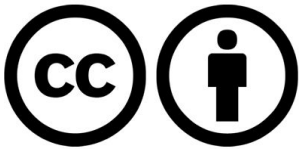
With PyTorch 0.2.0_2

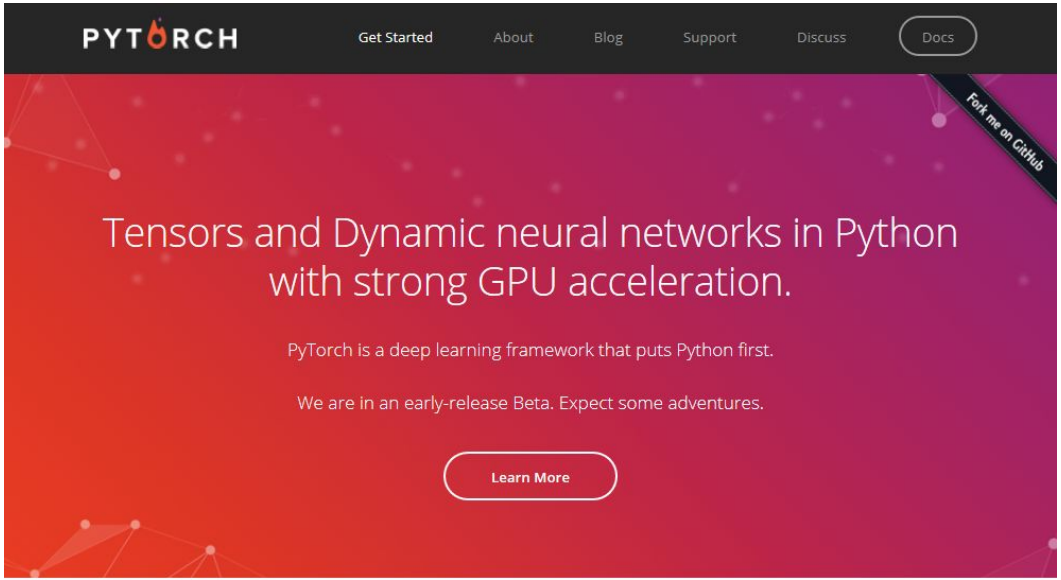
Lab 1

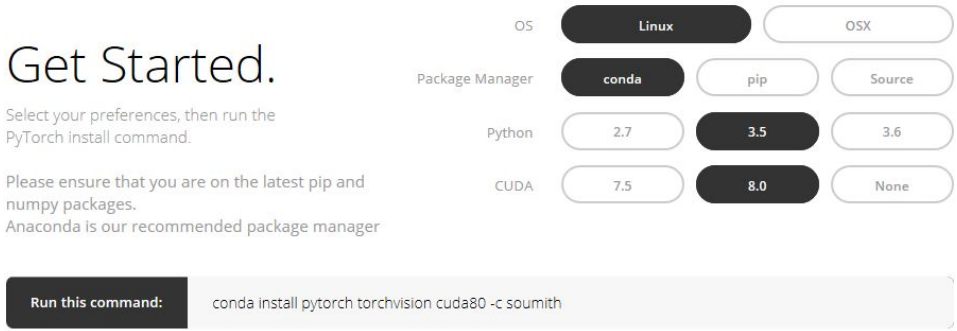
Pytorch Basics

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/







http://pytorch.org/

PYTORCH

1. Deep learning framework that puts Python first
2. Tensor Computation (like numpy)
   with strong GPU acceleration

http://pytorch.org/

PyTorch

A graph is created on the fly

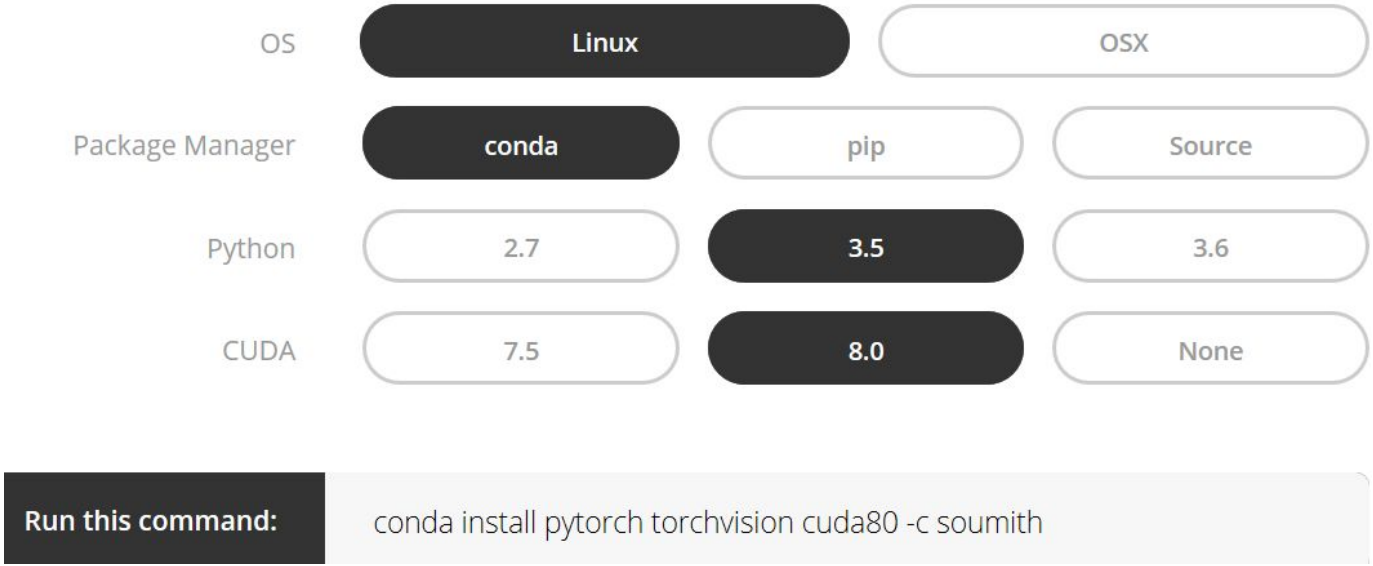

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

http://pytorch.org/

Installing PyTorch

- On the latest pip and numpy packages
- Anaconda is recommended package manager



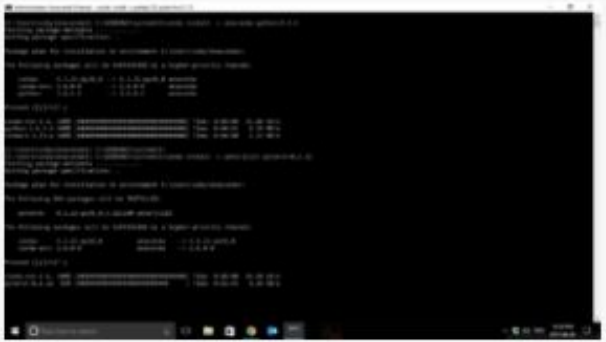
http://pytorch.org/

Installing PyTorch on windows

Operations in the Anaconda Prompt

a) Once the Anaconda Prompt is open, type in these commands in the order specified Enter y to proceed when prompted.

1. conda install -c anaconda python=3.6.1
2. conda install -c peterjc123 pytorch=0.1.12



https://www.superdatascience.com/pytorch/

https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/PythonZeroToAll/blob/master/lab-01-1-pytorch_basics.ipynb

Hello PyTorch!

What is PyTorch?

It's a Python based scientific computing package targeted at two sets of audiences:

- A replacement for numpy to use the power of GPUs
- A deep learning research platform that provides maximum flexibility and speed

https://github.com/hunkim/PythonZeroToAll/blob/master/lab-01-1-pytorch_basics.ipynb

Tensors

Tensors are similar to numpy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing.

In [3]:
```
from __future__ import print_function
import torch
```

Construct a 5x3 matrix, uninitialized:

In [5]:
```
x = torch.Tensor(5,3)
print(x)
```

```
1.00000e-36 *
  0.0000  0.0000  0.0000
  0.0000  0.0000  0.0000
  0.4113  0.0000  0.0000
  0.0000  0.0001  0.0000
  1.8967  0.0000  0.0000
[torch.FloatTensor of size 5x3]
```

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Tensors

Construct a randomly initialized matrix

In [9]:
```
x = torch.rand(5,3)
print(x)
```

```
 0.4381  0.1222  0.1948
 0.6345  0.0023  0.4593
 0.2548  0.3231  0.5043
 0.2990  0.9189  0.7335
 0.0187  0.8618  0.4062
[torch.FloatTensor of size 5x3]
```

Get its size

In [11]:
```
print(x.size())
```

```
torch.Size([5, 3])
```

- Note: torch Size is in fact a tuple, so it supports the same operations.

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Operations

There are multiple syntaxes for operations. Let's see addition as an example.

Addition: syntax 1

```
In [13]: y = torch.rand(5,3)
         print(x + y)
```

```
 0.8535  0.6359  1.0222
 0.6420  0.0091  0.6165
 0.9360  0.5439  0.9757
 1.1209  1.2988  1.6813
 0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

Addition: syntax2

```
In [14]: print(torch.add(x,y))
```

```
 0.8535  0.6359  1.0222
 0.6420  0.0091  0.6165
 0.9360  0.5439  0.9757
 1.1209  1.2988  1.6813
 0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Operations

Addition: giving an output tensor

In [16]:
```
result = torch.Tensor(5, 3)
torch.add(x, y, out=result)
print(result)
```

```
 0.8535  0.6359  1.0222
 0.6420  0.0091  0.6165
 0.9360  0.5439  0.9757
 1.1209  1.2988  1.6813
 0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

Addition: in-place

In [20]:
```
# adds x to y
y.add_(x)
print(y)
```

```
 1.2916  0.7581  1.2170
 1.2765  0.0114  1.0757
 1.1908  0.8670  1.4800
 1.4199  2.2177  2.4148
 0.4318  2.2995  1.0290
[torch.FloatTensor of size 5x3]
```

- Note: Any operation that mutates a tensor in-place is post-fixed with an _.
  For example, x.copy(y), x.t_(), will change x.

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Operations

You can use standard numpy-like indexing with all bells and whistles!

```
In [22]: print(x[:, 1])

 0.1222
 0.0023
 0.3231
 0.9189
 0.8618
[torch.FloatTensor of size 5]
```

- Read later: 100+ Tensor operations, including transposing, indexing, slicing, mathematical operations, linear algebra, random numbers, etc are described here http://pytorch.org/docs/torch

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Numpy Bridge

- Converting a torch Tensor to a numpy array and vice versa is a breeze.
- The torch Tensor and numpy array will share their underlying memory locations, and changing one will change the other.

Convertng torch Tensor to numpy Array

In [23]:
```
a = torch.ones(5)
print(a)
```

```
 1
 1
 1
 1
 1
[torch.FloatTensor of size 5]
```

In [25]:
```
b = a.numpy()
print(b)
```

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

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Numpy Bridge

See how the numpy array changed in value.

In [26]:
```
a.add_(1)
print(a)
print(b)
```

```
 2
 2
 2
 2
 2
[torch.FloatTensor of size 5]

[ 2.  2.  2.  2.  2.]
```

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

Numpy Bridge

Covering numpy Array to torch Tensor

See how changing the np array changedthe torch Tensor atomatically

In [27]:
```
import numpy as np
a = np.ones(5)
b = torch.from_numpy(a)
np.add(a, 1, out=a)
print(a)
print(b)
```

```
[ 2.  2.  2.  2.  2.]

 2
 2
 2
 2
 2
[torch.DoubleTensor of size 5]
```

All the Tensors on the CPU except a CharTensor support converting to Numpy and back.

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

CUDA Tensors

Tensors can be moved onto GPU using the .cude function..

In [32]:
```
# let us run this cell only if CUDA is available
if torch.cuda.is_available():
    x = x.cuda()
    y = y.cuda()
    x + y
    print(x + y)
```

```
 1.7296  0.8803  1.4118
 1.9110  0.0137  1.5350
 1.4456  1.1901  1.9842
 1.7189  3.1366  3.1483
 0.4506  3.1613  1.4352
[torch.cuda.FloatTensor of size 5x3 (GPU 0)]
```

- References: http://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#tensors

https://github.com/hunkim/DeepLearningZeroToAll/pytorch

With PyTorch 0.2.0_2

Lab 2

Linear Regression

Sung Kim <hunkim+ml@gmail.com>