Deep Learning With Python book

Contents

[Chapter 1: Welcome 4](#_Toc47351261)

[References 7](#_Toc47351262)

# Chapter 1: Welcome

Deep knowledge of concepts is not required, they can be picked up as necessary on the go. This book will use Keras API library, which wraps around Theano and TensorFlow. If you don’t know something pick it up.

# Chapter 2 Introduction to Theano

It is a numerical library.

It can be used to define tensors—things that can hold other sets of objects with a relationship to each other. In mathematics, a **tensor** is an algebraic object that describes a (multilinear) relationship between sets of algebraic objects related to a vector space.

Related links: [Tensor](https://en.wikipedia.org/wiki/Tensor),

(Wikipedia),

Theano can be used to compile mathematical expressions beforehand, instead of compiling them later, and then evaluate them. This can result in massive optimizations.

Pre-defining computation to be compiled for efficiency may be scaled up to large vector and matrix operations required for deep learning. At it’s heart Theano is a compiler for mathematical expressions in Python. It knows how to take your structures and turn them into very efficient code that uses NumPy, efficient native libraries like BLAS and native code to run as fast as possible on CPUs or GPUs.

It uses a host of clever code optimizations to squeeze as much performance as possible from your hardware.

## Example

In this example we define two symbolic floating-point variables a and b. We define an expression that uses these variables (c = a + b). We then compile this symbolic expression into a function using Theano that we can use later. Finally, we use our compiled expression by plugging in some real values and performing the calculation using efficient compiled Theano code under the covers.

""" A theano example, as found in chapter 2 """

import theano

from theano import tensor

# Define tensors as 2 symbolic floating point scalars

a = tensor.dscalar()

b = tensor.dscalar()

# create a symbolic expression and store it in c

c = a + b

# convert it into a callable object, a function, that takes a,b and returns the

# result

f = theano.function([a,b], c)

# call the function, by passing a = 10.5, b = 25.5, and evaluate c

result = f(10.5,25.5)

print(result)

## Result

'(Anaconda3)'C:\>python therano\_example.py

36.0

## A function in Theano is a callable expression.

def function(inputs, outputs=None, mode=None, updates=None, givens=None, no\_default\_updates=False, accept\_inplace=False, name=None, rebuild\_strict=True, allow\_input\_downcast=None, profile=None, on\_unused\_input=None)

Return a :class:callable object <theano.compile.function\_module.Function> that will calculate outputs from inputs.

#### Parameters

inputs : list of either Variable or In instances.

Function parameters, these are not allowed to be shared variables.

outputs : list or dict of Variables or Out instances.

If it is a dict, the keys must be strings. Expressions to compute.

mode : string or Mode instance.

#### Returns

:class:theano.compile.function\_module.Function instance

A callable object that will compute the outputs (given the inputs) and   
update the implicit function arguments according to the `updates`.

Internal documentation:

# References

**There are no sources in the current document.**