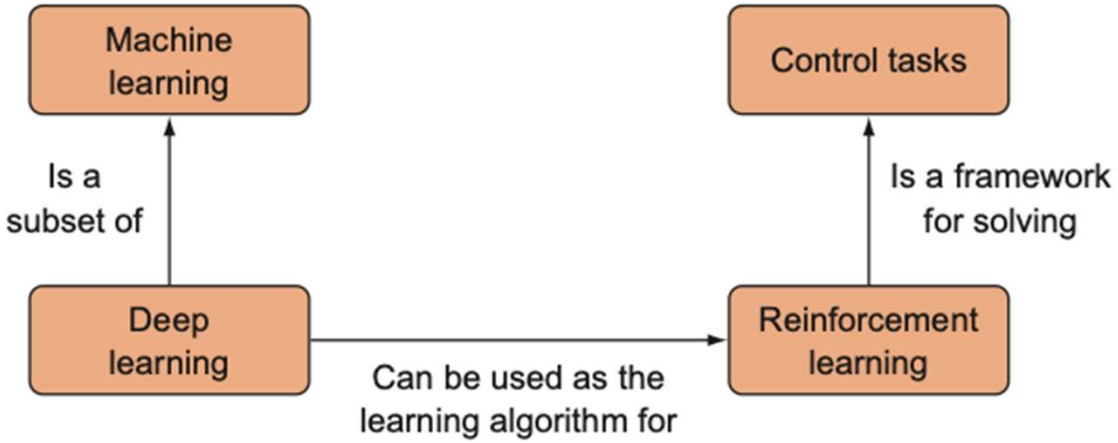
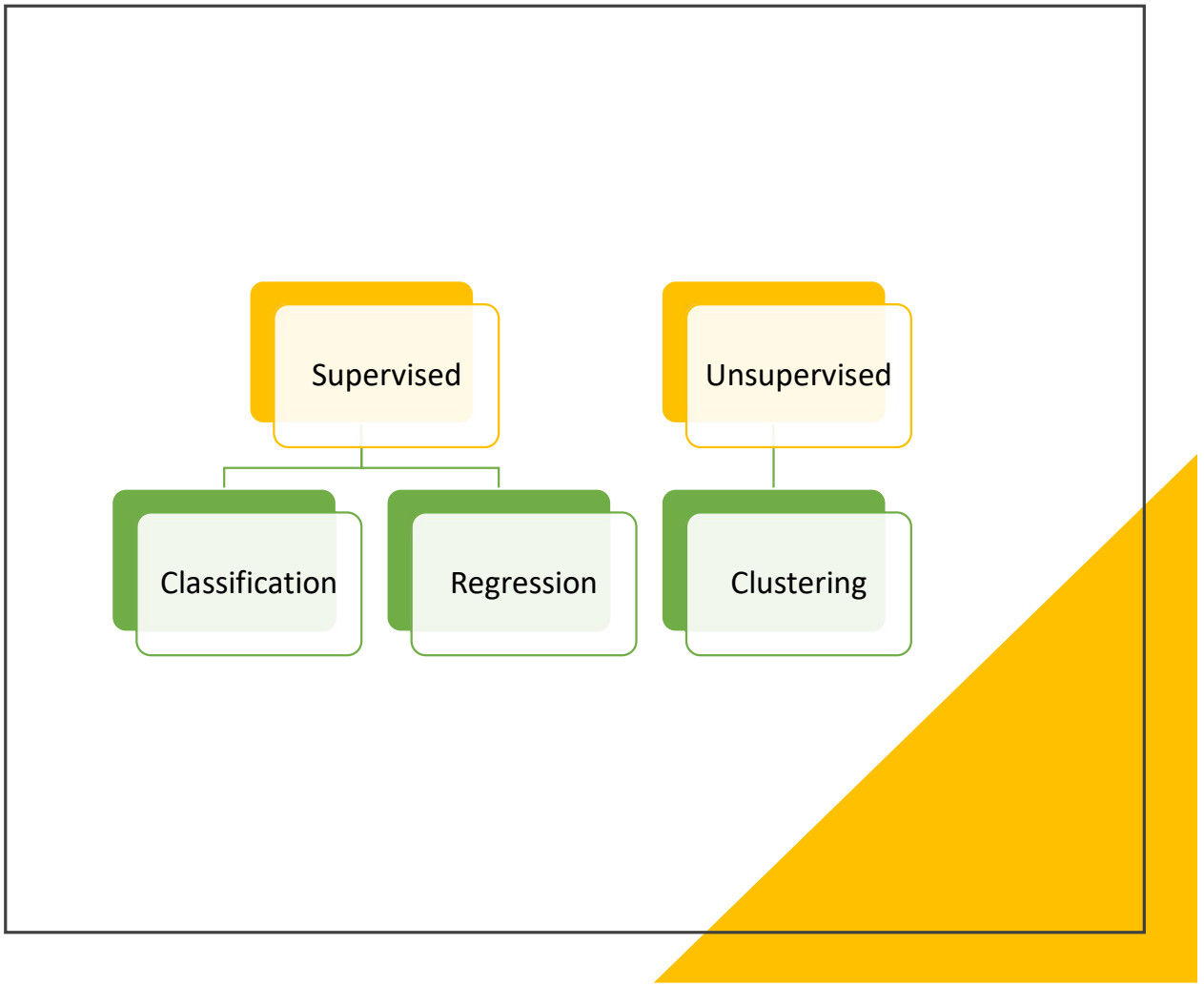
01: Reinforcement Learning

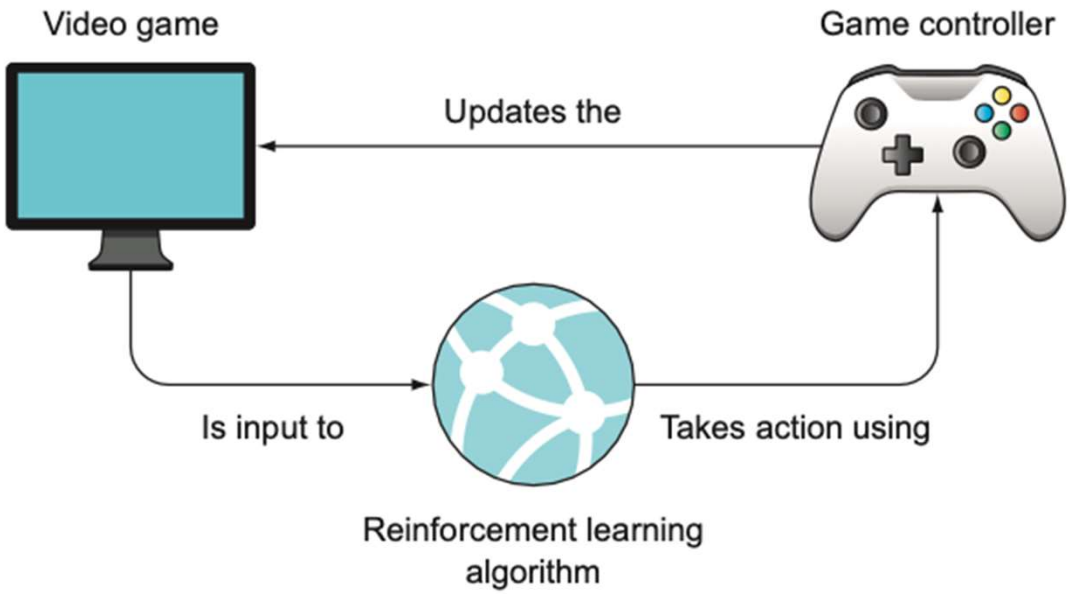
Antorweep Chakravorty



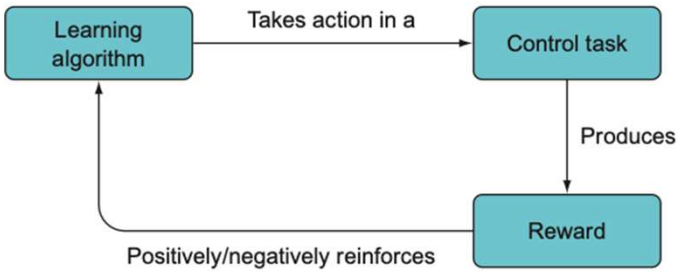
Automated Learning

Machine Learning



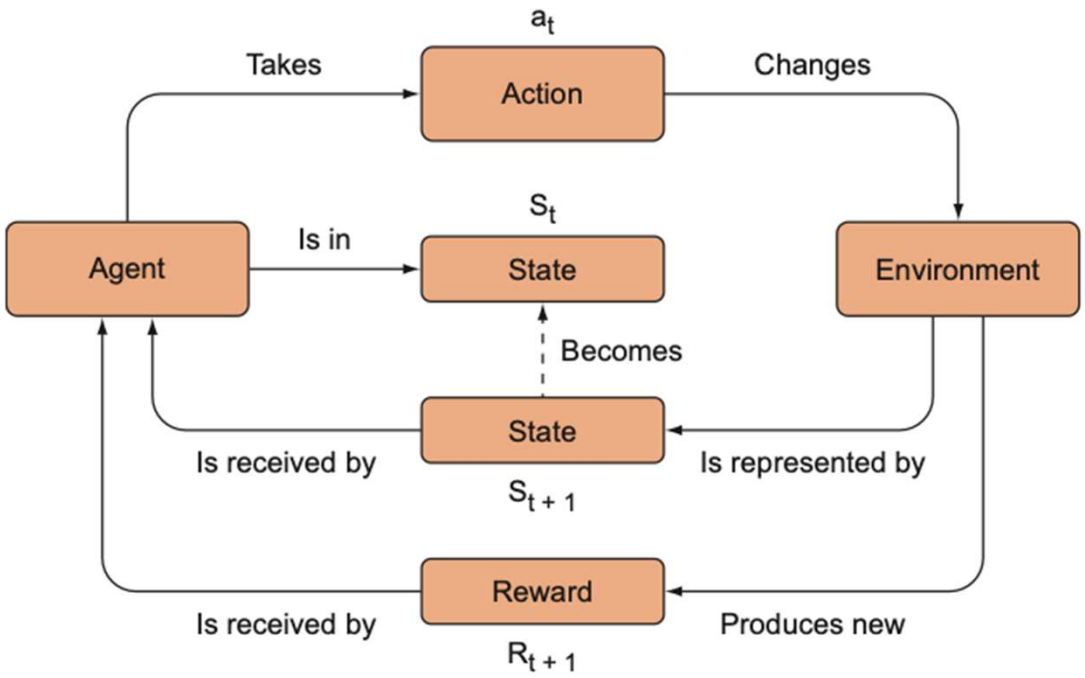


Reinforcement Learning



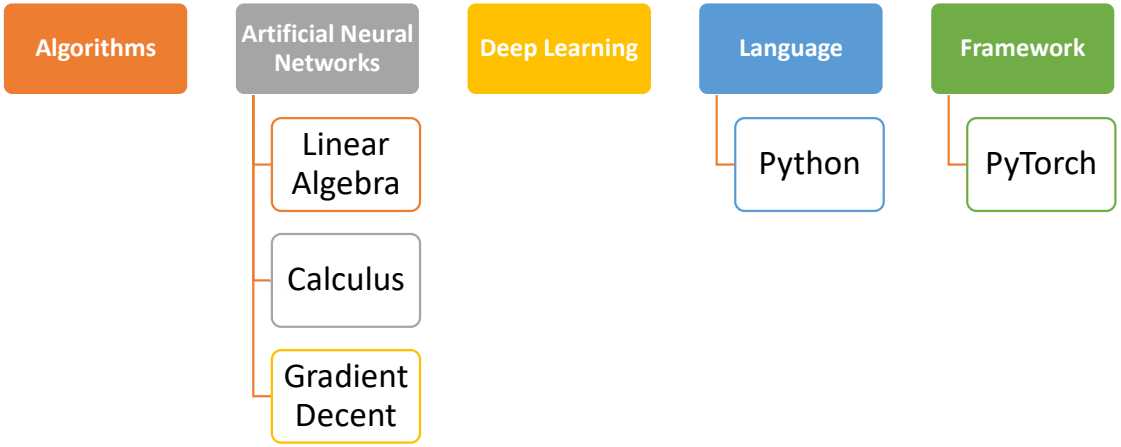
Reinforcement Learning

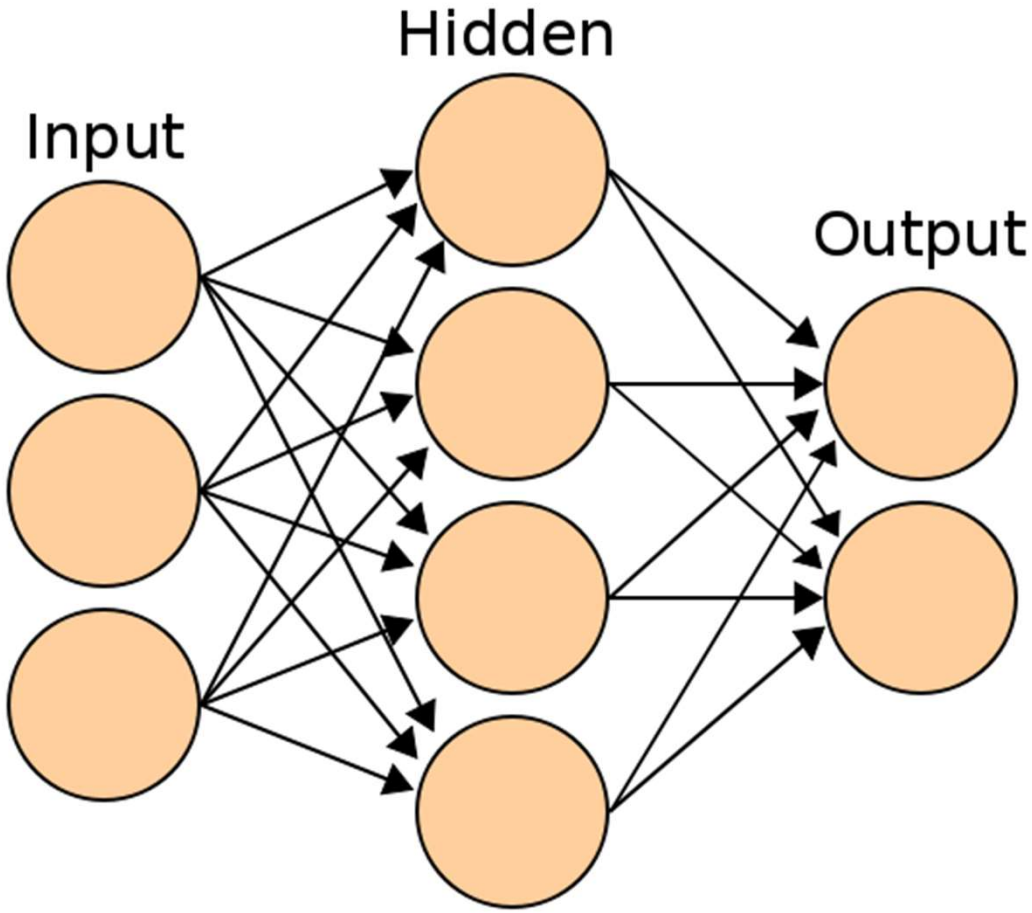
- learn how to act
- not merely to classify or predict
- decisions must be made, or some behavior must be enacted
- => used to solve problems, collectively called control tasks
  - data exists in both time and space
  - decision at a point in time is influenced by what happened at a previous time
- RL algorithms are
  - incentivized to accomplish some high-level goal
  - disincentivized it from doing things we don't want it to do
  - has a single objective—maximizing its reward



Reinforcement Learning

Reinforcement Learning

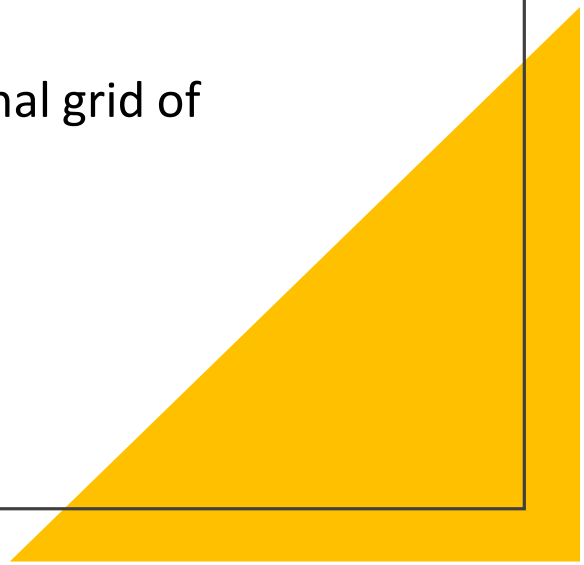




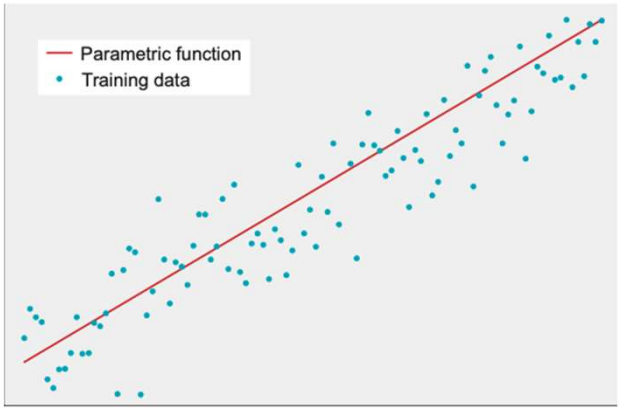
Artificial Neural Network (ANN)

Notations

- Scaler: A single value
- Vector: A single or 1-dimensional sequence of values
- Matrices: A 2-dimensional grid of values



Linear Algebra



- study of linear transformations
- a function that takes an input **x** and maps it to some output **y**

  => f(x) = Ax

a particular output **y** may be larger or smaller than the input **x**, or more generally in a *neighborhood* around an input **x** will be mapped to a larger or smaller neighborhood around the output **y**. The *neighborhood* refers to the set of points arbitrarily *close* to **x** or **y**.

- Often represented as matrices, represented as a rectangular grid of numbers
- Each matrix encode the coefficients for the multivariable linear functions

Linear Algebra

- linear transformations can be applied on a multi-dimensional input variable
- example: we want to map a *2-dimensional* input/point *(x,y)* to a new 2-dimensional point **(x', y')**.
  - we can achieve this by applying linear transformation to map **x** to a new **x'** and **y** to a new **y'**

    => $f_x(x, y) = Ax + By$ ..... Linear Transformation 1

    => $f_y(x, y) = Cx + Dy$ ..... Linear Transformation 2

  - Or a single equation

    => f(x, y) = (Ax + By, Cx + Dy)

  - It can be also represented as

=> $f(x, y) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$

Linear Algebra

In machine learning, we often add a constant to a linear transformation:

$f(x, y) = (Ax + By + B_1, Cx + Dy + B_2)$

OR

$$f(x, y) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

Here $\mathbf{B}_\star$ are constants and termed as *affine* transformations

Calculus

- Study of "differentiation" and integration
- *Differentiation* is the process of getting a derivative of a function

  => The ratio of an output interval to the input interval
- Example: $f(x) = x^2$

In machine learning we are trying to *optimize* a function, which means finding the input points to the function such that the output of the function is a maximum or minimum over all possible inputs

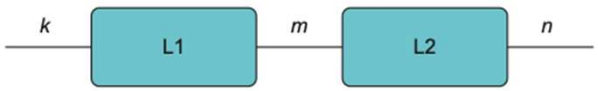
Partial Derivatives

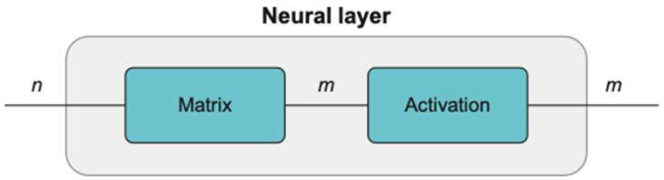
- In machine learning multiple functions work together to determine the output for a given input
- Such functions are called as *composite functions*
- Partial derivatives come into play that describes the curvature of a function with respect to each input segment
- Example: $f(x) = \log(x^4 + x^3 + 2)$

Gradient Decent

- An iterative algorithm to find the minima of a function
- Starts with a random *x* as a starting point
- Computes the derivative of the function at this point
  - Identifies the magnitude and direction of curvature at the point
- Chooses a new *x* point based on the old *x* point (derivative)
- A step-size parameter is applied to control the velocity of the movement

$$x_{new} = x_{old} - a \frac{df}{dx}$$



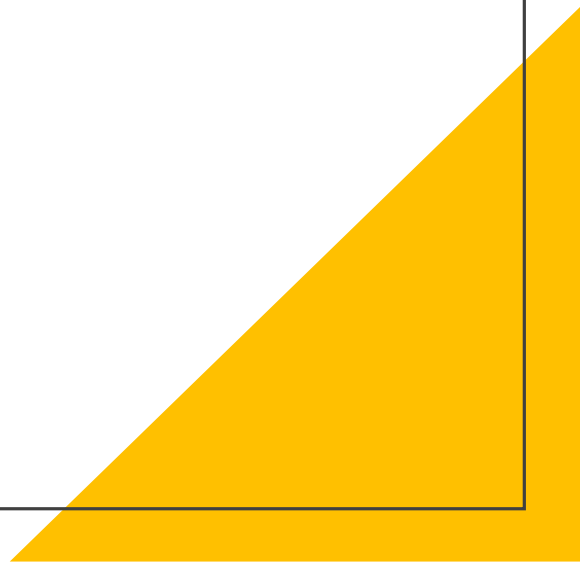


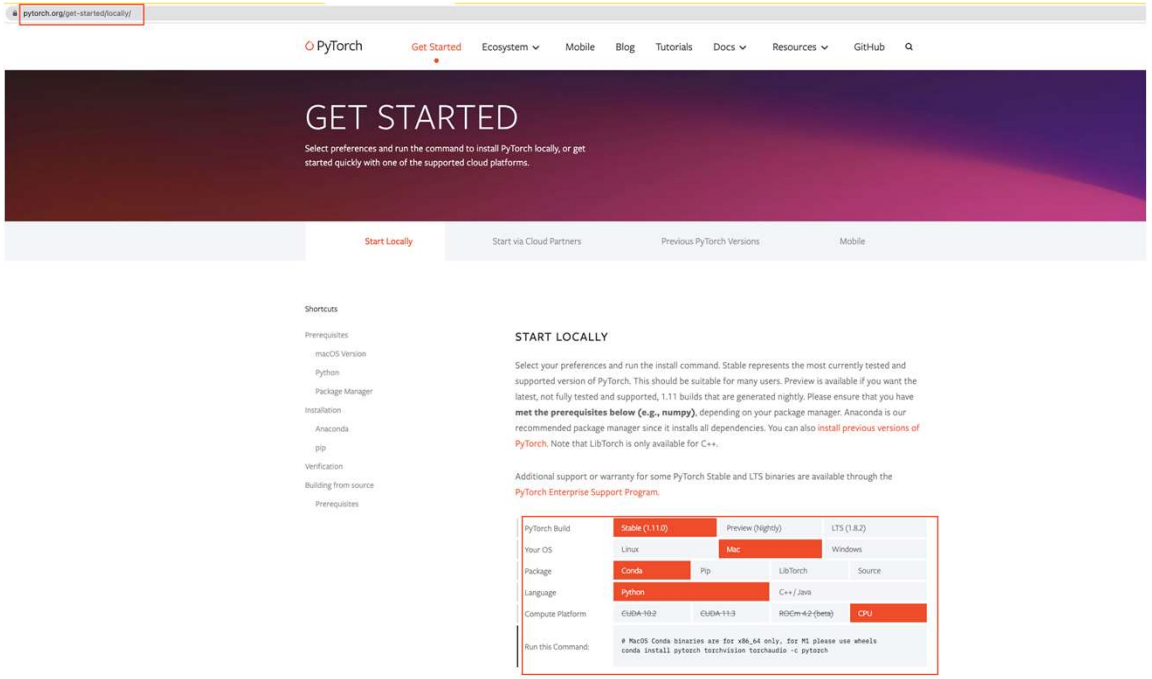
Deep Learning

- An ANN with multiple layers
- Each layer consists of a matrix multiplication followed by a non-linear activation function

PyTorch

- A Deep Learning Framework
- Native looking coding style
- Automatic differentiation and optimization
- Vary similar to NumPy
  - Most, if not all NumPy functionalities are available




pytorch.org/get-started/locally/
PyTorch
Get Started
Ecosystem
Mobile
Blog
Tutorials
Docs
Resources
GitHub
GET STARTED
Select preferences and run the command to install PyTorch locally, or get started quickly with one of the supported cloud platforms.
Start Locally
Start via Cloud Partners
Previous PyTorch Versions
Mobile
Shortcuts
Prerequisites
macOS Version
Python
Package Manager
Installation
Anaconda
pip
Verification
Building from source
Prerequisites
START LOCALLY
Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, 1.11 builds that are generated nightly. Please ensure that you have met the prerequisites below (e.g., numpy), depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.
Additional support or warranty for some PyTorch Stable and LTS binaries are available through the PyTorch Enterprise Support Program.
PyTorch Build
Stable (1.11.0)
Preview (Nightly)
LTS (1.8.2)
Your OS
Linux
Mac
Windows
Package
Conda
Pip
LibTorch
Source
Language
Python
C++ / Java
Compute Platform
CUDA 10.2
CUDA 11.3
ROCm 4.2 (beta)
CPU
Run this Command:
MacOS Conda binaries are for x86_64 only, for M1 please use wheels
conda install pytorch torchvision torchaudio -c pytorch

PyTorch