DEEP LEARNING Data Representation for a Neural Networks

Lecture 4

FARZEEN ASHFAQ

Important Libraries

- Several open-source software libraries are available for training *DNNs*
  - Caffe (Berkeley)
  - Theano (University of Montreal)
  - Tensorflow (Google Brain)
  - PyTorch (adopted by Facebook AI)

TensorFlow vs. Numpy

- Few people make this comparison, but 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 (+ no GPU support).

Data Representation

- Tensors = multidimensional numpy arrays
- A tensor is a container for data—almost always **numerical data**.
- So, it's a container for numbers.
- You may be already familiar with matrices, which are 2D tensors: tensors are a ***generalization of matrices to an arbitrary number of dimension***

Scalars (0D tensors)

- A tensor that contains only one number is called a scalar (or scalar tensor, or 0-dimensional tensor, or 0D tensor).
- In Numpy, a float32 or float64 number is a scalar tensor (or scalar array).
- You can display the number of axes of a Numpy tensor via the **ndim** attribute; a scalar tensor has 0 axes (ndim == 0). The number of axes of a tensor is also called its **rank**.

Example 0D tensor

```
>>> import numpy as np
>>> x = np.array(12)
>>> x
array(12)
>>> x.ndim
0
```

Vectors (1D tensors)

- An array of numbers is called a vector, or 1D tensor. A 1D tensor is said to have exactly one axis. Following is a Numpy vector:

Example ID tensor

```
>>> x = np.array([12, 3, 6, 14])
>>> x
array([12, 3, 6, 14])
>>> x.ndim
1
```

Matrices (2D tensors)

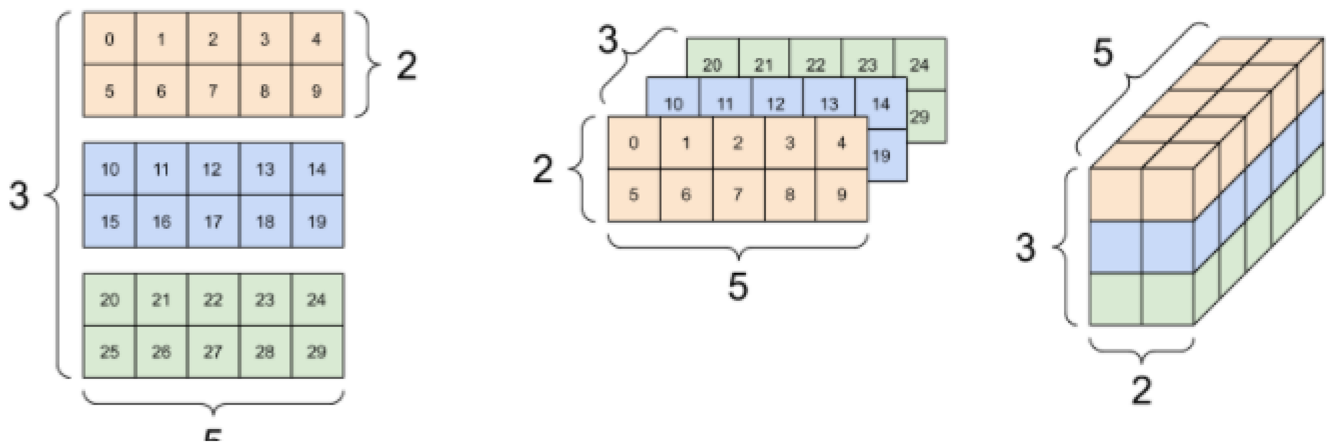
- An array of vectors is a matrix, or 2D tensor. A matrix has two axes (often referred to rows and columns). You can visually interpret a matrix as a rectangular grid of numbers.

Example 2D Tensor

```
>>> x = np.array([[5, 78, 2, 34, 0],
                  [6, 79, 3, 35, 1],
                  [7, 80, 4, 36, 2]])
>>> x.ndim
2
```

3D tensors and higher-dimensional tensors

- If you pack such matrices in a new array, you obtain a 3D tensor, which you can visually interpret as a cube of numbers.



Example n-d Tensor

```
>>> x = np.array([[[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]]])
>>> x.ndim
3
```

Summary – Key Attributes of tensor

- Number of axes (rank): For instance, a 3D tensor has three axes, and a matrix has two axes. This is also called the tensor's ndim in Python libraries such as Numpy.
- Shape: This is a tuple of integers that describes how many dimensions the tensor has along each axis. For instance, the previous matrix example has shape (3, 5), and the 3D tensor example has shape (3, 3, 5). A vector has a shape with a single element, such as (5,), whereas a scalar has an empty shape, ().
- Data type (usually called dtype in Python libraries):This is the type of the data contained in the tensor; for instance, a tensor's type could be float32, uint8, float64, and so on. On rare occasions, you may see a char tensor.

Note that string tensors don't exist in Numpy (or in most other libraries), because tensors live in preallocated, contiguous memory segments: and strings, being variable length, would preclude the use of this implementation.

TF Session and Graph

- A **graph** defines the computation. It doesn’t compute anything, it doesn’t hold any values, it just defines the operations that you specified in your code.
- A **session** allows to execute graphs or part of graphs. It allocates resources (on one or more machines) for that and holds the actual values of intermediate results and variables.

Tensor Flow Mechanics