Chapter 1: Introduction PyTorch, Tensors, Tensor Operations and Basics

**Introduction**

PyTorch has been evolving as a larger framework for writing dynamic models. Because of that it is very much popular among the data scientists and among the data engineers in deploying large scale deep learning frameworks. This book provides a structure for the experts in terms of handling the activities while working on a practical data science problem. As it is evident from various applications that we currently use in our day to day life, there are layers of intelligence embedded with the product features. Those features are enabled to provide better experience, better services to the user. As the world is moving towards artificial intelligence, there are two main components of it they are deep learning and machine learning, without which it is impossible to visualize artificial intelligence. The following image explains how the artificial intelligence and machine learning and deep learning are associated.

The industries where the artificial intelligence is being applied includes banking, financial services, insurance, healthcare, manufacturing, retail, clinical trials and drug testing etc. It involves in classifying the objects, recognizing the objects to detecting fraud etc. the following image explains the relationship between the three topics. Every learning system requires three things, input data, processing and output layer. If the performance of any learning system improves over time by learning from new examples or data, can be called as a machine learning system. When the machine learning system become too difficult to reflect the reality, it requires a deep learning system. In a deep learning system more than one layer of learning algorithm is being deployed.

Figure 1: relationship between ML, DL and AI

There are various challenges in deploying the deep learning models, it requires large volume of labeled data, better or rather faster computing machines, intelligent algorithms. The very success of any deep learning system requires good labelled data and better computing machines as the smart algorithms are already available. There are various use cases that requires deep learning implementation:

* Speech recognition
* Video analysis
* Anomaly detection from videos
* Natural language processing
* Machine translation
* Speech to text conversion

Development of NVIDIA GPU (graphics processing unit) computing for processing large scale data is another path breaking innovation. The programming language that is required to run in a GPU environment requires a different programming framework. Two major frameworks are very popular for implementing graphical computing, they are TensorFlow and PyTorch. In this book we are going to discuss about PyTorch as a framework to implement data science algorithms and make inferences.

## What is PyTorch:

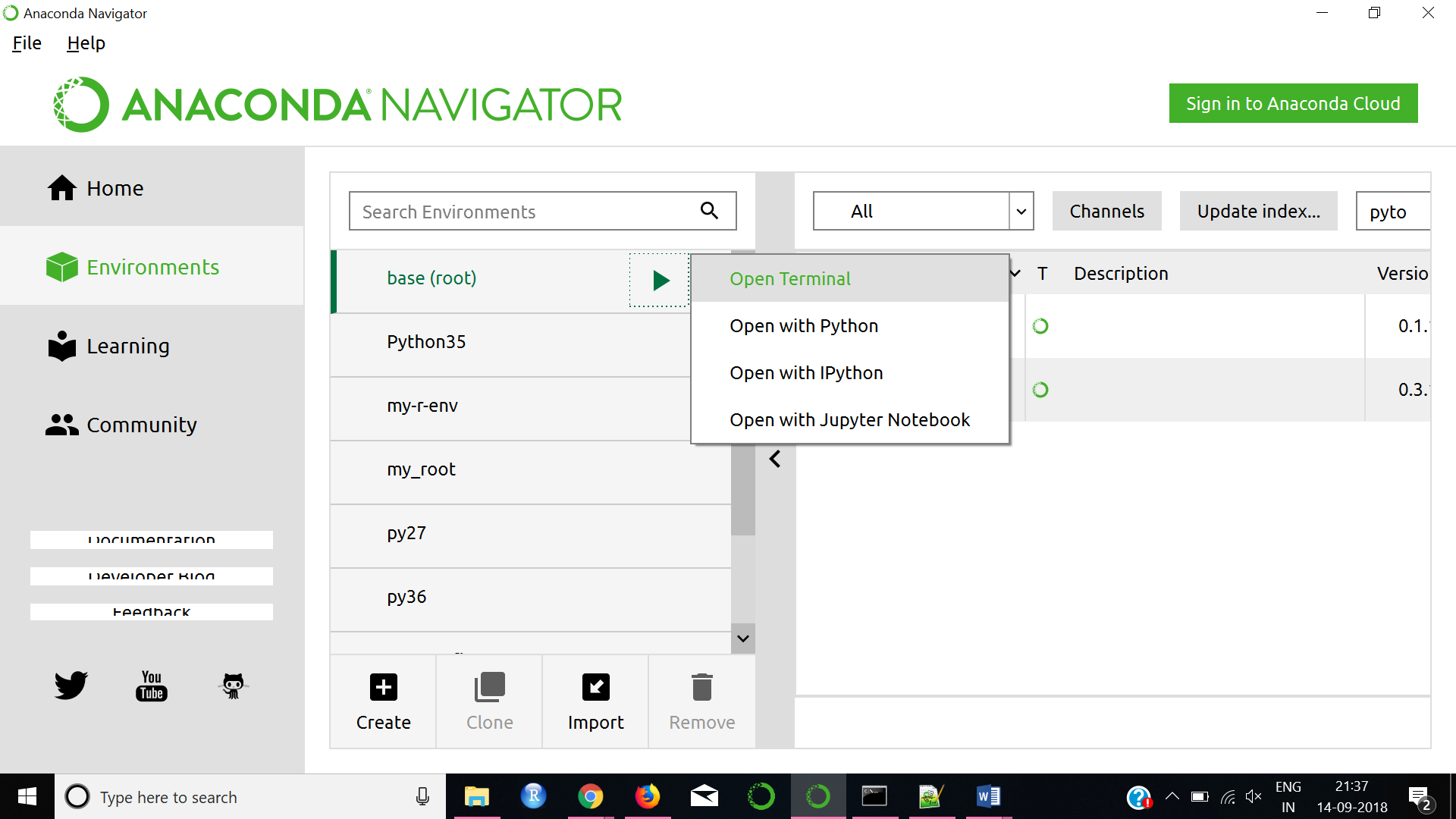
PyTorch is a machine learning and deep learning tool developed by Facebook’s artificial intelligence division to process large scale image analysis, including object detection, segmentation and classification. Not only limited to the above tasks it can be used along with other framework to implement complex algorithms. It is written using Python, C++ language. To process large scale computations in a GPU environment the programming languages should be modified accordingly. PyTorch provides a great framework to write function that automatically runs in a GPU environment.

## PyTorch Installation

Installation steps of PyTorch is quite simple. In a windows or Linux or MacOS it is very simple to install given that you are familiar with Anaconda and Conda environment for managing packages. The following steps display how to install PyTorch in windows/MacOS/Linux environment.

Step 1:

Open Anaconda navigator and go to the environment page as displayed in the following screenshot.



Step 2:

Open the terminal and terminal and type the following:

conda install -c peterjc123 pytorch

Step 3:

Launch Jupyter and open the IPython Notebook

Step 4:

Type the following command to check whether the PyTorch is installed or not.

from \_\_future\_\_ import print\_function

import torch

Step 5:

Check the version of the PyTorch.

A screenshot of a cell phone

Description generated with very high confidence

PyTorch has various components, let’s get familiar with those components.

* Torch: it has similar functionality like NumPy with GPU support
* Autograd: ``torch.autograd`` provides classes, methods and functions for implementing automatic differentiation of arbitrary scalar valued functions. It requires minimal changes to the existing code - you only need to declare, class :`Tensor`s for which gradients should be computed with the ``requires\_grad=True`` keyword.
* Nn: it is a neural network library in PyTorch
* Optim: this provides a lot of optimization algorithm that can be used for minimization and maximization of functions
* Multiprocessing: this is a useful library for memory sharing between multiple tensors
* Utils: it has utility functions to load data, and other functions

Now we are ready to proceed with the chapter.

## Problem

The data structure used in PyTorch is graph based and tensor based, hence it is important to understand basic operations and defining tensors.

## Solution

The solution to the above problem would be doing some practice on the tensors and its operations, which includes many examples that uses various operations. Though it is assumed that the user is familiar with PyTorch and Python basics, still a refresher on PyTorch is essential to create interest among the new users.

## How it Works

Now, let’s have a look at the following sample examples of tensors and basics of tensor operations including mathematical operations etc.

The object ‘x’ is a list and we can check whether an object in python is a tensor object or not by using the following syntax. Typically Is\_tensor function is used to check and is\_storage function is being used to check whether the object is stored as tensor object or not.

A screenshot of a cell phone

Description generated with very high confidence

Now, let’s create another object, which contains random numbers from torch similar to numpy library and we can check the tensor and storage type.

A screenshot of a cell phone

Description generated with very high confidence

The object ‘y’ is a tensor; however, it is not stored, also to check the total number of elements in the input tensor object the numerical element function can be used. Another example of creating zero values in a 2D tensor and counting the numerical elements in it, the following script can be used.

A screenshot of a cell phone

Description generated with very high confidence

Like NumPy operations, the eye function creates a diagonal matrix, of which the diagonal elements have 1 and off diagonal elements have zeros. Eye function can be manipulated by providing the shape option, the following example shows how to provide the shape parameter.

A screenshot of a cell phone

Description generated with high confidence

Linear space and points between the linear space can be created using tensor operations, let’s take an example of creating 25 points in a linear space starting from value 2 and ending with 10. The torch can read from numpy array format.

A screenshot of a cell phone

Description generated with high confidence

Like linear spacing, logarithmic spacing can be created.

A screenshot of a cell phone

Description generated with very high confidence

Random number generation is a common process in data science, to generate or gather sample data points in a space to simulate some structure in the data. Random numbers can be generated from a statistical distribution, can be generated from any two values and from a pre-defined distribution as well. Like NumPy function the random number can be generated using the following example. Uniform distribution can be defined as a distribution where each outcome has equal probability of happening. Hence the event probabilities are constant.

A screenshot of a cell phone

Description generated with very high confidence

The following script shows how the random number from two values 0 and 1 be selected and the result tensor can be reshaped to create a (4,5) matrix. The random numbers from a normal distribution with arithmetic mean 0 and standard deviation 1, can also be created as below.

A screenshot of a cell phone

Description generated with very high confidence

To select random values from a range of values using random permutation, requires defining the range first and this range can be create using arange function. While using arrange function it is required to define the step size, which will place all the values in a equal distance space. By default the step size is 1.

A screenshot of a cell phone

Description generated with high confidence

To find out the minimum and maximum values in a tensor of 1D, the argmin and argmax can be used, and also need to mention the dimension, if the input is a matrix, to search minimum values along rows or columns.

A screenshot of a cell phone

Description generated with very high confidence

Either it is a row or column, if it has a single dimension it is called 1D tensor and if the input is a matrix where the number of rows and columns are also present, it is called 2D tensors. If more than 2D is present it is called multidimensional tensor.

A screenshot of a cell phone

Description generated with very high confidence

Now, let’s create a sample 2D tensor and perform indexing and concatenation of more than 2D tensors using concat operation on the tensors.

A screenshot of a cell phone

Description generated with very high confidence

The sample tensor ‘x’ can be used in 3D ways as well. Again there are two different options to create three dimensional tensors, the third dimension can be extended over rows or over columns.

A screenshot of a cell phone

Description generated with very high confidence

A tensor can be split between multiple chunks, those small chunks can be created along dim rows and dim columns. The following example shows a sample tensor a of size (4,4), the chunk is created using the third argument in the function as 0 or 1.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidence

Gather function collects elements from a tensor and places it in another tensor using an index argument. The index position is determined by LongTensor function in PyTorch.

A screenshot of a cell phone

Description generated with high confidence

Either LongTensor can be used or index select function can be used in order to fetch relevant values from a tensor, the following sample code shows two options, selection along rows and selection along columns, if the second argument is 0 it is for rows and if it is 1 then it is along the columns.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

It is a common practice to check non-missing values in a tensor, the objective is to identify non-zero elements in a large tensor.

A screenshot of a cell phone

Description generated with very high confidence

Restructuring the input tensors into smaller tensor not only fasten the calculation process, but also help in distributed computing. Split function can be used to split a long tensor into smaller tensors given the length of the chunk decided a prior.

A screenshot of a cell phone

Description generated with very high confidence

Now, let’s have a look at the examples of how the input tensor can be resized given the computational difficulty. The transpose function primarily used to reshape the tensors. There are two ways of writing the transpose function, .t and .transpose.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

Unbind function is used to remove a dimension from a tensor, to remove the dimension row, value 0 need to be passed and to remove a column value 1 need to be passed.

A screenshot of a cell phone

Description generated with very high confidence

Mathematical functions are backbone of implementing any algorithm in PyTorch, therefore it is very much needed to go through functions that help in performing arithmetic based operations. A scalar is a single value, a tensor 1D is a row like NumPy. The scalar multiplication and addition with a 1D tensor can be done using add and mul function.

A screenshot of a cell phone

Description generated with very high confidence

The following script shows the scalar addition and multiplication with a tensor.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

Combined mathematical operations, such as expressing the linear equations as tensor operations can be done using the sample script as below. Here we are going to express the outcome object ‘y’ is a linear combination of beta values times independent object ‘x’ plus the constant term.

A screenshot of a cell phone

Description generated with very high confidence

Output = Constant + (beta \* Independent)

A screenshot of a cell phone

Description generated with high confidence

Like NumPy operations, the tensor values if required to be rounded up, by either using ceiling or flooring function can be done using the following syntax.

A screenshot of a cell phone

Description generated with high confidence

To limit the values of any tensor within a certain range can be done using the minimum and maximum argument and using the function called ‘clamp’. The same function can be used either to apply minimum and maximum in parallel or any one of them only to any tensor, be it 1D or 2D. the 1D is much simpler version, the following example shows the implementation in a 2D scenario.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

How to get the exponential of a tensor, and how to get the fractional portion of the tensor if it has decimal places and defined as floating data type.

A screenshot of a cell phone

Description generated with high confidence

The following syntax explains the logarithmic of the values in a tensor, the values with negative sign are converted to nan. The power function can be used to compute the exponential of any value in a tensor.

A screenshot of a cell phone

Description generated with high confidence

To compute the transformation functions such as sigmoid, hyperbolic tangent, radial basis function etc. which are most commonly used transfer functions in dep learning, it is required to construct those tensors. The following sample script shows how to create a sigmoid function and apply it on a tensor.

A screenshot of a cell phone

Description generated with very high confidence

**Conclusion:**

This chapter is just a refresher for the people who have prior experience into PyTorch, Python and it is a basic building block for the people who are new to the PyTorch framework. Before starting the advanced topics, it is important to make the reader familiar with the terminology and basic syntaxes etc. the next chapter would be on using PyTorch how to implement probabilistic models, which includes creation of random variable, application of statistical distributions, and its applications and making statistical inference.