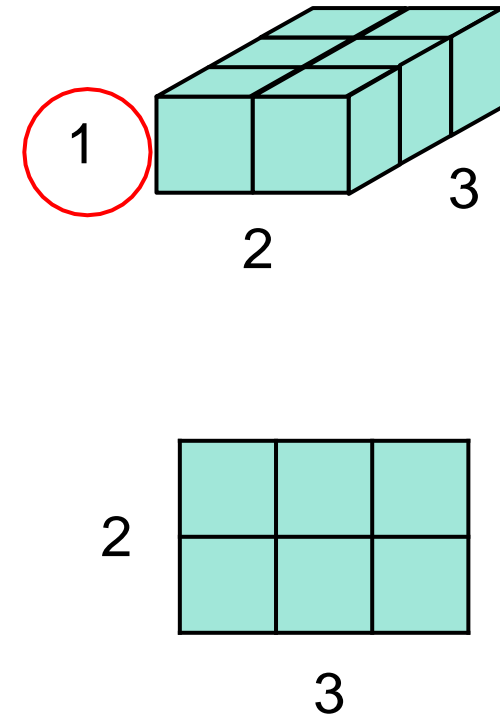
```
>>> x.shape
```

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

```
>>> x = x.squeeze(0)  
                        (dim = 0)
```

```
>>> x.shape
```

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



Tensors – Common Operations

- **Unsqueeze:** expand a new dimension

```
>>> x = torch.zeros([2, 3])
```

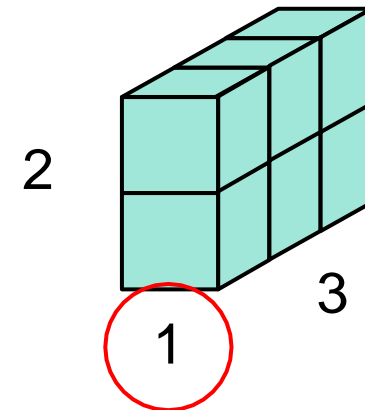
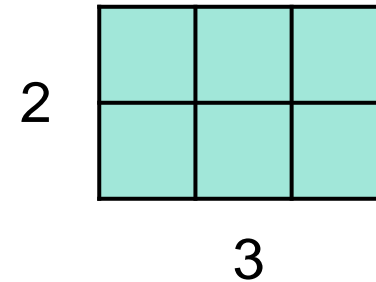
```
>>> x.shape
```

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

```
>>> x = x. unsqueeze(1) (dim = 1)
```

```
>>> x.shape
```

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



Tensors – Common Operations

- **Cat:** concatenate multiple tensors

```
>>> x = torch.zeros([2, 1, 3])
```

```
>>> y = torch.zeros([2, 3, 3])
```

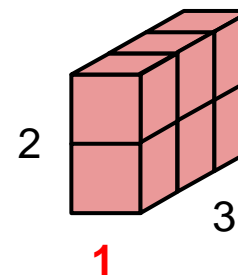
```
>>> z = torch.zeros([2, 2, 3])
```

```
>>> w = torch.cat([x, y, z], dim=1)
```

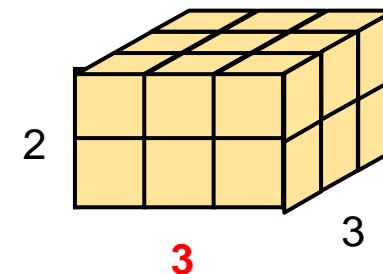
```
>>> w.shape
```

```
torch.Size([2, 6, 3])
```

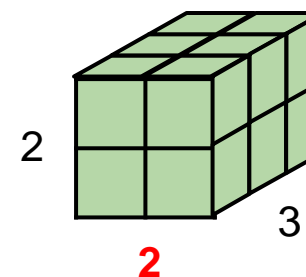
more operators: <https://pytorch.org/docs/stable/tensors.html>



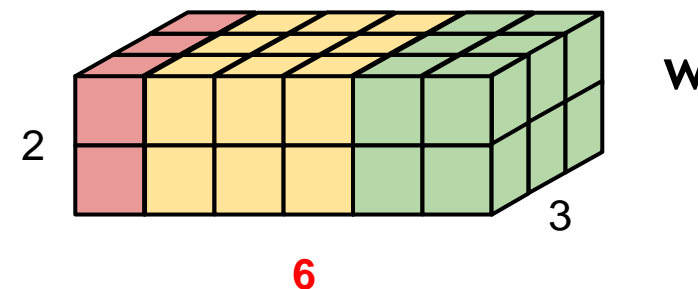
x



y



z



w

Tensors – Device

- Tensors & modules will be computed with **CPU** by default

Use `.to()` to move tensors to appropriate devices.

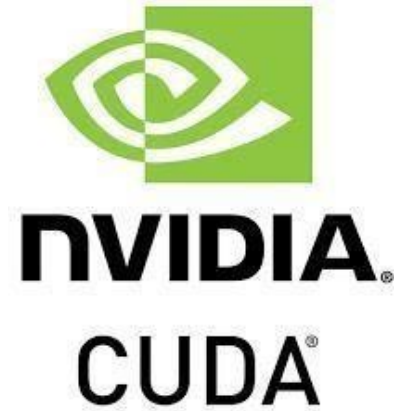
- CPU

```
x = x.to('cpu')
```

- GPU

```
x = x.to('cuda')
```

Tensors – Device (GPU)



- Check if your computer has NVIDIA GPU

`torch.cuda.is_available()`

- Multiple GPUs: specify

`'cuda:0' , 'cuda:1' , 'cuda:2' , ...`

- Why use GPUs?

- Parallel computing with more cores for arithmetic calculations
- See [What is a GPU and do you need one in deep learning?](#)

Tensors – Gradient Calculation

1 >>> x = torch.tensor([[1., 0.], [-1., 1.]], **requires_grad=True**)

2 >>> z = x.pow(2).sum()

3 >>> z.**backward()**

4 >>> x.**grad**

tensor([[2., 0.],
 [-2., 2.]])

1
$$x = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

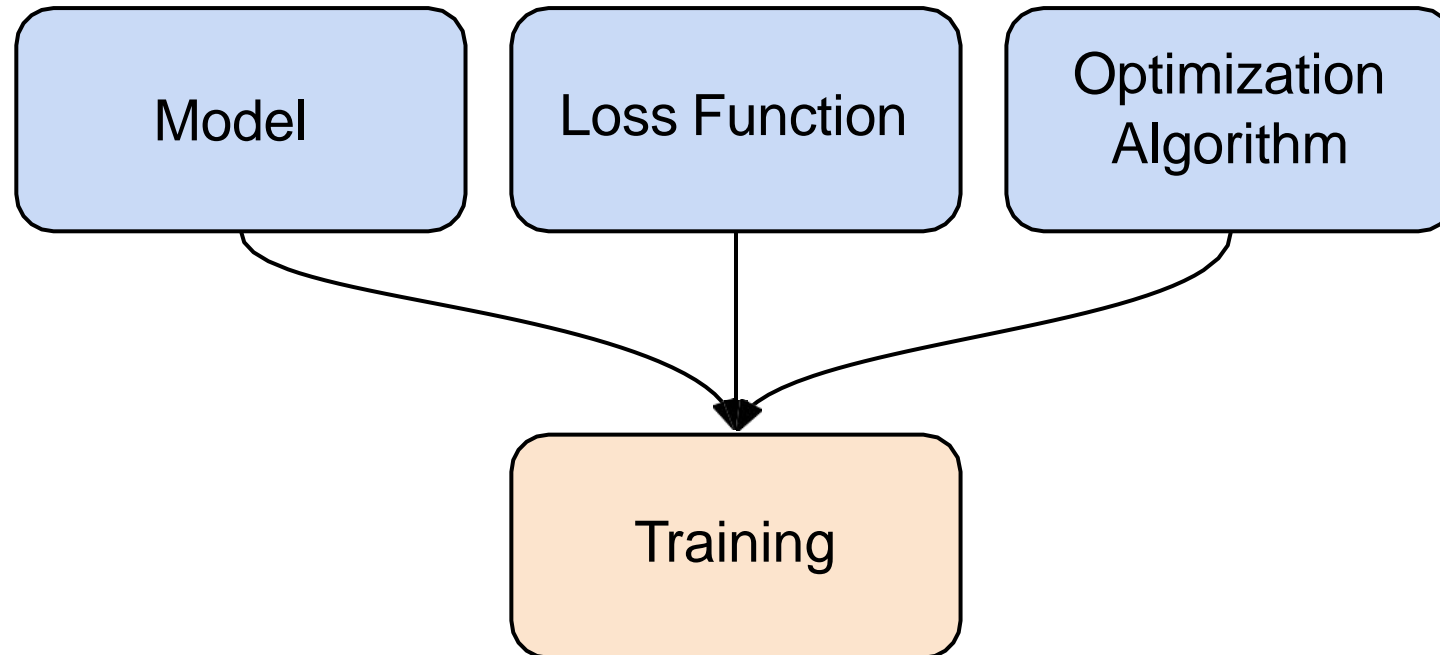
2
$$z = \sum_i \sum_j x_{i,j}^2$$

3
$$\frac{\partial z}{\partial x_{i,j}} = 2x_{i,j}$$

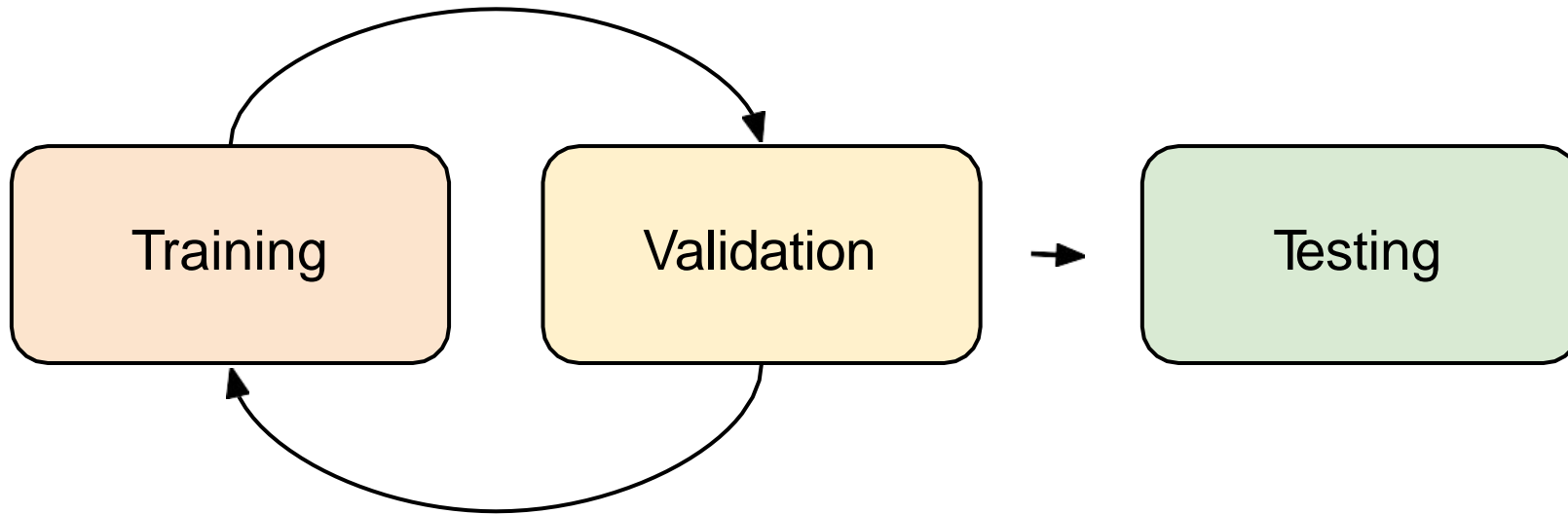
4
$$\frac{\partial z}{\partial x} = \begin{bmatrix} 2 & 0 \\ -2 & 2 \end{bmatrix}$$

See [here](#) to learn about gradient calculation.

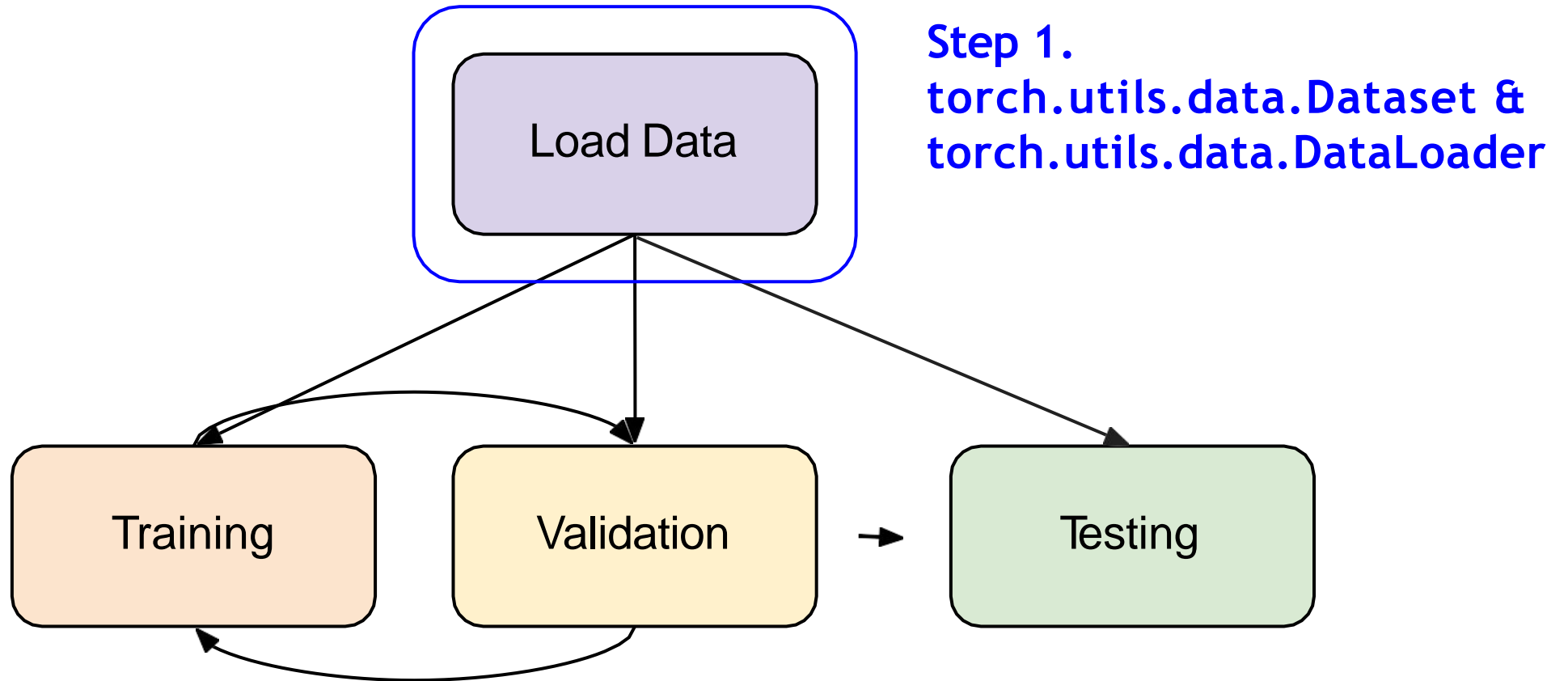
Training a Model



Training & Testing Models



Training & Testing Models



Dataset & Dataloader

- Dataset: stores data samples and expected values
- Dataloader: groups data in batches, enables multiprocessing
- `dataset = MyDataset(file)`
- `dataloader = DataLoader(dataset, batch_size, shuffle=True)`



Training: True
Testing: False

More info about batches and shuffling [here](#).

Dataset & Dataloader

```
from torch.utils.data import Dataset, DataLoader
```

```
class MyDataset(Dataset):  
    def init (self, file):  
        self.data = ...
```



Read data & preprocess

```
def __getitem__(self, index):  
    return self.data[index]
```



Returns one sample at a time

```
def __len__(self):  
    return len(self.data)
```

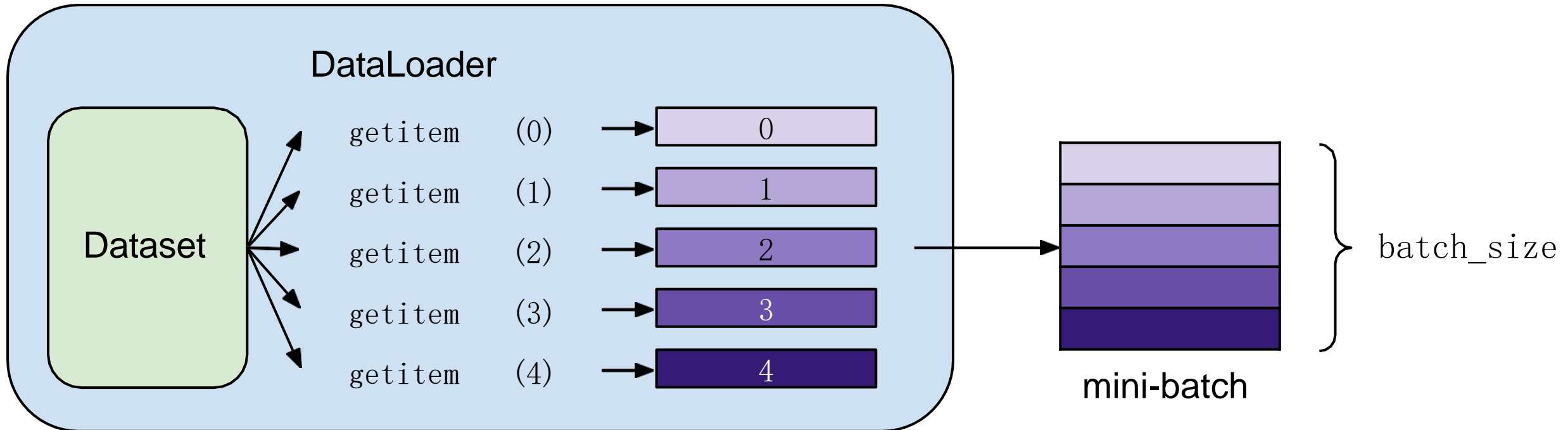


Returns the size of the dataset

Dataset & Dataloader

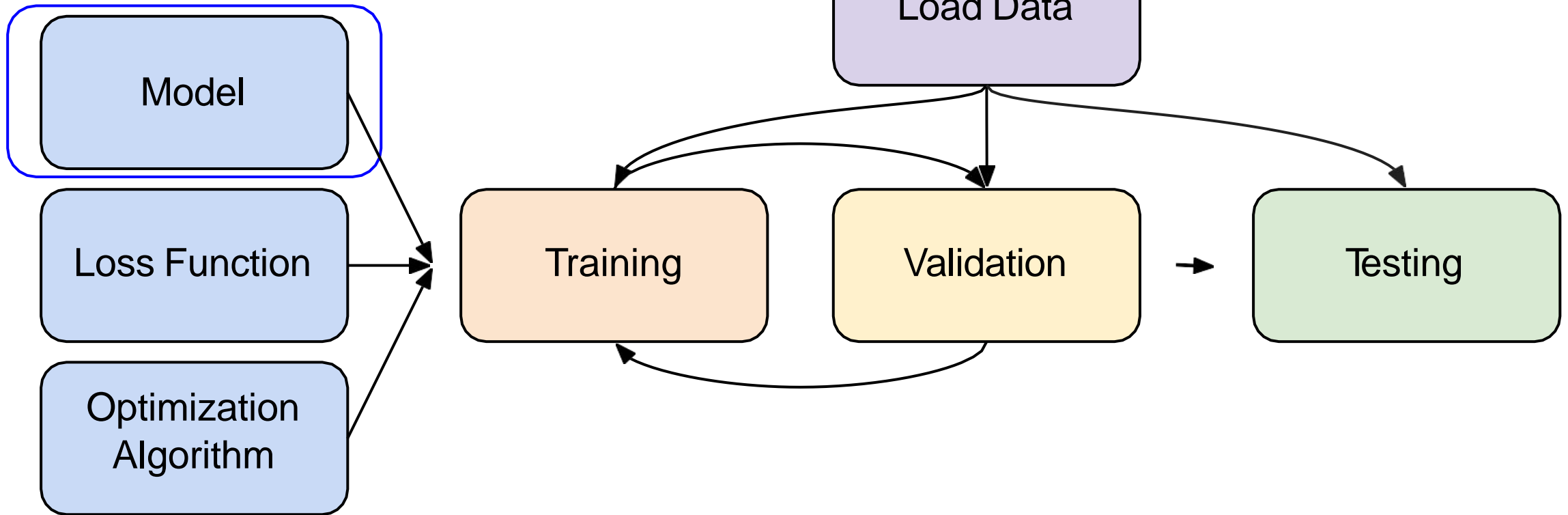
```
dataset = MyDataset(file)
```

```
dataloader = DataLoader(dataset, batch_size=5, shuffle=False)
```



Training & Testing Models

Step 2.
`torch.nn.Module`

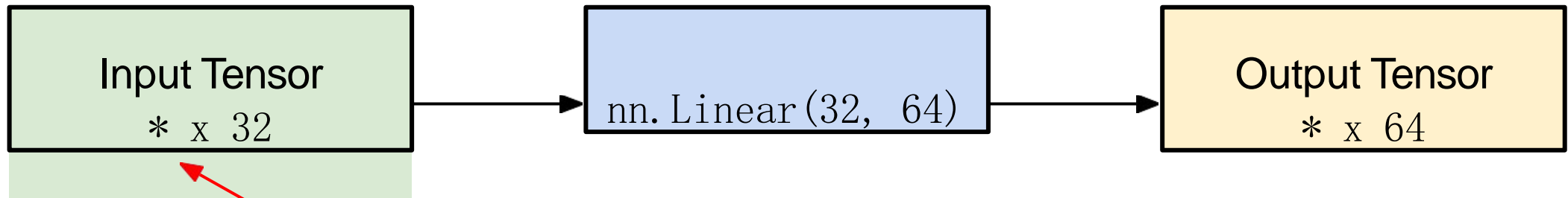


torch.nn – Network Layers

- Linear Layer (**Fully-connected** Layer)

`nn.Linear(in_features, out_features)`

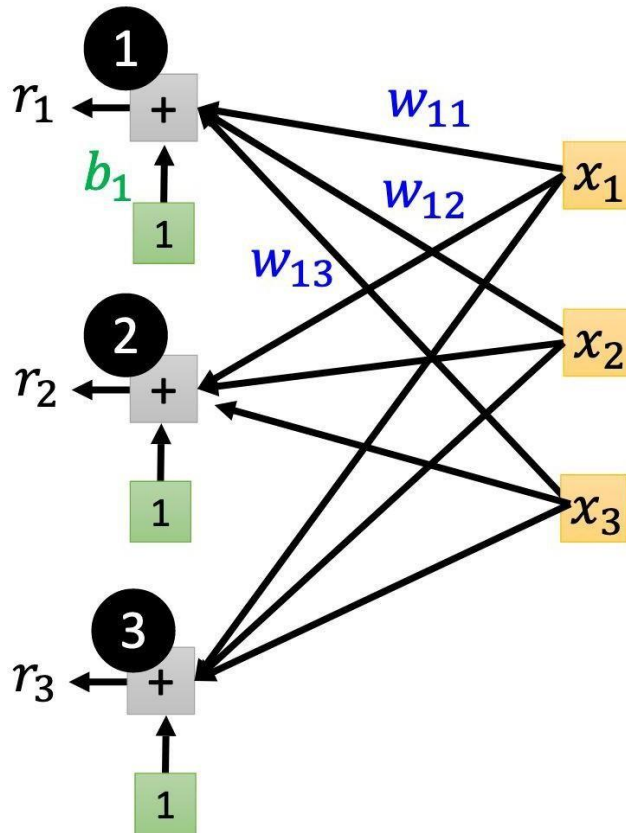
- We use this for our Logistic Regression Model



can be any shape (but last dimension must be 32)
e.g. (10, 32), (10, 5, 32), (1, 1, 3, 32), ...

torch.nn – Network Layers

- Linear Layer (**Fully-connected** Layer)

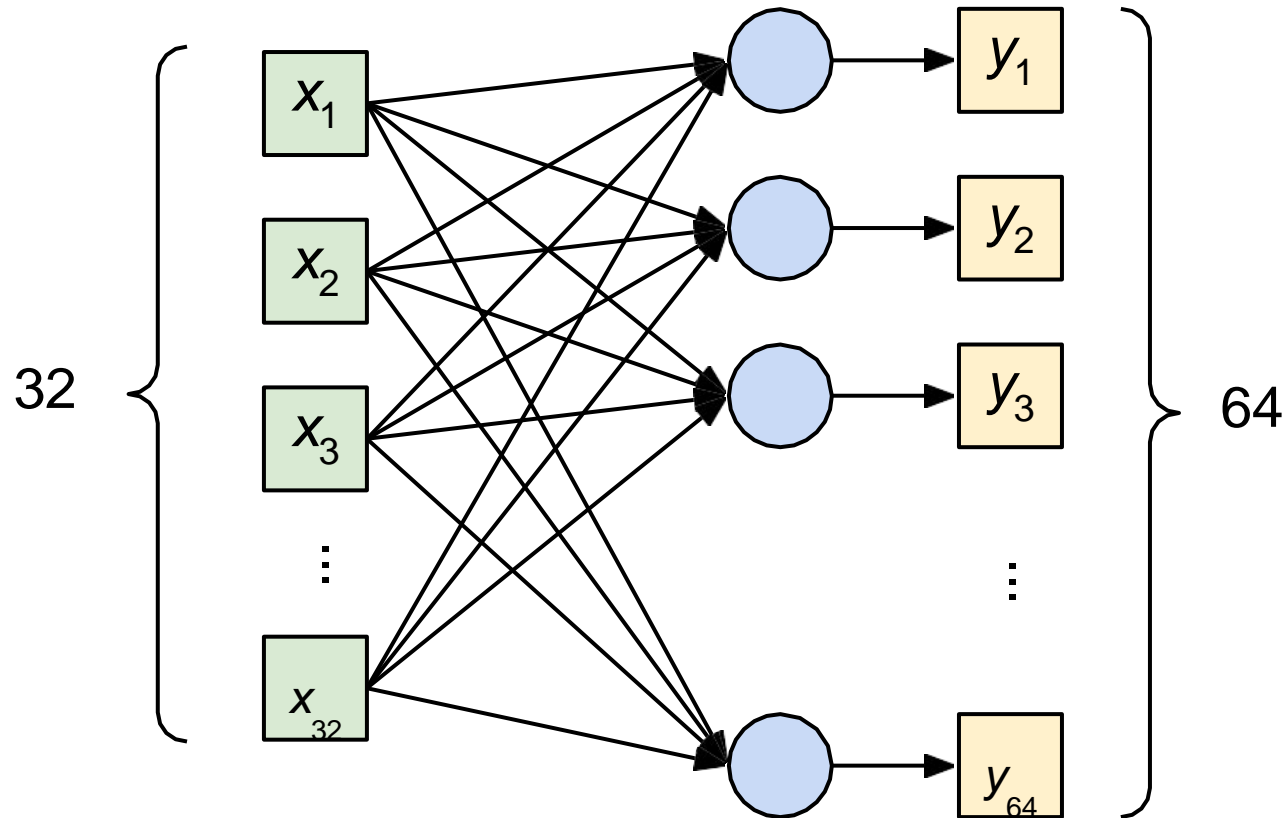


$$b + Wx$$

ref: [last year's lecture video](#)

torch.nn – Network Layers

- Linear Layer (**Fully-connected** Layer)



$$\begin{matrix} \text{[Blue Box]} \\ \mathbf{W} \\ (64 \times 32) \end{matrix} \times \begin{matrix} \text{[Green Box]} \\ \mathbf{x} \end{matrix} + \begin{matrix} \text{[Blue Box]} \\ \mathbf{b} \end{matrix} = \begin{matrix} \text{[Yellow Box]} \\ \mathbf{y} \end{matrix}$$

torch.nn – Network Parameters

- Linear Layer (**Fully-connected** Layer)

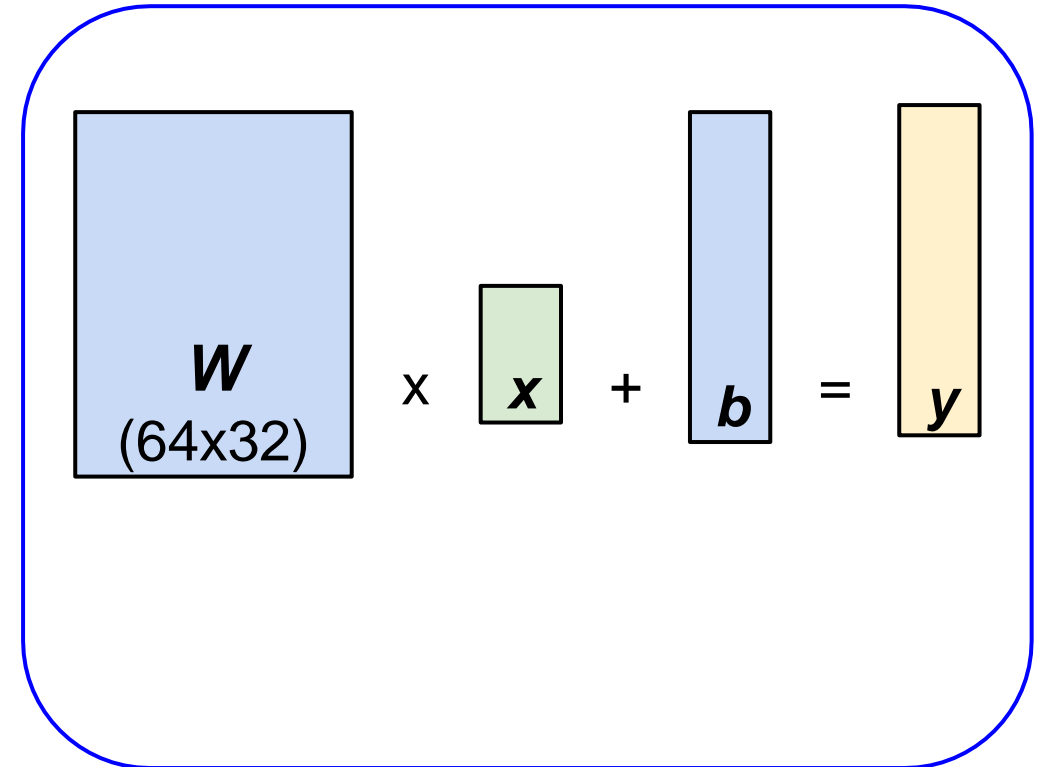
```
>>> layer = torch.nn.Linear(32, 64)
```

```
>>> layer.weight.shape
```

```
torch.Size([64, 32])
```

```
>>> layer.bias.shape
```

```
torch.Size([64])
```



torch.nn – Non-Linear Activation Functions

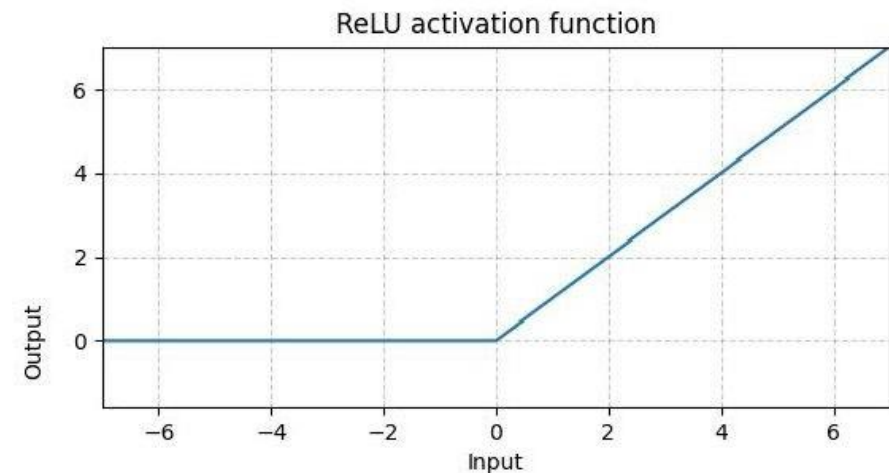
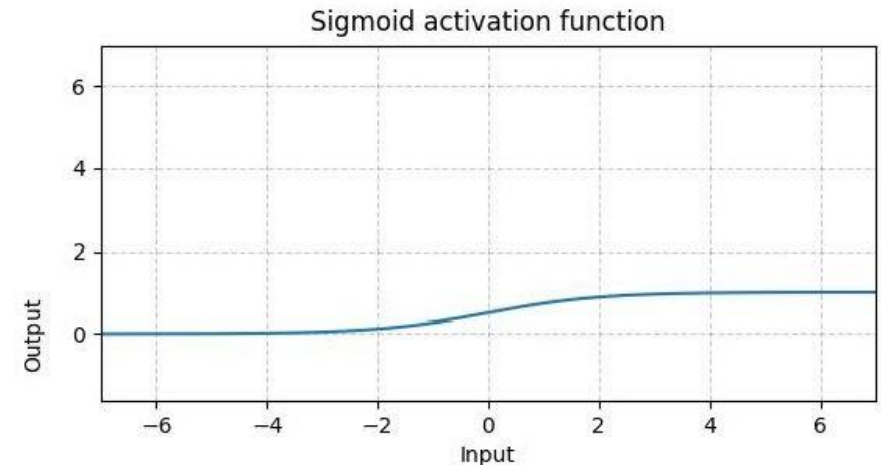
- These are for more complex Neural Networks, which will be introduced in the future lessons.
- Sigmoid Activation

`nn.Sigmoid()`

- ReLU Activation

`nn.ReLU()`

See [here](#) to learn about why we need activation functions.

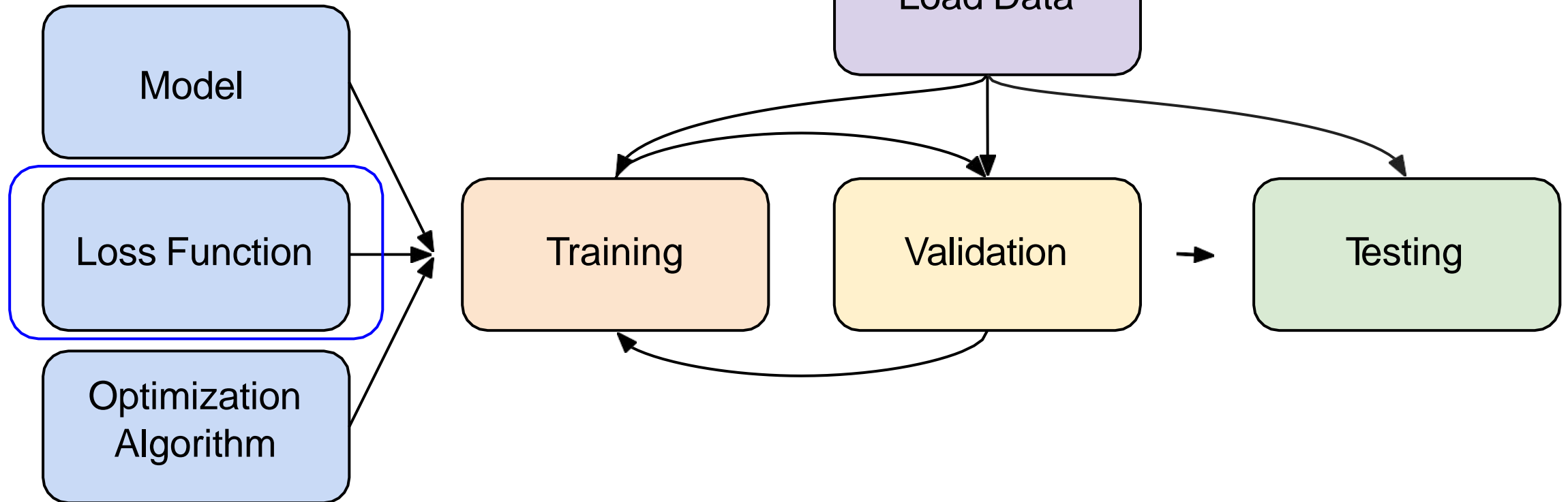


Training & Testing Models

Step 3.

`torch.nn.MSELoss`

`torch.nn.CrossEntropyLoss` etc.



torch.nn – Loss Functions

- Mean Squared Error (for regression tasks)

```
criterion = nn.MSELoss()
```

- Cross Entropy (for classification tasks, a Multi-class Version of Logistic Regression)

```
criterion = nn.CrossEntropyLoss()
```

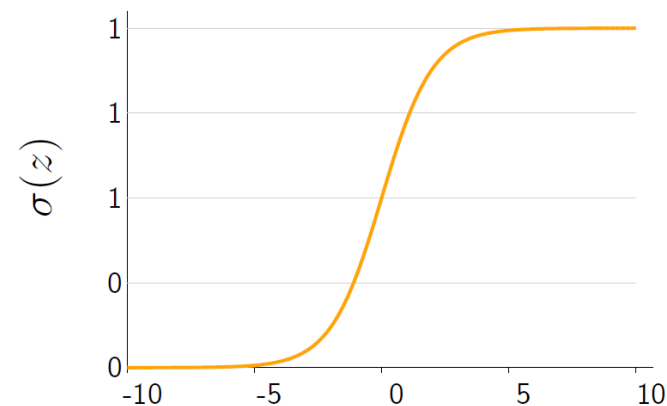
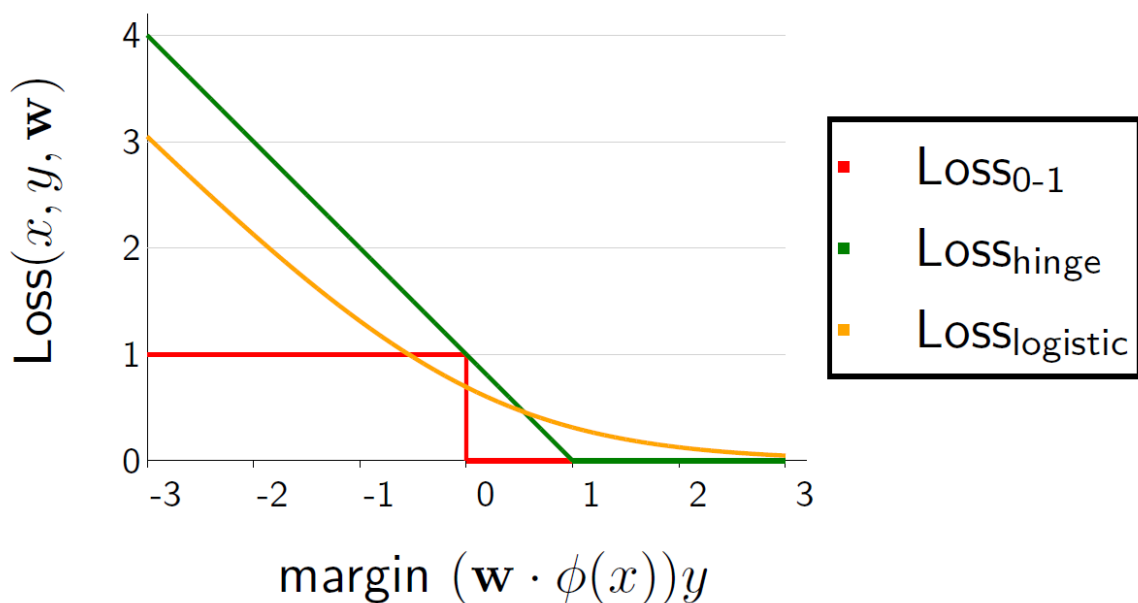
- `loss = criterion(model_output, expected_value)`

Logistic Regression – For 2 classes (binary classification)

逻辑回归

逻辑回归的另外一个角度

$$\text{LOSS}_{\text{logistic}}(x, y, \mathbf{w}) = \log(1 + e^{-(\mathbf{w} \cdot \phi(x))y})$$



Logistic: $\frac{1}{1+e^{-z}}$

Logistic Regression, Softmax, Cross Entropy

- Logistic Regression
- Softmax

$$\text{softmax}(\mathbf{v}) = \frac{1}{\sum_{k=1}^K e^{v_k}} \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_K} \end{bmatrix}.$$

Equivalent to the multi-class version of Log. Func.

Logistic: $\frac{1}{1+e^{-z}}$

- Cross Entropy Loss – the multi-class version of Log. Reg. $H(\mathbf{P}, \mathbf{Q}) = - \sum_{i=1}^K p_i \log(q_i),$

`loss = nn.CrossEntropyLoss(model_output, expected_value)`

`model_output: (2.0, 3.0, 0.0)`

Calculate $H(\mathbf{P}, \mathbf{Q})$

\mathbf{P} : one-hot
ground truth
labels like
(0, 1, 0)

Consider 3 classes

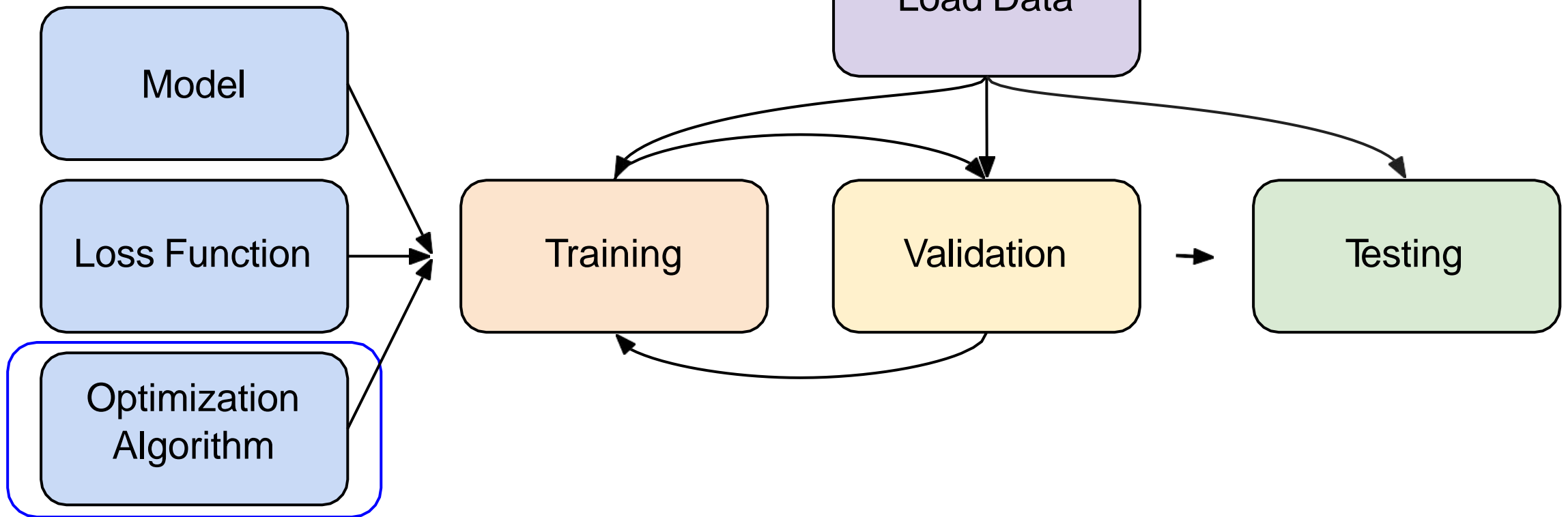
Softmax

\mathbf{Q} : (0.26, 0.71, 0.04)

For Detailed Explanation: [here](#)

Training & Testing Models

Step 4.
`torch.optim`



torch.optim

- Gradient-based **optimization algorithms** that adjust network parameters to reduce error. (See [Adaptive Learning Rate](#) lecture video)
- E.g. Stochastic Gradient Descent (SGD)

```
torch.optim.SGD(model.parameters(), lr, momentum = 0)
```

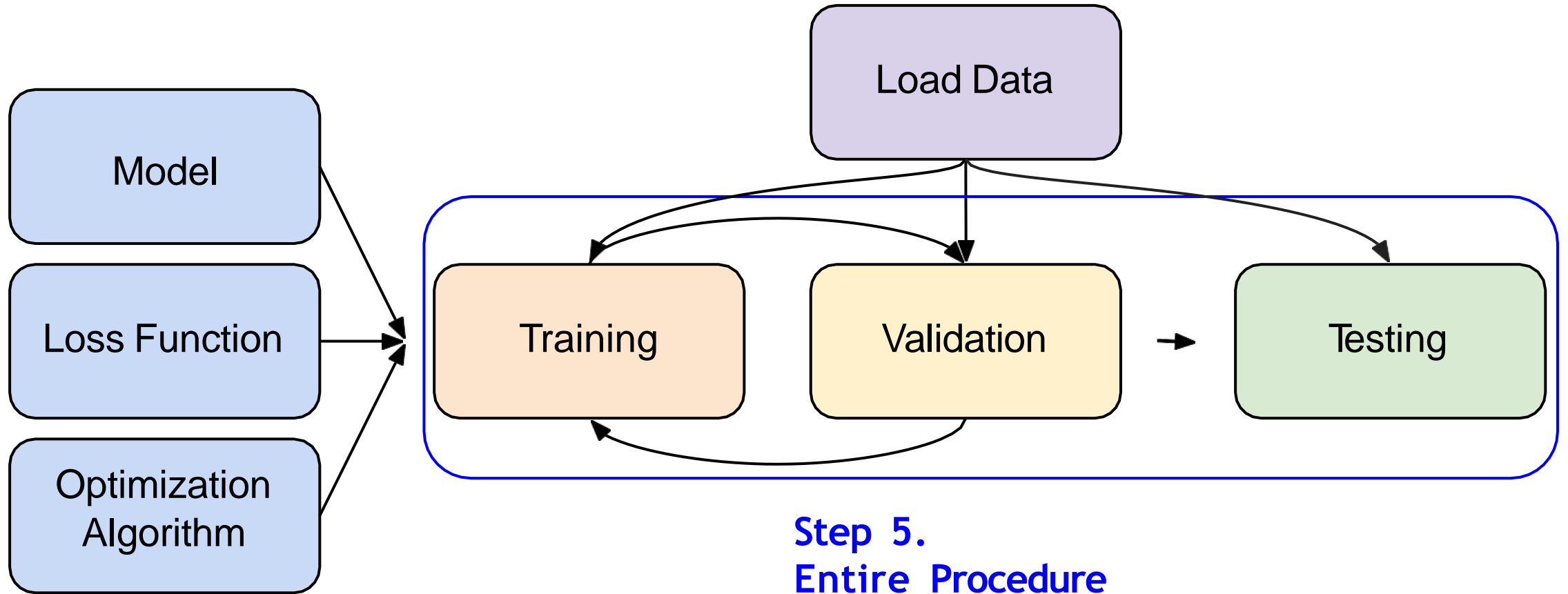
torch.optim

```
optimizer = torch.optim.SGD(model.parameters(), lr, momentum = 0)
```

- For every batch of data:
 1. Call `optimizer.zero_grad()` to reset gradients of model parameters.
 2. Call `loss.backward()` to backpropagate gradients of prediction loss.
 3. Call `optimizer.step()` to adjust model parameters.

See [official documentation](#) for more optimization algorithms.

Training & Testing Models



Training Setup – Linear Regression

X (input): 1, 2, ..., 10
Y (output): 2, 4, ..., 20

`dataset = MyDataset(file)`

read data via MyDataset

`tr_set = DataLoader(dataset, batch_size=2, shuffle=True)`

put dataset into Dataloader

`model = MyModel().to(device)`

construct model and move to device (cpu/cuda)

`criterion = nn.MSELoss()`

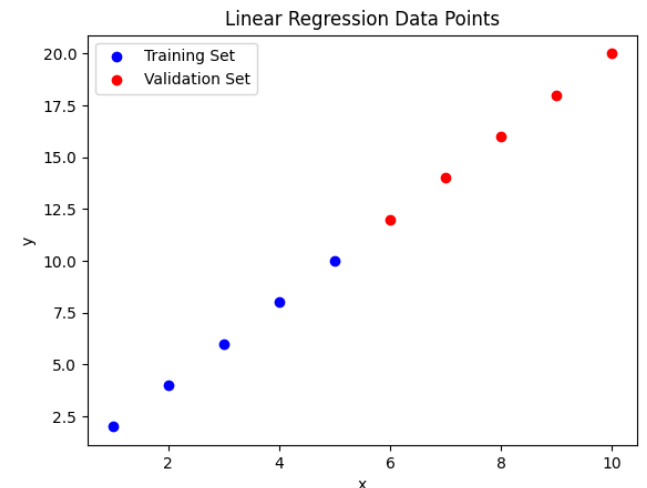
set loss function

`optimizer = torch.optim.SGD(model.parameters(), 0.1)`

set optimizer

Linear(1, 1): w & b

Give model an x, output is $wX+b$



Training Loop – Linear Regression

- `for epoch in range(n_epochs):`

- `model.train()`

- `for x, y in tr_set:`

- `optimizer.zero_grad()`

- `x, y = x.to(device), y.to(device)`

- `pred = model(x)`

- `loss = criterion(pred, y)`

- `loss.backward()`

- `optimizer.step()`

iterate `n_epochs`

set model to train mode

iterate through the dataloader

set gradient to zero

move data to device (cpu/cuda)

forward pass (compute output)

compute loss

compute gradient (backpropagation)

update model with optimizer

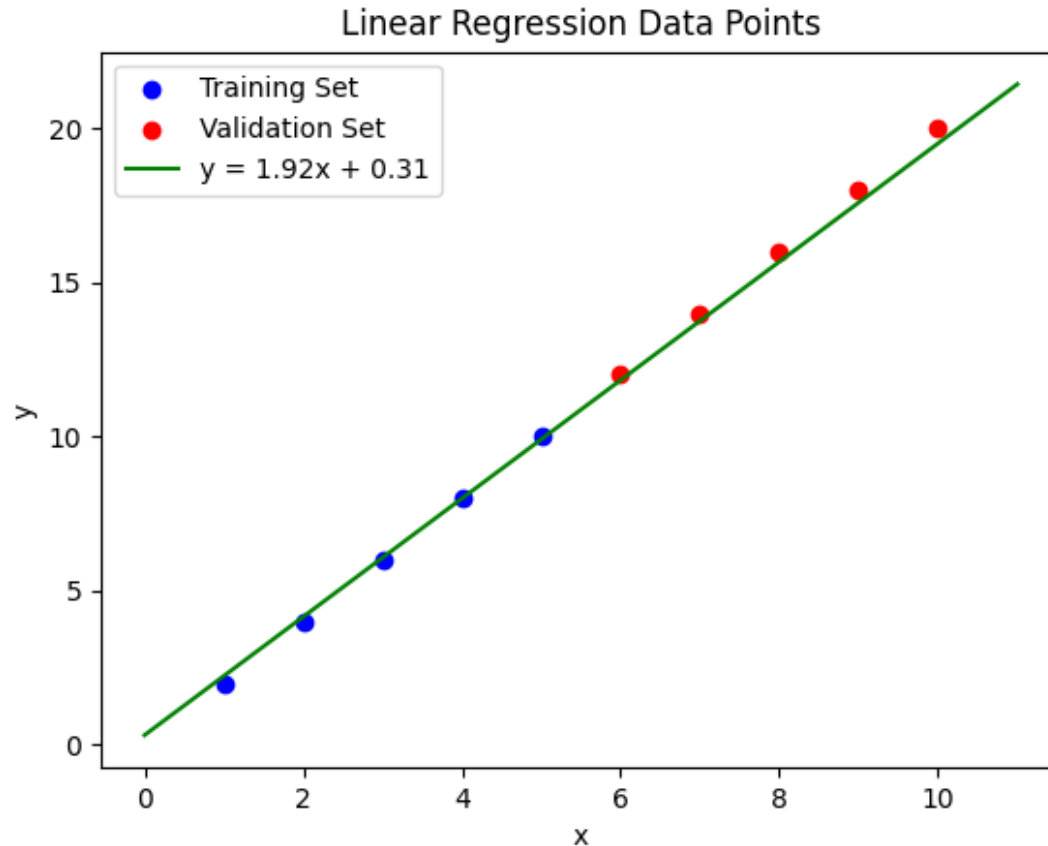
$wX + b$

$\|wX + b - Y\|_2$

We use `MSELoss` here
for regression.

Training Loop – Linear Regression

- After training, we get the model: $w=1.92$, $b=0.31$



Validation / Testing Loop

model.eval()

set model to evaluation mode

```
preds = []
```

```
for x in tt_set:
```

iterate through the dataloader

```
    x = x.to(device)
```

move data to device (cpu/cuda)

with torch.no_grad():

disable gradient calculation

```
    pred = model(x)
```

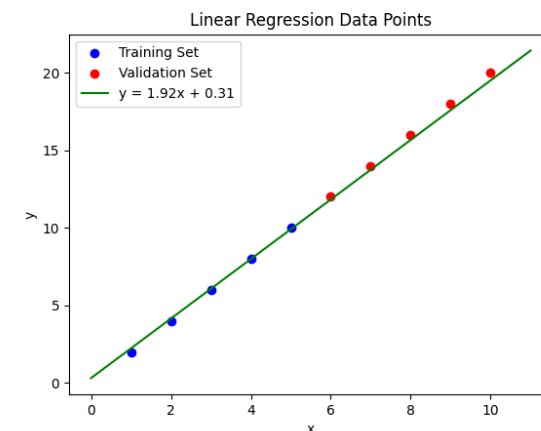
forward pass (compute output)

```
    preds.append(pred.cpu())
```

collect prediction

$1.92X + 0.31$

Preds is [11.8, 13.8, 15.7, 17.6, 19.5]
Y is [12, 14, 16, 18, 20]



Notice - `model.eval()`, `torch.no_grad()`

- **`model.eval()`**
Changes behaviour of some model layers, such as dropout and batch normalization.
- **`with torch.no_grad()`**
Prevents calculations from being added into gradient computation graph.
Usually used to prevent accidental training on validation/testing data.

Save / Load Trained Models

- Save

```
torch.save(model.state_dict(), path)
```

- Load

```
ckpt = torch.load(path)
```

```
model.load_state_dict(ckpt)
```

More About PyTorch

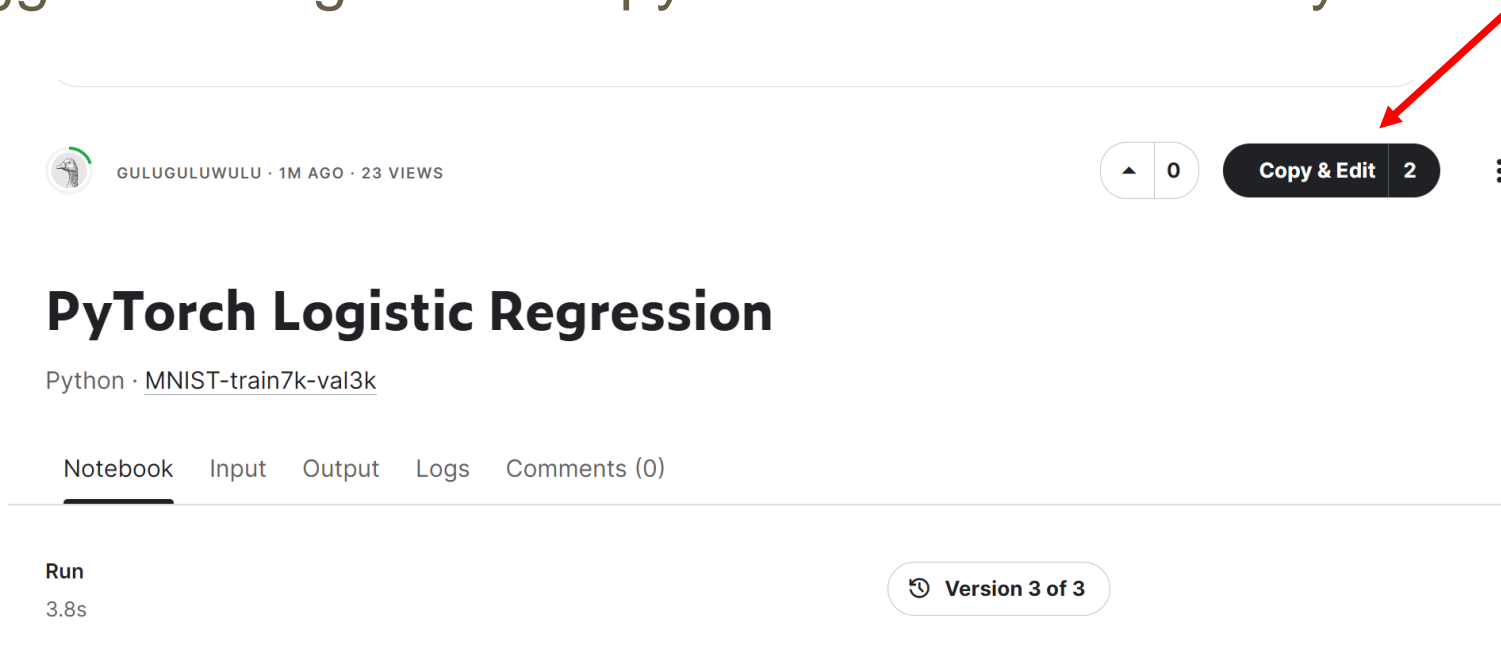
- Additional packages
 - torchvision for Computer Vision
 - torchaudio for Audio
- Useful github repositories using PyTorch
 - [Huggingface Transformers](#) (transformer models: BERT, GPT, ...)
 - [Fairseq](#) (sequence modeling for NLP & speech)
 - [ESPnet](#) (speech recognition, translation, synthesis, ...)
 - Most implementations of recent deep learning papers
 - ...

Homework Specifics

- Two parts:
- Ctrl + 点击访问链接
 - PyTorch Tensors: [kaggle link](#)
 - PyTorch Logistic Regression: [kaggle link](#)

Homework Specifics

- Use Kaggle – 1. Login and Copy the two notebooks to your own account



PyTorch训练逻辑回归模型，完成手写数字分类任务

在这个notebook中，我们介绍用PyTorch训练一个模型的基本流程：

- 构建数据集
- 构建模型
- 定义损失函数以及优化算法进行训练。

Table of Contents



PyTorch训练逻辑回归模型，完成手...

Homework Specifics

- Use Kaggle – 1. Login and Copy the two notebooks to your own account

The screenshot displays the Kaggle user interface for a logged-in user. On the left, a sidebar contains navigation links: 'Create', 'Home', 'Competitions', 'Datasets', 'Models', 'Code', 'Discussions', 'Learn', and 'More'. The 'Your Work' link is highlighted with a red rectangular box. The main area is titled 'Your Work' and includes a 'Create' button. Below this is a horizontal tab bar with 'Overview', 'Collections', 'Code', 'Datasets', 'Models', 'Competitions', 'Discussions', and 'Bookmarks'. A search bar labeled 'Search Your Work' is positioned above a row of filter buttons: 'All Filters', 'Owner', 'Privacy', 'Tags', and 'Language'. The section 'Your Notebooks (2)' follows, showing a list of notebooks. The first notebook, 'PyTorch NN', is highlighted with a red rectangular box. It features a user profile picture, the title 'PyTorch NN', a description 'Notebook copied with edits from guluguluwulu · Updated 15 seconds ago', and the status 'Private · 0 comments'. To the right of the notebook list, there are controls for 'Recently Run' (a dropdown), a trash icon, a counter '0', and a 'draft' status with a menu icon.

Homework Specifics

- Use Kaggle – 2. Notebook provides a Python environment, where Kaggle installs the PyTorch package for us by default.

正文：教程+可运行的代码块

The screenshot displays the Kaggle Notebook interface for a tutorial titled "Lab 1.1 PyTorch Tensor基础". The interface includes a top navigation bar with "File", "Edit", "View", "Run", "Add-ons", and "Help". A sidebar on the left contains icons for file management, search, and other functions. The main content area shows the tutorial text and a code cell with the following code:

```
[2]: import torch
print(torch.cuda.is_available())
print(torch.__version__)
```

The output of the code cell shows:

```
False
2.0.0+cpu
```

A red box highlights the "Notebook" menu, which is open, showing options like "Start session", "Factory reset", "Restart & Clear Cell Outputs", "View Session Metrics", "Upgrade to Google Cloud AI Notebooks", and "Accelerator". The "Accelerator" option is selected, and a sub-menu is shown with the following options:

TRIGGER	SELECT
None	<input checked="" type="checkbox"/>
GPU T4 x2	<input type="checkbox"/>
GPU P100	<input type="checkbox"/>
TPU VM v3-8	<input type="checkbox"/>

The "GPU T4 x2" option is highlighted with a red box. Below the code cell, there is a text box asking "为什么需要张量?" (Why do we need tensors?).

On the right side of the interface, there is a "Notebook" panel with an "Input" section containing "Add Input" and "Upload" buttons, and an "Output" section showing the current directory as "/kaggle/working".

输入

输出

Homework Specifics

- Use Kaggle – 2. Notebook

The screenshot displays a Kaggle Notebook titled "PyTorch Tensors" with a "Draft saved" status. The interface includes a top menu bar with "File", "Edit", "View", "Run", "Add-ons", and "Help". Below the menu, there are icons for adding new content, running cells, and other notebook functions. The main content area shows a markdown cell titled "Lab 1.1 PyTorch Tensor基础" with introductory text and a code cell containing the following Python code:

```
[2]: import torch
print(torch.cuda.is_available())
print(torch.__version__)
```

The output of the code cell shows "False" and "2.0.0+cpu". A red box highlights the "Notebook" menu, which is open, showing options like "Start session", "Factory reset", "Restart & Clear Cell Outputs", "View Session Metrics", "Upgrade to Google Cloud AI Notebooks", and "Accelerator". The "Accelerator" option is selected, and a sub-menu is shown with "None" (selected), "GPU T4 x2", "GPU P100", and "TPU VM v3-8". A red box also highlights the "GPU T4 x2" option.

On the right side of the interface, there is a "Notebook" panel with an "Input" section containing "Add Input" and "Upload" buttons, and an "Output" section showing the file path "/kaggle/working". Below these is a "Session options" section.

输入：可以存放notebook所挂载的数据集，供模型的训练推理使用。

Homework Specifics

- Use Kaggle – 2. Notebook

The screenshot shows the Kaggle Notebook interface for a notebook titled "PyTorch Tensors". The interface is divided into several sections:

- Header:** "PyTorch Tensors" with a "Draft saved" status. Navigation links include "File", "Edit", "View", "Run", "Add-ons", and "Help".
- Toolbar:** Includes icons for adding new cells, deleting, copying, pasting, and running cells. A "Run All" button and a "Markdown" dropdown are also present.
- Main Content Area:**
 - Title:** "Lab 1.1 PyTorch Tensor基础"
 - Text:** "在这个notebook中, 我们将学习PyTorch Tensor的一些基础知识, 并完成最后的练习。"
 - Text:** "同学可以阅读教程并逐个运行代码块来观察程序输出。Lab 1.1中不需要用到大量的GPU加速, Kaggle对GPU加速做了时间上的限制, 只需要同学们在notebook最后部分尝试。如果要打开GPU加速, 请看下图:"
 - Code Cell:** Contains the following code:

```
# Lab 1.1 PyTorch Tensor基础
在这个notebook中, 我们将学习PyTorch Tensor的一些基础知识, 同学可以阅读教程并逐个运行代码块来观察程序输出。Lab 1.1中不需要用到大量的GPU加速, Kaggle对GPU加速做了时间上的限制, 只需要同学们在notebook最后部分尝试。如果要打开GPU加速, 请看下图

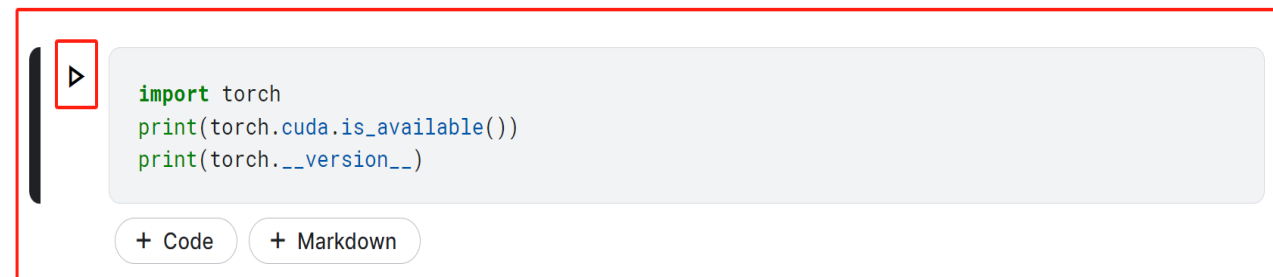
import torch
print(torch.cuda.is_available())
print(torch.__version__)
```

The output of the code cell is:

```
False
2.0.0+cpu
```
 - Accelerator Selection:** A dropdown menu is open, showing options: "None" (selected), "GPU T4 x2", "GPU P100", and "TPU VM v3-8". The "GPU T4 x2" option is highlighted.
- Right Sidebar:**
 - Notebook:** Includes "Share" and "Save Version" (0) buttons.
 - Input:** A section with "Add Input" and "Upload" buttons. It contains a message: "No input attached. Attach a Kaggle dataset, model, or competition".
 - Output:** A section showing the current directory: "/kaggle/working".
 - Session options:** A section with a dropdown arrow.

输出：当前环境一个临时的文件区域，你的代码可以把一些输出存放在Output。例如训练得到的模型权重文件。

Homework Specifics



```
import torch
print(torch.cuda.is_available())
print(torch.__version__)
```

+ Code + Markdown

为什么需要张量？

为了回答这个问题，我们将PyTorch张量和Python列表、NumPy数组做了一个简单的对比。同时我们比较使用GPU加速的PyTorch张量计算和仅用CPU计算的情况。下面这段代码同学们可以不用运行，结果已经给出。

- Use Kaggle – 3. Running a code block in Notebook
- 点击箭头运行
- 代码块可以修改
- 同一个notebook中，命名空间共享：
 - 我在前一个代码块中做了import torch的操作，并运行该代码块。该notebook下的import操作就完成了，那么下一个代码块中即使不显式地再次import，也可以调用torch.xxx。
 - 前一个代码块声明变量a=1，并运行。那么在下一个代码块中我依然能访问到变量a。

Homework Specifics

- 阅读两个notebook中的教程
- 运行教程里的代码块，观察输出
- 完成**PyTorch Tensor**部分练习

作业有五道题，完成四题即可满分（25*4），多做不加分（做错了也不扣分），填入空缺代码并通过测试用例。

提交：

- **DDL: 3.31, 23:59；晚交一天扣10分**
- 完成**PyTorch Tensor**练习。完成后，先运行你所完成练习的代码块，再下载 (File -> Download Notebook) .ipynb文件（**如果不运行，ipynb文件中无法看到你的代码块输出**）
- 把ipynb文件命名为学号_姓名.ipynb，上传到elearning。
- **禁止抄袭**，鼓励大家在各自的小组助教群提问

Thanks!

Any questions?