Neural Network from scratch using NumPy

Why NumPy for neural network from scratch?

Why NumPy for neural network from scratch?

What is NumPy?

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Why NumPy for neural network from scratch?

What is NumPy?

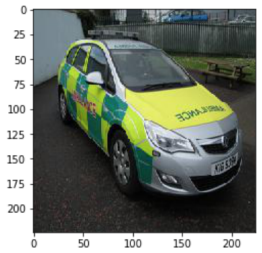
NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Why NumPy for neural network from scratch?

```
In [4]: # read an image
        img = plt.imread('../datasets/emergency_classification/images/0.jpg')

In [5]: # print image
        img

Out[5]: array([[[115, 134, 141],
                [116, 135, 142],
                [116, 135, 142],
                ...,
                [ 70,  81,  87],
                [ 74,  85,  91],
                [ 82,  93,  99]],

               [[115, 134, 141],
                [116, 135, 142],
                [116, 135, 142],
                ...,
                [ 82,  93,  99],
                [ 82,  93,  99],
                [ 83,  94, 100]],

               [[115, 134, 141],
                [116, 135, 142],
                [116, 135, 142],
                ...,
                [ 77,  88,  94]

In [6]: # plot image
        plt.imshow(img)

Out[6]: <matplotlib.image.AxesImage at 0x7f1675b79d68>
```



Analytics Vidhya
Learn everything about analytics

Why NumPy for neural network from scratch?

1. You already worked with numpy

Why NumPy for neural network from scratch?

1. You already worked with numpy

2. Good starting point to focus on neural network compared to tool

Why NumPy for neural network from scratch?

1. You already worked with numpy

2. Good starting point to focus on neural network compared to tool

3. NumPy works well with matrices/ array

Why NumPy for neural network from scratch?

1. You already worked with numpy
2. Good starting point to focus on neural network compared to tool
3. NumPy works well with matrices/ array
4. Learn while coding the forward and backward propagation