ECBM E4040 Neural Networks and Deep Learning
2016 Fall
HOMEWORK #0 solution

PROBLEM #1:

(a) In the Jupyter notebook, execute the following two python code examples from the Deep Learning Tutorials, using a GPU:

- code/logistic_sgd.py
- code/convolutional_mlp.py

(b) Repeat the same using a CPU. See this link for how to switch between using GPU and CPU. A suggested way is to use the following command in the Jupyter notebook

Solution:(all the results are generated using the course provided AMI with type g2.2xlarge, exact number can be different)

- code/logistic_sgd.py :
  - GPU 6.1s
  - CPU 21.7s
- code/convolutional_mlp.py:
  - GPU 42.78m
  - CPU 400.12m

PROBLEM #2:

Perform the following operations in the Jupyter notebook:

1. Create a Theano single precision floating point vector **x**.
2. Create a RandomStream with a seed in theano, define a random 10 × 1 vector **a** and a random 10 × 1 vector **b** using this RandomStream, both from a uniform distribution.
3. Create a Theano function that performs the inner product **<x + a, b>**, use one shared variable in this function to record the randomly generated value of **a** and another shared variable to record the value of **b**.
4. Evaluate the above function one time using a 10 × 1 numpy vector of your choice as input **x**, and show that you can use the shared variables to evaluate the output of the function.
5. For verification repeat the same procedure using numpy operations (without using shared variables) and show that both methods give the same result.

Solution for Part 2

In [1]:
```
import theano
import theano.tensor as T
import numpy as np
from theano import shared
from theano.tensor.shared_randomstreams import RandomStreams
from theano import function
```

```
Using gpu device 0: GRID K520 (CNMeM is disabled, cuDNN Version is too old. Update to v5, was 3007.)
```

In [2]:
```
# creates vector x using Theano's predefined tensor type
x = T.vector('x', dtype='float32')
# checks the type of x(single precision floating point)
print x.type()
```

```
<TensorType(float32, vector)>
```

In [3]:
```
# creates a RandomStream object(a random number generator)
srng = RandomStreams(seed=234)
# draws random streams of 10x1 vectors from a normal distribution
a_n = srng.normal((10,))
b_n = srng.normal((10,))
f_a = function([], a_n, no_default_updates=True)
f_b = function([], b_n, no_default_updates=True)
# calls the function to generated uniform numbers.
a   = f_a()
b   = f_b()
# creates shared variables and constructs the inner product function.
a_record = shared(a)
b_record = shared(b)
z = T.dot(T.transpose(x+a_record),b_record)
f = theano.function([x], z)
```

In [4]:
```
# creates a vector input using numpy function
data_in = np.ones((10,), dtype='float32')
print data_in
```

```
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

In [5]:
```
# feed the input vector to both the theano and numpy functions to verify the output
output1 = f(data_in)
output2 = np.dot(np.transpose(data_in+a),b)
print output1
print output2
```

```
1.50674283504
1.50674
```

PROBLEM#3: Define a function in python fib(n) which returns the n^{th} fibonacci number in the fibonacci sequence.
The function definition should be as follows:

```
def fib(n):
    #theano code
    #theano code
    #theano code
```

solution1

In [7]:
```
# this function prints the first n fibonacci numbers using theano function and shared variables.
def fib1(n):
    # define the first number in fibonaci sequence
    if n==1:
        t=1
    # defines a shared variable to store the internal state
    a = shared(1)
    # creates a scalar increament varaible
    inc = T.lscalar('inc')
    # construcs a function to replace the shared variable value with the sume of the current state
    # and the increament amount
    f = function([inc],a,updates=[(a,a+inc)])
    t = a.get_value()
    for i in range(n-1):
        # iterates n  times to print n fibonacci numbers
        # updates inc variable in each iteration
        t = f(t)
    return t
```

solution2

In [12]:
```
# this function prints the first n fibonacci numbers using theano function and shared variables.
def fib2(n):
    #define the first two numbes in fibonaci sequence
    if n==1:
        t=1
    if n==2:
        t=1
    a = shared(1)
    b = shared(1)
    c = T.lscalar('c')
    c = a + b
    f = function([],c,updates=[(a,b), (b,a+b)])
    for i in range(n-2):
        t=f()
    return t
```