TensorFlow… PyTorch…

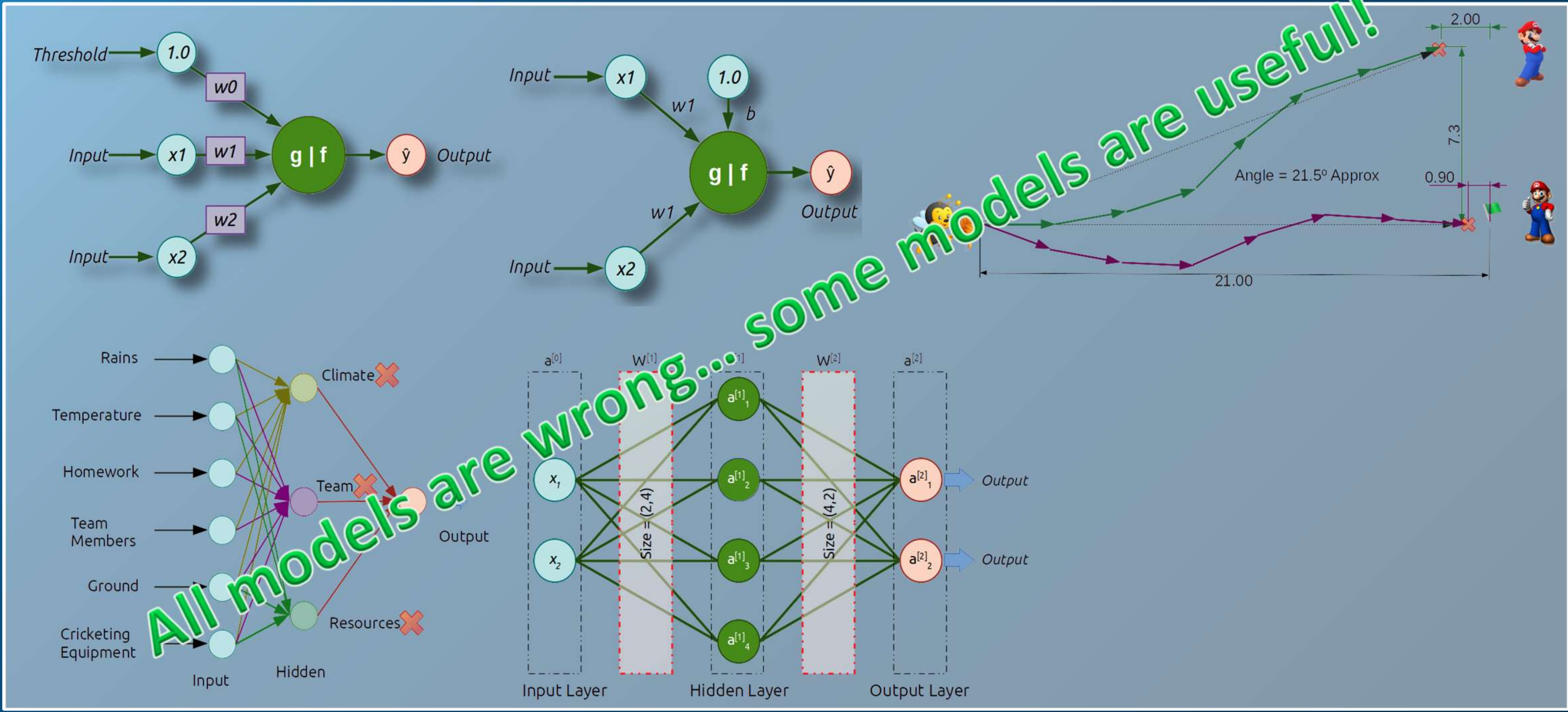
Deep Neural Networks

Session 06

Pramod Sharma
pramod.sharma@prasami.com

Agenda

Story So far…

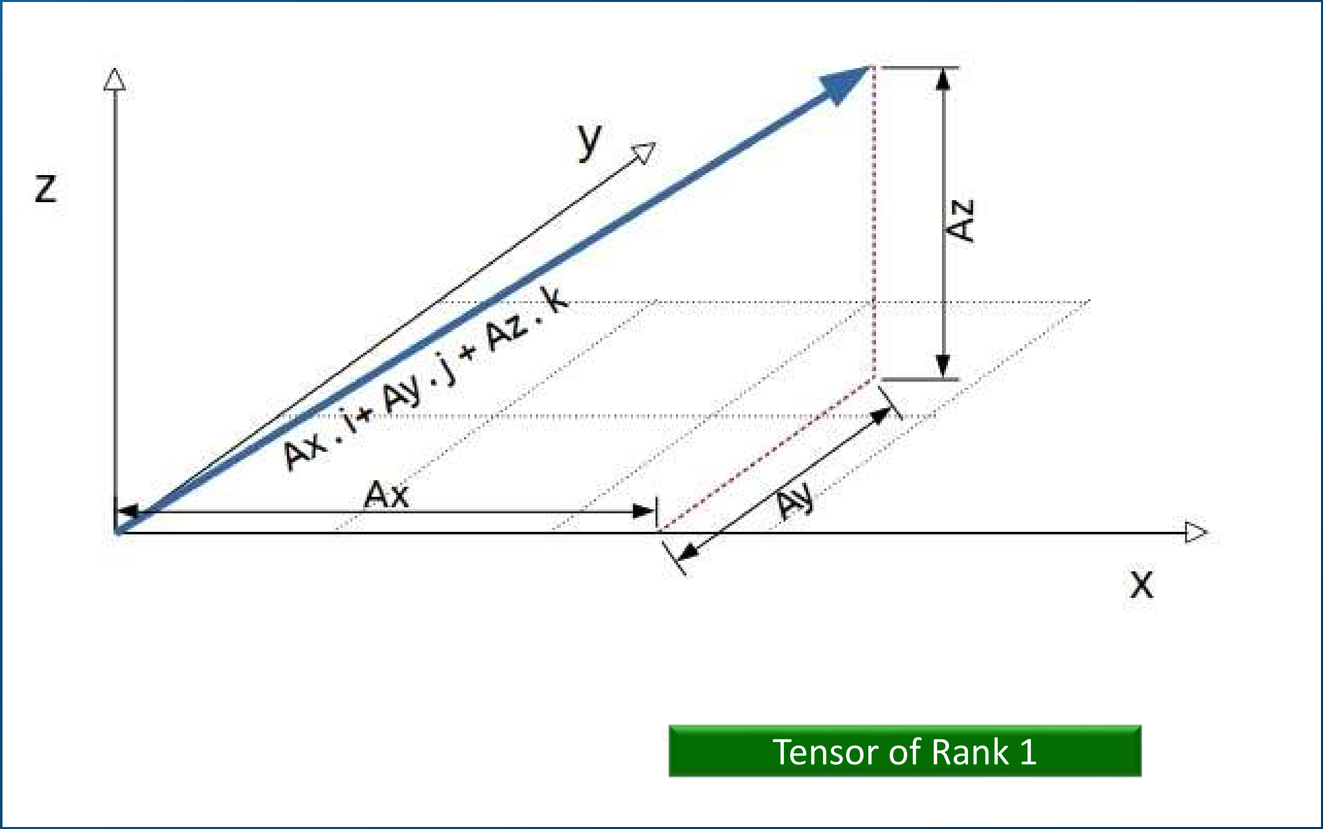


pra-sâmi

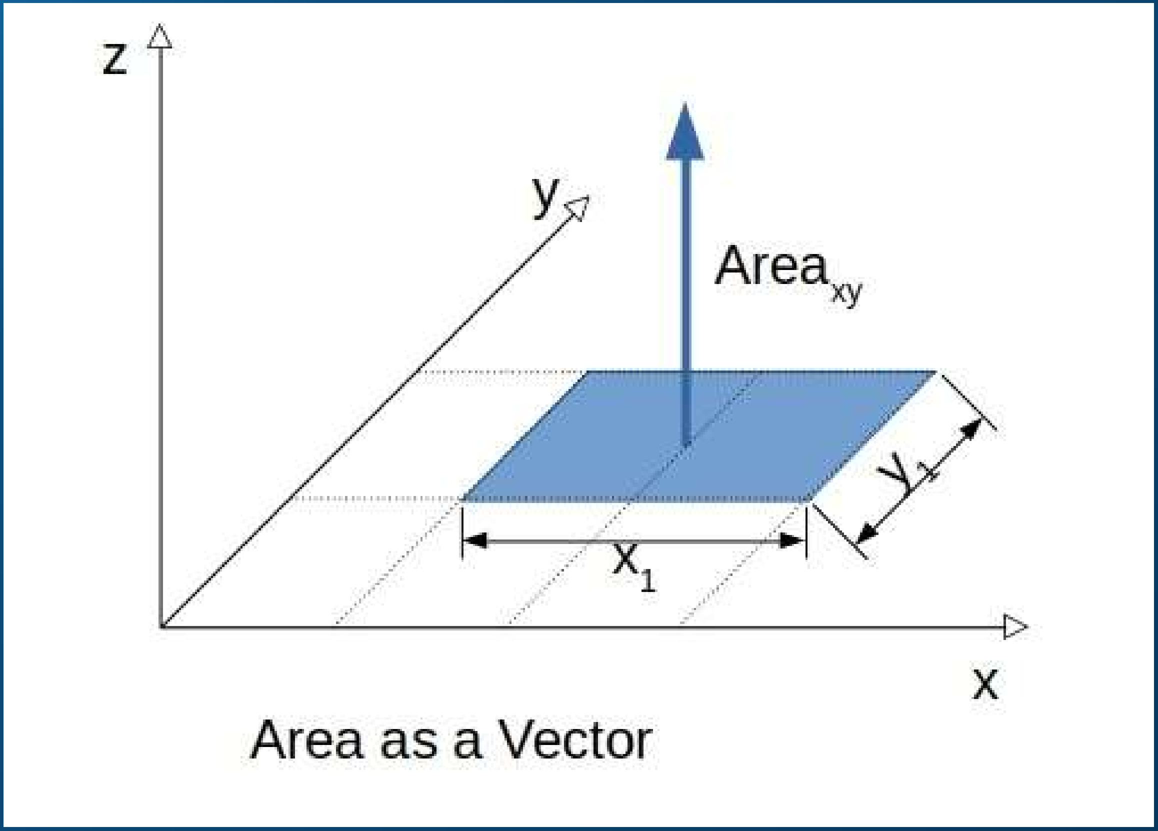
What is Tensor…

What is a vector????

Vector – Rank 1 Tensor

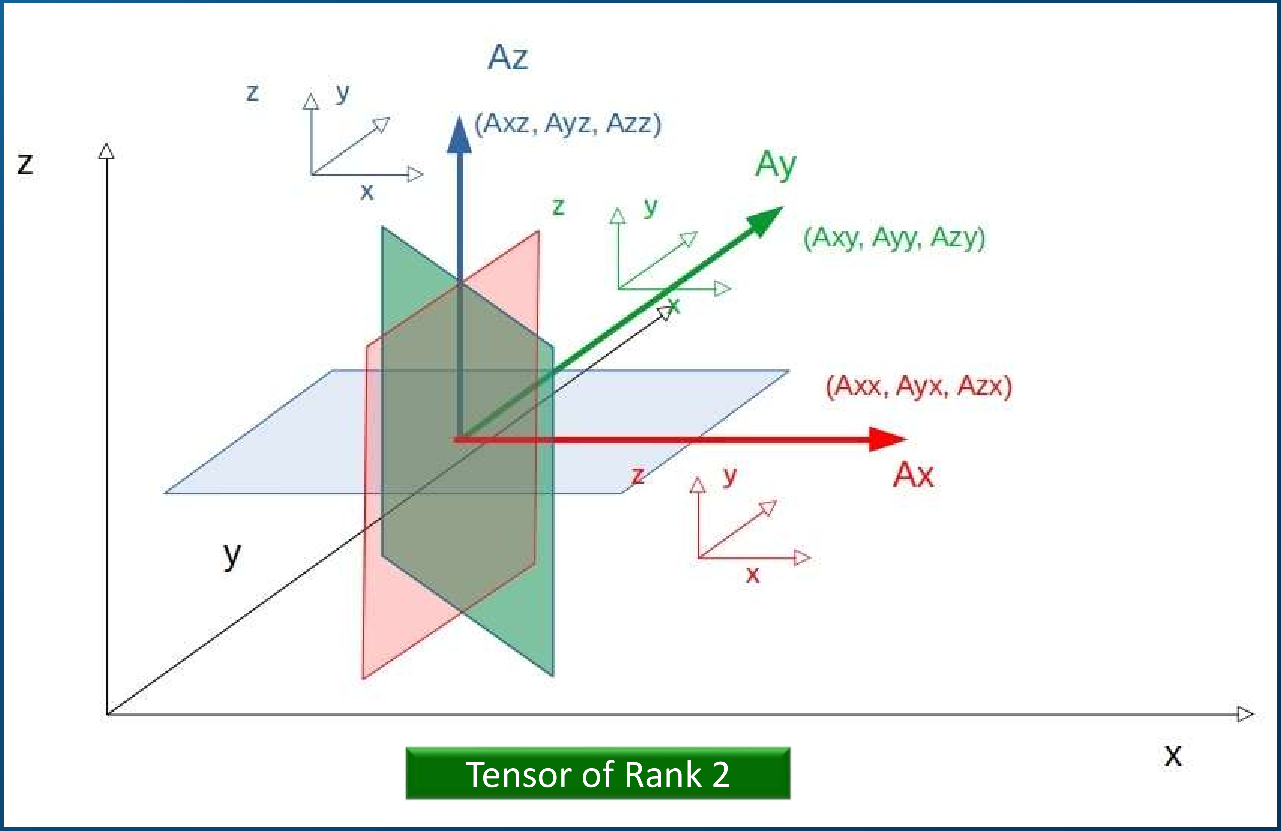


Area too can be represented as vector….

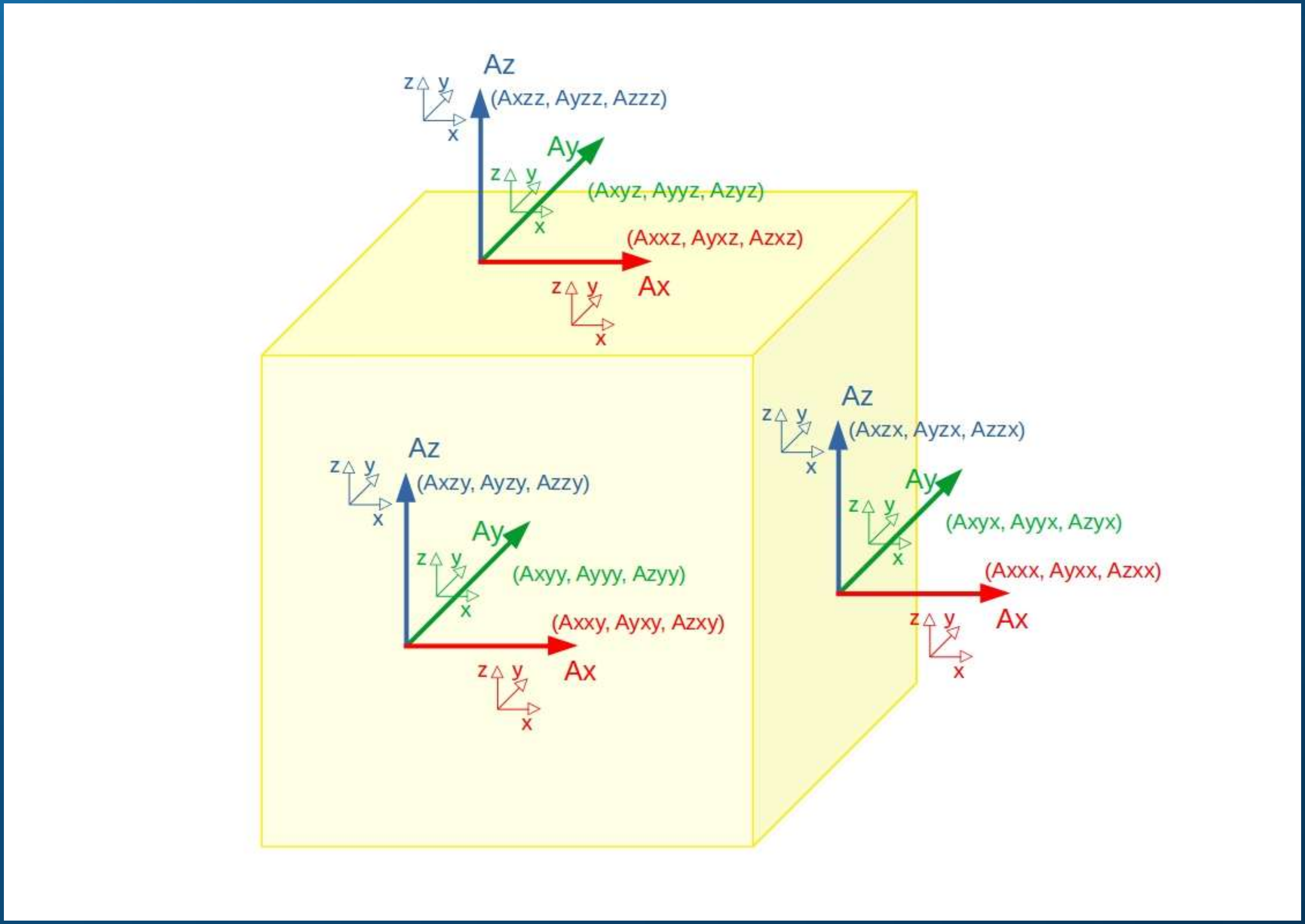


Area as a Vector

Area can have three vectors attached to it…



So What is Rank 3 Tensor….



pra-sâmi

Are there more than Rank 3 Tensors….

- The rank(order) R of a tensor is independent of the number of dimensions N of the underlying space
- Consider intuitively that a tensor represents a physical entity which may be characterized by magnitude and multiple directions simultaneously (Fleisch 2012).
- Therefore, the number of simultaneous directions is denoted R and is called the rank of the tensor in question.
- A rank-0 tensor (i.e., a scalar) can be represented by N^0 = 1
- A rank-1 tensor (i.e., a vector) in N-dimensional space can be represented by N^1 = N
- A general ranked tensor by N^R numbers

How many rank tensor can have?

Rank	Object
0	Scalar
1	Vector
2	Matrix
>=3	Tensor

So What are Tensors

- ❑ A tensor is a multidimensional array with a uniform data type
- ❑ You can never update a tensor but create a new one
- ❑ Looks similar to Numpy Array, even behave similar way in some aspects
- ❑ A Tensor is a suitable choice on GPU
- ❑ A tensor can reside in accelerator’s memory

As a Data Scientist…

- A tensor is a type of *multidimensional array* with certain *transformation properties*

- Let's take for example velocity of some object:
  - It can be represented by three numbers, or a multidimensional array (1 x 3).
  - Value in this array depends on your system on reference.
  - In one system of reference these numbers can be [100, 0, 0].
  - In another system of reference the numbers corresponding to the velocity of this very object at this very moment can be absolutely different.
  - Let's say [60, 0, -80]

- You toss a ball in the air, how many numbers do I need to define it's velocity?
  - 1… 2… 6! Right?
    - $v_x, v_y, v_z, r_x, r_y, r_z$
  - What if I am standing outside earth?
  - Outside our galaxy…. My head is spinning already!

- It's the rules of changing representation when switching between systems of reference that make multidimensional array a tensor.

"Tensors are fact of universe" - Lillian Liebe

“*Numpy are excellent but run on CPU only*

Tensors in Tensorflow and PyTorch are attempts to make it run on GPU.”

Why Tensors!

Deep Learning Frameworks Landscape

Deep Learning Frameworks Landscape

TensorFlow

PyTorch

K Keras

Sonnet

mxnet

GLUON

DL4J

Chainer

pandas

CuPy

matplotlib

Deep Learning Frameworks Landscape

pra-sâmi

Deep Learning Frameworks Landscape

pra-sâmi

Deep Learning Frameworks Landscape

TensorFlow

Keras

Sonnet

mxnet

GLUON

DL4J

Chainer

pandas

CuPy

matplotlib

Deep Learning Frameworks Landscape

Deep Learning Frameworks Landscape

Deep Learning Frameworks Landscape

TensorFlow

PyTorch

K Keras

Sonnet

GLUON

DL4J

Chainer

pandas

CuPy

matplotlib

Deep Learning Frameworks Landscape

TensorFlow

PyTorch

K Keras

DL4J

GLUON

Chainer

pandas

CuPy

matplotlib

Deep Learning Frameworks Landscape

Deep Learning Frameworks Landscape

TensorFlow

PyTorch

K Keras

Sonnet

Chainer

DL4J

pandas

CuPy

matplotlib

TensorFlow

TensorFlow vs. Theano

- Theano was an inspiration for Tensorflow
  - A deep-learning library with python wrapper
- Theano and TensorFlow are very similar systems
- TensorFlow has better support for distributed systems though
- Development of Tensorflow is funded by Google, while Theano is an academic project.
- TensorFlow and Numpy are quite similar
  - Both are N-d array libraries!
- Numpy has Ndarray support, but doesn’t offer methods to create tensor functions and automatically compute derivatives
- Numpy had no GPU support
  - CuPy for GPU support

Formats are near similar!

Numpy	TensorFlow
a = np.zeros((2,2)); b = np.ones((2,2))	a = tf.zeros((2,2)), b = tf.ones((2,2))
np.sum(b, axis=1)	tf.reduce_sum(a,reduction_indices=[1])
a.shape	a.get_shape()
np.reshape(a, (1,4))	tf.reshape(a, (1,4))
b * 5 + 1	b * 5 + 1
np.dot(a,b)	tf.matmul(a, b)
a[0,0], a[:,0], a[0,:]	a[0,0], a[:,0], a[0,:]

TensorFlow requires explicit evaluation!

- a = np.zeros((2,2))
- ta = tf.zeros((2,2))
- print(a)

```
[[ 0. 0.]
 [ 0. 0.]]
```

- print(ta)

```
Tensor("zeros_1:0", shape=(2, 2), dtype=float32)
```

- print(ta.eval())

```
[[ 0. 0.]
 [ 0. 0.]]
```

TensorFlow computations define a computation graph that has no numerical value until evaluated!

Session Object

- Till version 1:
  "A Session object encapsulates the environment in which Tensor objects are evaluated"
  - TensorFlow Docs

- a = tf.constant (5.0)
  b = tf.constant (6.0)

- c = a * b

- with tf.Session() as sess:

```
        print(sess.run(c))
        print(c.eval())
```

Session Object

- Till version 1 - "A Session obj tes the environment in which Tensor objects are evaluated" - TensorFl
- a = tf.constant(5.0
  b = tf.constant(
- c = a * b
- with tf.Sess s sess:

  ```
          print( un(c))
          print( )
  ```

Not available in Version 2.0!

pra-sâmi

TensorFlow

- "TensorFlow programs are usually structured into a construction phase, that assembles a graph, and an execution phase that uses a session to execute ops in the graph." - TensorFlow docs
- All computations add nodes to global default graph (docs)
- "When you train a model you use variables to hold and update parameters. Variables are in-memory buffers containing tensors" - TensorFlow Docs.
- Variables are created and tracked via the tf.Variable class.
  - A tf.Variable represents a tensor whose value can be changed by running ops on it.
  - Specific ops allow you to read and modify the values of this tensor.
  - Higher level libraries like tf.keras use tf.Variable to store model parameters. -- TensorFlow Docs

A lot could have changed since last update

Must Read: https://www.tensorflow.org/guide/variable

PyTorch

Pytorch

- An open source machine learning framework that accelerates the path from research prototyping to production deployment

- KEY FEATURES & CAPABILITIES
  - Production Ready
    - Transition seamlessly between eager and graph modes with TorchScript, and accelerate the path to production with TorchServe.
  - Distributed Training
    - Scalable distributed training and performance optimization in research and production is enabled by the torch.distributed backend.
  - Robust Ecosystem
    - A rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.
  - Cloud Support
    - PyTorch is well supported on major cloud platforms, providing frictionless development and easy scaling.

What is PyTorch?

- It’s a Python-based scientific computing package targeted at two sets of audiences:
  - A replacement for NumPy to use the power of GPUs
  - A deep learning research platform that provides maximum flexibility and speed

- A PyTorch Tensor is basically the same as a numpy array:
  - It does not know anything about deep learning or computational graphs or gradients, and
  - Is just a generic n-dimensional array to be used for arbitrary numeric computation

https://pytorch.org/tutorials/beginner/basics/intro.html

https://pytorch.org/tutorials/beginner/examples_tensor/two_layer_net_tensor.html

PyTorch

- Three levels of abstraction:
  - Tensor: Tensor is an imperative n-dimensional array which runs on GPU.
  - Variable: It is a node in the computational graph. This stores data and gradient.
  - Module: Neural network layer will store state the otherwise learnable weight.

- **torch.from_numpy():**
  - To create a tensor from numpy.ndarray. The ndarray and return tensor share the same memory.
  - If we make any changes in the returned tensor, then it will reflect the ndarray also

- **Variable** is a package which is used to wrap a tensor.

- **autograd.variable:**
  - Provides classes and functions for implementing automatic differentiation of arbitrary scalar-valued functions.
  - We only need to declare tensor for which gradients should be computed with the '*requires_grad=True*'

- Calculating **Gradient**:
  - Initialization of the function for which we will calculate the derivatives.
  - Set the value of the variable which is used in the function.
  - Compute the derivative of the function by using the backward () method.
  - Print the value of the derivative using grad.

Tensorflow vs. PyTorch

Comparison

Features	PyTorch	TensorFlow 2.0
Created by	FAIR Lab (Facebook AI Research Lab)	Google Brain Team
Based on	Torch	Theano
Production	Research focused	Industry-focused
Visualization	Visdom	Tensorboard
Deployment	Torch Serve (experimental)	TensorFlow Serve
Mobile Deployment	Yes (experimental)	Yes
Device Management	CUDA	Automated
Graph Generation	Dynamic and static mode	Eager and static mode
Learning Curve	Easier for developers and scientists	Easier for industry-level projects
Use Cases	Facebook CheXNet Tesla Autopilot Uber PYRO	Google Sinovation Ventures PayPal China Mobile

Comparison

TENSORFLOW	PYTORCH
PROS:	**PROS:**
Simple built-in high-level API	Python-like coding
Visualizing training with Tensorboard	Dynamic graph
Production-ready thanks to TensorFlow serving	Easy & quick editing
Easy mobile support	Good documentation and community support
Open source	Open source
Good documentation and community support	Plenty of projects out there using PyTorch
CONS:	**CONS:**
Static graph	Third-party needed for visualization.
Debugging method	API server needed for production.
Hard to make quick changes	

Visualization

PyTorch and TensorFlow both have tools for quick visual analysis.

TensorFlow	PyTorch
❑ Tensorboard is used for visualizing data. ❖ The interface is interactive and visually appealing. ❖ Tensorboard provides a detailed overview of metrics and training data. ❖ The data is easily exported and looks great for presentation purposes. ❖ Plugins make Tensorboard available for PyTorch as well. ❑ However, Tensorboard is cumbersome and complicated to use.	❑ PyTorch uses Visdom for visualization. ❖ The interface is lightweight and straightforward to use. ❖ Visdom is flexible and customizable. ❖ Direct support for PyTorch tensors makes it simple to use. ❑ Visdom lacks interactivity and many essential features for overviewing data.

pra-sâmi

Production Deployment

When it comes to deploying trained models to production, TensorFlow is the clear winner.

TensorFlow	PyTorch
❑ We can directly deploy models in TensorFlow using TensorFlow serving which is a framework that uses REST Client API.	❑ In PyTorch, these production deployments became easier to handle than in it's latest 1.0 stable version, but it doesn't provide any framework to deploy models directly on to the web. You'll have to use either Flask, FastAPI or Django as the backend server. So, TensorFlow serving may be a better option if performance is a concern.

Defining a Simple Neural Network

TensorFlow

- ❑ Recently Keras, a neural network framework which uses TensorFlow as the backend was merged into TF Repository.
- ❑ From then on the syntax of declaring layers in TensorFlow was similar to the syntax of Keras.
- ❑ First, we declare the variable and assign it to the type of architecture we will be declaring, in this case a “Sequential()” architecture.
- ❑ Next, we directly add layers in a sequential manner using model.add() method.
- ❑ The type of layer can be imported from tf.layers

PyTorch

- ❑ Neural network will be a class and using torch.nn package
- ❑ Import the necessary layers that are needed to build your architecture.
- ❑ All the layers are first declared in the __init__() method, and then in the forward() method we define how input x is traversed to all the layers in the network.
- ❑ Declare a variable model and assign it to the defined architecture (model = NeuralNet())

Reflect…

- Loss Function:
  - Used to evaluate how well our algorithm is modeling training data.
    - If our prediction is completely off, then the function will output a higher number else it will output a lower number
- Activation function
  - A neuron should be activated or not, is determined by an activation function
  - Neuron has two functions – an Aggregation function and an Activation function
- Perceptron:
  - Perceptron is a single neuron neural network. Perceptron is a binary classifier, and it is used in supervised learning.
    - A simple model of a biological neuron in an artificial neural network is known as Perceptron
- Backpropagation:
  - Backpropogation of error to learn model parameters.
  - Algorithms are a set of methods to solve NN as optimization problem.

Reflect…

- ❑ Which of the subsequent declaration(s) effectively represents an actual neuron in TensorFlow?
  a. A neuron has a single enter and a single output best
  b. A neuron has multiple inputs but a single output only
  c. A neuron has a single input, however, more than one outputs
  d. A neuron has multiple inputs and more than one outputs
  e. All of the above statements are valid

Let’s Code…

Next Session

- Loss / Cost Optimization
- Stochastic Gradient Descent (SGD) and others
- Momentum Learning Rates
- Adaptive Learning Rates

pra-sâmi