EBook Gratis

APRENDIZAJE theano

Free unaffiliated eBook created from
Stack Overflow contributors.

#theano

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: theano

It is an unofficial and free theano ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official theano.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con theano

Observaciones

Theano es una biblioteca de Python, que maneja la definición y evaluación de expresiones simbólicas sobre variables tensoriales. Tiene varias aplicaciones, pero la más popular es el aprendizaje profundo.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar theano.

Instalando Theano y configurando la GPU en Ubuntu 14.04

Puede usar las siguientes instrucciones para instalar Theano y configurar la GPU (suponga un Ubuntu 14.04 recién instalado):

```
# Install Theano
sudo apt-get install python-numpy python-scipy python-dev python-pip python-nose g++
libopenblas-dev git
sudo pip install Theano

# Install Nvidia drivers, CUDA and CUDA toolkit, following some instructions from
http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html
wget http://developer.download.nvidia.com/compute/cuda/7.5/Prod/local_installers/cuda-repo-
ubuntu1404-7-5-local_7.5-18_amd64.deb # Got the link at https://developer.nvidia.com/cuda-
downloads
sudo dpkg -i cuda-repo-ubuntu1404-7-5-local_7.5-18_amd64.deb
sudo apt-get update
sudo apt-get install cuda

sudo reboot
```

En ese punto, la ejecución de `nvidia-smi` debería funcionar, pero la ejecución de `nvcc` no funcionará.

```
# Execute in console, or (add in ~/.bash_profile then run "source ~/.bash_profile"):
export PATH=/usr/local/cuda-7.5/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-7.5/lib64:$LD_LIBRARY_PATH
```

En ese punto, tanto `nvidia-smi` como `nvcc` deberían funcionar.

Para probar si Theano es capaz de usar la GPU:

Copie y pegue lo siguiente en `gpu_test.py` :

```
# Start gpu_test.py
```

```
# From http://deeplearning.net/software/theano/tutorial/using_gpu.html#using-gpu
from theano import function, config, shared, sandbox
import theano.tensor as T
import numpy
import time

vlen = 10 * 30 * 768  # 10 x #cores x # threads per core
iters = 1000

rng = numpy.random.RandomState(22)
x = shared(numpy.asarray(rng.rand(vlen), config.floatX))
f = function([], T.exp(x))
print(f.maker.fgraph.toposort())
t0 = time.time()
for i in xrange(iters):
    r = f()
t1 = time.time()
print("Looping %d times took %f seconds" % (iters, t1 - t0))
print("Result is %s" % (r,))
if numpy.any([isinstance(x.op, T.Elemwise) for x in f.maker.fgraph.toposort()]):
    print('Used the cpu')
else:
    print('Used the gpu')
# End gpu_test.py
```

y ejecutarlo:

```
THEANO_FLAGS='mode=FAST_RUN,device=gpu,floatX=float32' python gpu_test.py
```

que debe devolver:

```
f@f-Aurora-R4:~$ THEANO_FLAGS='mode=FAST_RUN,device=gpu,floatX=float32' python gpu_test.py
Using gpu device 0: GeForce GTX 690
[GpuElemwise{exp,no_inplace}(<CudaNdarrayType(float32, vector)>),
HostFromGpu(GpuElemwise{exp,no_inplace}.0)]
Looping 1000 times took 0.658292 seconds
Result is [ 1.23178029  1.61879349  1.52278066 ...,  2.20771813  2.29967761
  1.62323296]
Used the gpu
```

Para conocer tu versión CUDA:

```
nvcc -V
```

Ejemplo:

```
username@server:~$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2015 NVIDIA Corporation
Built on Tue_Aug_11_14:27:32_CDT_2015
Cuda compilation tools, release 7.5, V7.5.17
```

Añadiendo cuDNN

Para agregar cuDNN (instrucciones de http://deeplearning.net/software/theano/library/sandbox/cuda/dnn.html) :

1. Descargue cuDNN desde https://developer.nvidia.com/rdp/cudnn-download (necesita registro, que es gratuito)
2. `tar -xvf cudnn-7.0-linux-x64-v3.0-prod.tgz`
3. Haz una de las siguientes

Opción 1: copie los archivos `*.h` en `CUDA_ROOT/include` y los archivos `*.so* CUDA_ROOT/lib64 *.so*` en `CUDA_ROOT/lib64` (de forma predeterminada, `CUDA_ROOT` es `/usr/local/cuda` en Linux).

```
sudo cp cuda/lib64/* /usr/local/cuda/lib64/
sudo cp cuda/include/cudnn.h /usr/local/cuda/include/
```

Opcion 2:

```
export LD_LIBRARY_PATH=/home/user/path_to_CUDNN_folder/lib64:$LD_LIBRARY_PATH
export CPATH=/home/user/path_to_CUDNN_folder/include:$CPATH
export LIBRARY_PATH=/home/user/path_to_CUDNN_folder/lib64:$LD_LIBRARY_PATH
```

Por defecto, Theano detectará si puede usar cuDNN. Si es así, lo utilizará. Si no, las optimizaciones de Theano no introducirán las operaciones cuDNN. Entonces, Theano seguirá funcionando si el usuario no los introdujo manualmente.

Para obtener un error si Theano no puede usar cuDNN, use este indicador de Theano: `optimizer_including=cudnn` .

Ejemplo:

```
THEANO_FLAGS='mode=FAST_RUN,device=gpu,floatX=float32,optimizer_including=cudnn' python
gpu_test.py
```

Para conocer tu versión cuDNN:

```
cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2
```

Añadiendo CNMeM

La biblioteca CNMeM es una "biblioteca simple para ayudar a los marcos de Deep Learning a administrar la memoria CUDA".

```
# Build CNMeM without the unit tests
git clone https://github.com/NVIDIA/cnmem.git cnmem
cd cnmem
mkdir build
cd build
sudo apt-get install -y cmake
cmake ..
make
```

```
# Copy files to proper location
sudo cp ../include/cnmem.h /usr/local/cuda/include
sudo cp *.so /usr/local/cuda/lib64/
cd ../..
```

Para usar con Theano, necesitas agregar la bandera `lib.cnmem` . Ejemplo:

```
THEANO_FLAGS='mode=FAST_RUN,device=gpu,floatX=float32,lib.cnmem=0.8,optimizer_including=cudnn'
python gpu_test.py
```

La primera salida del script debe ser:

```
Using gpu device 0: GeForce GTX TITAN X (CNMeM is enabled with initial size: 80.0% of memory,
cuDNN 5005)
```

`lib.cnmem=0.8` significa que puede usar hasta el 80% de la GPU.

Se ha informado que CNMeM ofrece algunas mejoras de velocidad interesantes, y cuenta con el apoyo de Theano, Torch y Caffee.

Theano - fuente 1 :

> La velocidad depende de muchos factores, como las formas y el modelo en sí. La velocidad aumenta de 0 a 2 veces más rápido.

Theano - fuente 2 :

> Si no cambia la bandera de Theano allow_gc, puede esperar un 20% de aceleración en la GPU. En algunos casos (modelos pequeños), vimos una aceleración del 50%.

Problemas comunes:

- Importando theano: AttributeError: el objeto 'módulo' no tiene atributo 'find_graphviz'

Ejecutando Theano en múltiples núcleos de CPU

Puede ejecutar Theano en múltiples núcleos de CPU con el `OMP_NUM_THREADS=[number_of_cpu_cores]` .

Ejemplo:

```
OMP_NUM_THREADS=4 python gpu_test.py
```

El script `theano/misc/check_blas.py` genera información sobre qué BLAS se utiliza:

```
cd [theano_git_directory]
OMP_NUM_THREADS=4 python theano/misc/check_blas.py
```

Tu primer programa de theano

En este ejemplo, compilaremos funciones que calculen la suma y la diferencia dados dos números reales.

```
from __future__ import print_function
import theano
import theano.tensor as T

#define two symbolic scalar
s_x = T.fscalar()
s_y = T.fscalar()

#compute something
s_sum = s_x + s_y
s_diff = s_x - s_y

#compile a function that adds two number
#theano will call system compiler at here
fn_add = theano.function(inputs=[s_x, s_y], outputs=s_sum)
fn_diff = theano.function(inputs=[s_x, s_y], outputs=s_diff)

#call the compiled functions
print(fn_add(2., 2.)) #4.
print(fn_diff(2., 2.)) #0.
```

Lea Empezando con theano en línea: https://riptutorial.com/es/theano/topic/5439/empezando-con-theano

Capítulo 2: Loops con theano

Examples

Uso básico de escaneo

`scan` se usa para llamar a la función varias veces sobre una lista de valores, la función puede contener el estado.

sintaxis de `scan` (a partir de theano 0.9):

```
scan(
    fn,
    sequences=None,
    outputs_info=None,
    non_sequences=None,
    n_steps=None,
    truncate_gradient=-1,
    go_backwards=False,
    mode=None,
    name=None,
    profile=False,
    allow_gc=None,
    strict=False)
```

Esto puede ser muy confuso a primera vista. Explicaremos varios usos de `scan` básicos pero importantes en varios ejemplos de código.

Los siguientes ejemplos de código suponen que ha ejecutado importaciones:

```
import numpy as np
import theano
import theano.tensor as T
```

`sequences` - mapea una función sobre una lista

En el caso más simple, el escaneo simplemente asigna una función pura (una función sin estado) a una lista. Las listas se especifican en el argumento de `sequences`

```
  s_x = T.ivector()
  s_y, _ = theano.scan(
      fn = lambda x:x*x,
      sequences = [s_x])
  fn = theano.function([s_x], s_y)
  fn([1,2,3,4,5]) #[1,4,9,16,25]
```

La `scan` notas tiene dos valores de retorno, el primero es la lista resultante y el último es la actualización del valor de estado, que se explicará más adelante.

`sequences` - comprimir una función sobre una lista

Casi lo mismo que arriba, solo da a las `sequences` argumento una lista de dos elementos. El orden de los dos elementos debe coincidir con el orden de los argumentos en `fn`

```
s_x1 = T.ivector()
s_x2 = T.ivector()
s_y, _ = theano.scan(
    fn = lambda x1,x2:x1**x2,
    sequences = [s_x1, s_x2])
fn = theano.function([s_x], s_y)
fn([1,2,3,4,5],[0,1,2,3,4]) #[1,2,9,64,625]
```

`outputs_info` - Acumula una lista

La acumulación implica una variable de estado. Las variables de estado necesitan valores iniciales, que se especificarán en el parámetro `outputs_info` .

```
s_x = T.ivector()
v_sum = th.shared(np.int32(0))
s_y, update_sum = theano.scan(
    lambda x,y:x+y,
    sequences = [s_x],
    outputs_info = [s_sum])
fn = theano.function([s_x], s_y, updates=update_sum)

v_sum.get_value() # 0
fn([1,2,3,4,5]) # [1,3,6,10,15]
v_sum.get_value() # 15
fn([-1,-2,-3,-4,-5]) # [14,12,9,5,0]
v_sum.get_value() # 0
```

Ponemos una variable compartida en `outputs_info` , lo que provocará actualizaciones de retorno de `scan` a nuestra variable compartida, que luego se pueden poner en `theano.function` .

`non_sequences` y `n_steps` - órbita del mapa logístico `x -> lambda*x*(1-x)`

Puede dar entradas que no cambian durante la `scan` en el argumento `non_sequences` . En este caso, `s_lambda` es una variable que no cambia (pero NO una constante, ya que debe suministrarse durante el tiempo de ejecución).

```
s_x = T.fscalar()
s_lambda = T.fscalar()
s_t = T.iscalar()
s_y, _ = theano.scan(
    fn = lambda x,l: l*x*(1-x),
    outputs_info = [s_x],
    non_sequences = [s_lambda],
    n_steps = s_t
)
fn = theano.function([s_x, s_lambda, s_t], s_y)

fn(.75, 4., 10) #a stable orbit

#[ 0.75,  0.75,  0.75,  0.75,  0.75,  0.75,  0.75,  0.75,  0.75,  0.75]

fn(.65, 4., 10) #a chaotic orbit
```

```
#[ 0.91000003,  0.32759991,  0.88111287,  0.41901192,  0.97376364,
# 0.10219204,  0.3669953 ,  0.92923898,  0.2630156 ,  0.77535355]
```

Grifería - Fibonacci

Los estados / entradas pueden venir en múltiples pasos de tiempo. Esto se hace por:

- poniendo `dict(input=<init_value>, taps=<list of int>)` dentro del argumento de `sequences` .
- poniendo `dict(initial=<init_value>, taps=<list of int>)` dentro del argumento `outputs_info` .

En este ejemplo, usamos dos `outputs_info` en `outputs_info` para calcular la relación de recurrencia `x_n = x_{n-1} + x_{n-2}` .

```
s_x0 = T.iscalar()
s_x1 = T.iscalar()
s_n = T.iscalar()
s_y, _ = theano.scan(
    fn = lambda x1,x2: x1+x2,
    outputs_info = [dict(initial=T.join(0,[s_x0, s_x1]), taps=[-2,-1])],
    n_steps = s_n
)
fn_fib = theano.function([s_x0, s_x1, s_n], s_y)
fn_fib(1,1,10)
# [2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```

mapa theano y reducir

`theano.map` y `theano.scan_module.reduce` son envolturas de `theano_scan` . Pueden ser vistos como versión para discapacitados de `scan` . Puede ver la sección **Uso de escaneo básico** para referencia.

```
import theano
import theano.tensor as T
s_x = T.ivector()
s_sqr, _ = theano.map(
    fn = lambda x:x*x,
    sequences = [s_x])
s_sum, _ = theano.reduce(
    fn = lambda: x,y:x+y,
    sequences = [s_x],
    outputs_info = [0])
fn = theano.function([s_x], [s_sqr, s_sum])
fn([1,2,3,4,5]) #[1,4,9,16,25], 15
```

haciendo bucle while

A partir de theano 0.9, mientras que los bucles se pueden hacer a través de `theano.scan_module.scan_utils.until` . Para usarlo, debe regresar `until` objeto en `fn` de `scan` .

En el siguiente ejemplo, creamos una función que verifica si un número complejo está dentro del

conjunto de Mandelbrot. Un número complejo `z_0` está dentro del conjunto de mandelbrot si la serie `z_{n+1} = z_{n}^2 + z_0` no converge.

```
MAX_ITER = 256
BAILOUT = 2.
s_z0 = th.cscalar()
def iterate(s_i_, s_z_, s_z0_):
    return [s_z_*s_z_+s_z0_,s_i_+1], {}, until(T.abs_(s_z_)>BAILOUT)
(_1, s_niter), _2 = theano.scan(
    fn = iterate,
    outputs_info = [0, s_z0],
    non_sequences = [s_z0],
    n_steps = MAX_ITER
)
fn_mandelbrot_iters = theano.function([s_z0], s_niter)
def is_in_mandelbrot(z_):
    return fn_mandelbrot_iters(z_)>=MAX_ITER

is_in_mandelbrot(0.24+0.j) # True
is_in_mandelbrot(1.j) # True
is_in_mandelbrot(0.26+0.j) # False
```

Lea Loops con theano en línea: https://riptutorial.com/es/theano/topic/7609/loops-con-theano

Creditos

S. No	Capítulos	Contributors
1	Empezando con theano	Alleo, Community, Franck Dernoncourt, Kh40tiK
2	Loops con theano	Kh40tiK