Chapter 2: Probability Distributions using PyTorch

**Introduction**

Probability and random variable are integral part of computation in a graph computing platform like PyTorch. Understanding of the probability and associated concepts are essential. In this chapter we are going to cover probability distributions, and implementation using PyTorch and to interpret the results from the test. In probability and statistics, the random variable is also known as a stochastic variable whose outcome is dependent on a purely stochastic phenomenon, or random phenomenon. There are different types of probability distribution such as normal distribution, binomial distribution, multinomial distribution, Bernoulli distribution etc. each statistical distribution has its own properties.

## Problem:

Weight initialization is an important task in training a neural network and any kind of deep learning models such as convolutional neural network (CNN), deep neural network (DNN) and recurrent neural network (RNN). The question always remains how we initialize the weights.

## Solution:

Weight initialization can be done by using various methods, including random weight initialization, weight initialization based on a distribution, if it is distribution, it can be done using uniform distribution, Bernoulli distribution, multinomial distribution and normal distribution. How to do it using PyTorch is explained below.

## How It works:

For execution of neural network, a set of initial weight need to be passed to the back-propagation layer to compute the loss function and hence the accuracy can be calculated. The selection of a method depends on data type, kind of task and optimization required for the model. Here we are going to look at all types of approaches to initialize weights.

If the use case requires to reproduce the same set of results again to maintain consistency, then manual seed need to be set.

A screenshot of a cell phone

Description generated with high confidence

The seed value can be customized, the random number is getting generated purely based on chance. The random numbers can also be generated from a statistical distribution. The probability density function of the continuous uniform distribution is defined by the following formula.

A close up of a clock

Description generated with high confidence

Where the function of x has two points a and b, a is the start point and b is the end. In continuous uniform distribution each number has equal chance of getting selected. In the following example start is 0 and end is 1, between those two digits all the 16 elements are selected randomly.

A screenshot of a cell phone

Description generated with very high confidence

In statistics the Bernoulli distribution is considered as the discrete probability distribution, which has two possible outcomes, if the event happens then the value will be 1 and if the event does not happen then the value would be 0. For discrete probability distribution we calculate probability mass function instead of probability density function. The probability mass function would look like as below formula.

A picture containing object

Description generated with high confidence

From the Bernoulli distribution we will create sample tensors by considering the uniform distribution of size 4 and 4 in a matrix format as below.

A screenshot of a cell phone

Description generated with very high confidence

Generation of sample random values from the multinomial distribution is defined by the following script. From a multinomial distribution we can choose with replacement and without replacement. By default, multinomial function picks up without replacement and returns the result as index position for the tensors. If we need to run it with replacement, then we need to specify that while sampling.

A screenshot of a cell phone

Description generated with very high confidence

Sampling from multinomial distribution with replacement returns the tensors index values.

A screenshot of a cell phone

Description generated with very high confidence

The weight initialization from the normal distribution is a method which is used in fitting a neural network, fitting a deep neural network and CNN and RNN too. Let’s have a look at the process of creating a set of random weights generated from a normal distribution.

A screenshot of a cell phone

Description generated with high confidence

## Problem:

What is variable in PyTorch, how it is defined and what is random variable in PyTorch.

## Solution:

In PyTorch the algorithms are represented as computational graph. A variable is considered as a representation around the tensor object, corresponding gradients and a reference to the function from where it is created. For the sake of simplicity gradients are considered as slope of the function. The slope of the function can be computed by the derivative of the function with respect to the parameters that are present in the function. For example, in case of linear regression (Y = W\*X + Alpha) representation the variable would look like the one shown in the graph below. PyTorch variable is basically a node in a computational graph, which stores data and gradient.

+

X

## How It works:

An example of how a variable is used to create a computational graph is displayed below. There are three variable objects around tensors x1, x2 and x3 with random points generated from a= 12 and b= 23, then the graph computation involves only multiplication and addition and the final result with the gradient is displayed below.

A screenshot of a cell phone

Description generated with very high confidence

## Problem:

How to compute basic statistic such as mean, median, mode etc. from Torch tensor.

## Solution:

Computation of basic statistics using PyTorch will enable the user to apply probability distributions and statistical tests to make inferences from data. Though the Torch functionality will be like that of Numpy, but Torch functions will have GPU acceleration, hence let’s have a look at the functions to create basic statistic.

## How It works:

From 1D tensor the mean computation is simple to write, however for 2D tensor extra argument need to be passed as mean or median or mode computation across which dimension required, need to be specified.

A screenshot of a cell phone

Description generated with very high confidence

Median, mode and standard deviation Computation also can be written in the same way.

A screenshot of a cell phone

Description generated with very high confidence

Standard deviation shows the deviation from the measures of central tendency, which is an indication of consistency of data/variable. It shows whether there is enough fluctuation in data or not.

A screenshot of a cell phone

Description generated with very high confidence

## Problem:

How to compute basic gradients from the sample tensors using PyTorch.

## Solution:

We are going to consider a sample dataset, where two variables x and y both are present, with initial weight given, can we computationally get the gradients after each iteration, let’s have a look at the example.

## How It works:

X\_data and y\_data both are lists and to compute the gradient of the two data lists, it requires computation of a loss function, a forward pass and running the stuff in a loop.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidence

A close up of text on a white background

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

## Problem:

How to compute or perform operations based on variables such as matrix multiplication.

## Solution:

Tensors will be wrapped within the variable, which has three beautiful properties, such as requires grad, volatile and gradient.

## How It works:

Let’s create two sample variables around the tensors and perform matrix multiplication.

A screenshot of a cell phone

Description generated with high confidence

A screenshot of a cell phone

Description generated with very high confidence

**Conclusion:**