EBook Gratuito

APPRENDIMENTO theano

Free unaffiliated eBook created from
Stack Overflow contributors.

#theano

Sommario

Di

Capitolo 1: Iniziare con theano

Osservazioni

Theano è una libreria python che gestisce la definizione e la valutazione di espressioni simboliche su variabili tensoriali. Ha varie applicazioni, ma il più popolare è l'apprendimento profondo.

Examples

Installazione o configurazione

Istruzioni dettagliate su come installare o installare Theano.

Installazione di Theano e configurazione della GPU su Ubuntu 14.04

È possibile utilizzare le seguenti istruzioni per installare Theano e configurare la GPU (si presume che Ubuntu 14.04 sia installato di recente:

```
# 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
```

A quel punto, l'esecuzione di `nvidia-smi` dovrebbe funzionare, ma l'esecuzione di `nvcc` non funzionerà.

```
# 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
```

A quel punto, dovrebbero funzionare sia `nvidia-smi` che `nvcc` .

Per verificare se Theano è in grado di utilizzare la GPU:

Copia-incolla quanto segue in `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
```

ed eseguilo:

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

che dovrebbe restituire:

```
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
```

Per conoscere la tua versione CUDA:

```
nvcc -V
```

Esempio:

```
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
```

Aggiunta di cuDNN

Per aggiungere cuDNN (istruzioni da http://deeplearning.net/software/theano/library/sandbox/cuda/dnn.html) :

1. Scarica cuDNN da https://developer.nvidia.com/rdp/cudnn-download (è necessaria la registrazione, che è gratuita)
2. `tar -xvf cudnn-7.0-linux-x64-v3.0-prod.tgz`
3. Effettuare una delle seguenti operazioni

Opzione 1: copia i `*.h` in `CUDA_ROOT/include` e i file `*.so*` in `CUDA_ROOT/lib64` (per impostazione predefinita, `CUDA_ROOT` è `/usr/local/cuda` su Linux).

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

Opzione 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
```

Per impostazione predefinita, Theano rileva se può utilizzare cuDNN. Se è così, lo userà. Altrimenti, le ottimizzazioni di Theano non introdurranno le operazioni cuDNN. Quindi Theano funzionerà ancora se l'utente non li ha introdotti manualmente.

Per ottenere un errore se Theano non può usare cuDNN, usa questo flag di Theano: `optimizer_including=cudnn` .

Esempio:

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

Per conoscere la tua versione cuDNN:

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

Aggiunta di CNMeM

La libreria CNMeM è una "libreria semplice per aiutare i framework Deep Learning a gestire 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 ../..
```

Per usarlo con Theano, è necessario aggiungere il flag `lib.cnmem` . Esempio:

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

Il primo output dello script dovrebbe essere:

```
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 che può utilizzare fino all'80% della GPU.

È stato segnalato che CNMeM offre alcuni interessanti miglioramenti della velocità ed è supportato da Theano, Torch e Caffee.

Theano - fonte 1 :

> La velocità dipende da molti fattori, come le forme e il modello stesso. La velocità aumenta da 0 a 2x più veloce.

Theano - source 2 :

> Se non cambi il flag di Theano allow_gc, puoi aspettarti un aumento del 20% della GPU. In alcuni casi (modelli piccoli), abbiamo rilevato un aumento del 50%.

Problemi comuni:

- Importazione di theano: AttributeError: l'oggetto 'module' non ha attributo 'find_graphviz'

Esecuzione di Theano su più core CPU

È possibile eseguire Theano su più core CPU con il `OMP_NUM_THREADS=[number_of_cpu_cores]` .

Esempio:

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

Lo script `theano/misc/check_blas.py` informazioni su quale BLAS è utilizzato:

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

Il tuo primo programma theano

In questo esempio, compileremo funzioni che calcolano somma e differenza in base a due numeri reali.

```
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.
```

Leggi Iniziare con theano online: https://riptutorial.com/it/theano/topic/5439/iniziare-con-theano

Capitolo 2: Loop con theano

Examples

Utilizzo di scansione di base

`scan` viene utilizzata per chiamare la funzione più volte su un elenco di valori, la funzione può contenere stato.

sintassi di `scan` (a partire da 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)
```

Questo può essere molto confuso a prima vista. Spiegheremo diversi semplici ma importanti utilizzi di `scan` in più esempi di codice.

I seguenti esempi di codice presuppongono che tu abbia eseguito le importazioni:

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

`sequences` : mappare una funzione su un elenco

Nel caso più semplice, la scansione esegue solo il mapping di una funzione pura (una funzione senza stato) ad un elenco. Le liste sono specificate nell'argomento delle `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` note ha due valori di ritorno, il primo è l'elenco risultante e il secondo è l'aggiornamento al valore dello stato, che verrà spiegato in seguito.

`sequences` - Zip una funzione su un elenco

Quasi come sopra, basta dare alle `sequences` un elenco di due elementi. L'ordine dei due elementi deve corrispondere all'ordine degli argomenti in `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` - Accumula una lista

L'accumulo implica una variabile di stato. Le variabili di stato necessitano di valori iniziali, che devono essere specificati nel parametro `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
```

`outputs_info` una variabile condivisa in `outputs_info` , ciò causerà aggiornamenti della `scan` ritorno alla nostra variabile condivisa, che può quindi essere inserita in `theano.function` .

`non_sequences` e `n_steps` - Orbita della mappa logistica `x -> lambda*x*(1-x)`

Puoi dare input che non cambiano durante la `scan` in argomento `non_sequences` . In questo caso `s_lambda` è una variabile non variabile (ma NON una costante poiché deve essere fornita durante il runtime).

```
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]
```

Rubinetti - Fibonacci

stati / input possono arrivare in più timestep. Questo è fatto da:

- mettendo `dict(input=<init_value>, taps=<list of int>)` all'interno dell'argomento delle `sequences` .
- mettendo `dict(initial=<init_value>, taps=<list of int>)` all'interno dell'argomento `outputs_info` .

In questo esempio, usiamo due tocchi in `outputs_info` per calcolare la relazione di ricorrenza `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]
```

theano mappa e ridurre

`theano.map` e `theano.scan_module.reduce` sono wrapper di `theano_scan` . Possono essere visti come versione di `scan` disabili. È possibile visualizzare la sezione **sull'uso della scansione di base** come riferimento.

```
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
```

fare mentre ciclo

A partire da theano 0.9, mentre i loop possono essere eseguiti tramite `theano.scan_module.scan_utils.until` . Per utilizzare, è necessario tornare `until` all'oggetto in `fn` di `scan` .

Nell'esempio seguente, costruiamo una funzione che controlla se un numero complesso si trova all'interno del set Mandelbrot. Un numero complesso `z_0` è all'interno del set di `z_{n+1} = z_{n}^2 + z_0` se la serie `z_{n+1} = z_{n}^2 + z_0` non 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
```

Leggi Loop con theano online: https://riptutorial.com/it/theano/topic/7609/loop-con-theano

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con theano	Alleo, Community, Franck Dernoncourt, Kh40tiK
2	Loop con theano	Kh40tiK