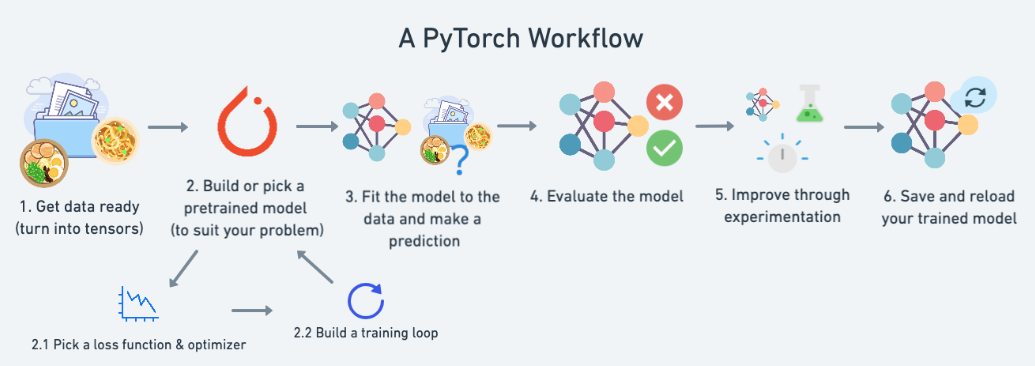
PyTorch Workflow Fundamentals



## **What we're going to cover**

|  |  |
| --- | --- |
| **Topic** | **Contents** |
| 1. Getting data ready | Data can be almost anything but to get started we're going to create a simple straight line |
| 2. Building a model | Here we'll create a model to learn patterns in the data, we'll also choose a loss function, optimizer and build a training loop. |
| 3. Fitting the model to data (training) | We've got data and a model, now let's let the model (try to) find patterns in the (training) data. |
| 4. Making predictions and evaluating a model (inference) | Our model's found patterns in the data, let's compare its findings to the actual (testing) data. |
| 5. Saving and loading a model | You may want to use your model elsewhere, or come back to it later, here we'll cover that. |
| 6. Putting it all together | Let's take all of the above and combine it. |

## **Libraries used for this session**

import torch

from torch import nn # nn contains all of PyTorch's building blocks for neural networks

import matplotlib.pyplot as plt

torch.nn 🡪 library for graphs 🡪 neural networks

link 🡪 <https://pytorch.org/docs/stable/nn.html>

# Data (preparing and loading)

Data can be anything:

1. Excel spread sheet
2. Images
3. Videos
4. Census
5. Text

Machine learning is a game of two parts:

1. Turn your data, whatever it is, into numbers (a representation).
2. Pick or build a model to learn the representation as best as possible.

**How to create data**

For this session, we will use Linear Regression formula to prepare data( most common and popular DL/ML algo)

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



General equation 🡪 y = bx + a

b = slope / regression coefficient(weight)

a = constant/ random error in DL (bias)

*# Create \*known\* parameters*

weight **=** 0.7

bias **=** 0.3

*# Create data*

start **=** 0

end **=** 1

step **=** 0.02

X **=** torch**.**arange(start, end, step)**.**unsqueeze(dim**=**1)

y **=** weight **\*** X **+** bias

X[:10], y[:10]

# Split data into training and test sets