Getting started with Theano

Jon Barker, Solutions Architect
Agenda

- What is Theano?
- Simple first application
- Theano for deep learning
- Step-by-step example
- Related projects
- Example applications
What is Theano?
A mathematical symbolic expression compiler

A Python library for symbolic maths - far broader than just Deep Learning
Tightly integrated with the Python ecosystem
Fast C/CUDA back-end and transparent GPU acceleration
What is Theano?

Symbolic expression compiler

Variables and expressions are symbolic - more like maths than code...

```python
1 import theano
2 from theano import tensor as T
3 x = T.vector('x')
4 W = T.matrix('W')
5 b = T.vector('b')

...but, symbolic expressions use a familiar NumPy-like syntax

8 dot = T.dot(x, W)
9 out = T.nnet.sigmoid(dot + b)
```
What is Theano?

Developed and used since January 2008, created at Université de Montréal

Large contributor community

Tools for inspecting and debugging code

Great tutorials and examples - http://deeplearning.net/tutorial/
What is Theano?

Theano defines a language, a compiler and a library

Recipe for a Theano application:

Define symbolic expressions

Compile a function that can compute numeric values using those expressions

Execute that function on data
Example Theano application

\[ y = a \times b \]
\[ a, b \in \mathbb{R} \]

```python
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()
y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print multiply(3, 2) #6
print multiply(4, 5) #20
```
Example Theano application

\[ y = a \times b \]
\[ a, b \in \mathbb{R} \]

```python
import theano
from theano import tensor as T

a = T.scalar()  # Initialize symbolic variables
b = T.scalar()

y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print multiply(3, 2) # 6
print multiply(4, 5) # 20
```
Example Theano application

\[ y = a \times b \]

\[ a, b \in \mathbb{R} \]

```python
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()

y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print(multiply(3, 2)) #6
print(multiply(4, 5)) #20
```
Example Theano application

$$y = a \times b$$
$$a, b \in \mathbb{R}$$

```python
import theano
from theano import tensor as T

a = T.scalar()  # Initialize symbolic variables
b = T.scalar()

y = a * b  # Define symbolic expression

multiply = theano.function(inputs=[a, b], outputs=y)  # Compile a function

print multiply(3, 2)  # 6
print multiply(4, 5)  # 20
```
Example Theano application

\[ y = a \times b \]
\[ a, b \in \mathbb{R} \]

```python
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()

y = a * b  # Define symbolic expression

multiply = theano.function(inputs=[a, b], outputs=y)  # Compile a function

print multiply(3, 2)  # Use on numeric data
print multiply(4, 5)  
```
Theano for deep learning

Symbolic computation for tensors
Sub-modules of tensor operations relevant to DL
Highly expressive
Symbolic differentiation
Transparent GPU acceleration
Easily integrates with Python ecosystem
Theano for deep learning
Symbolic computation for tensors

\[ y_1 = \sigma(w_{11}x_1 + w_{21}x_2 + w_{31}x_3) \]
\[ y_2 = \sigma(w_{12}x_1 + w_{22}x_2 + w_{32}x_3) \]
\[ \bar{y} = \sigma(W\bar{x}) \]

```
1 W = T.matrix('W')
2 x = T.matrix('x')
3 dot = T.dot(x, W)
4 y = T.nnet.sigmoid(dot)
```
Theano for deep learning
Highly expressive: easily add new activation or loss functions

```python
# leaky rectified linear activation function
def leaky(x):
    return theano.tensor.switch(x<0, 0.01 * x, x)

# cross-entropy cost
def ce_loss(x, z):
    return -T.sum(x * T.log(x) + (1 - x) * T.log(1 - z), axis=1)
```
Theano for deep learning
Symbolic differentiation

1. \text{gradient} = \text{T}.grad(\text{cost}=L, \text{wrt}=W)
2. \text{updates} = \text{[[w, w - gradient \times 0.01]]}
Example: Linear model

Fit a linear model to this data, i.e. find $w \in \mathbb{R}$ such that $y = wx$

Example: Linear model

```
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradien = T.grad(cost=cost, wrt=W)
updates = [[W, W - gradient * 0.01]]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example: Linear model

```python
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradient = T.grad(cost=cost, wrt=W)
updates = [[W, W - gradient * 0.01]]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradients = T.grad(cost=cost, wrt=W)
updates = [[W, W - gradient * 0.01]]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```python
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradient = T.grad(cost=cost, wrt=W)
updates = [[W, W - gradient * 0.01]]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradient = T.grad(cost=cost, wrt=W)
updates = [(W, W - gradient * 0.01)]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```python
X = T.scalar()
Y = T.scalar()
def model(X, W):
    return X * W
W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)
cost = T.mean(T.sqr(y - Y))
gradiant = T.grad(cost=cost, wrt=W)
updates = [([W, W - gradient * 0.01])]
train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)
for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.array(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradiant = T.grad(cost=cost, wrt=W)
updates = [[W, W - gradient * 0.01]]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example credit: “Introduction to Deep Learning with Python”, Alec Radford, [https://www.youtube.com/watch?v=S75EdAcXHKk](https://www.youtube.com/watch?v=S75EdAcXHKk)
Example: Linear model

```
X = T.scalar()
Y = T.scalar()

def model(X, W):
    return X * W

W = theano.shared(np.asarray(0., dtype=theano.config.floatX))
y = model(X, W)

cost = T.mean(T.sqr(y - Y))
gradients = T.grad(cost=cost, wrt=W)
updates = [(W, W - gradient * 0.01)]

train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)

for i in range(100):
    for x, y in zip(train_x, train_y):
        train(x, y)
```

Example: Linear model

Before training

After 100 iterations
Designing more complex architectures

Layer types are typically defined as Python classes

```python
class ConvLayer(object):
    def __init__(self, rng, input, filter_shape, image_shape, poolsize=(2, 2)):
        self.input = input
        self.W = theano.shared(np.asarray(
            rng.uniform(size=filter_shape),
            dtype=theano.config.floatX
        ), borrow=True)
        conv_out = conv.conv2d(
            input=input,
            filters=self.W,
            filter_shape=filter_shape,
            image_shape=image_shape
        )
        self.output = T.tanh(conv_out)
        self.params = [self.W]
```
Designing more complex architectures

Networks can be defined symbolically as sequences of class instances

```python
layer0 = ConvLayer(
    rng,
    input=layer0_input,
    image_shape=(batch_size, 1, 28, 28),
    filter_shape=(n_filter0, 1, 5, 5)
)

layer1 = ConvLayer(
    rng,
    input=layer0.output,
    image_shape=(batch_size, n_filter0, 12, 12),
    filter_shape=(n_filter1, n_filter0, 5, 5)
)
```
Related projects
Built on top of Theano (mostly machine learning)

Blocks
Keras
Lasagne
Morb
Pylearn2
PyMC 3
sklearn-theano
theano-rnn
more...

Typically simplify syntax and interface for artificial neural network training at the expense of expressiveness
Related projects

CNN example in Keras

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD

model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='full'))
model.add(Activation('relu'))
model.add(MaxPooling2D(poolsize=(2, 2)))
model.add(Flatten())
model.add(Dense(64*8*8, 256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(256, 10))
model.add(Activation('softmax'))
sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
model.fit(X_train, Y_train, batch_size=32, nb_epoch=1)
```
Example applications
Music recommendation at Spotify

http://benanne.github.io/2014/08/05/spotify-cnns.html
Example applications

Classifying galaxies and plankton and winning Kaggle contests

http://benanne.github.io/2014/04/05/galaxy-zoo.html

http://benanne.github.io/2015/03/17/plankton.html
Example applications

Many more examples at: http://deeplearning.net/tutorial/

Unsupervised training and auto-encoders

Recurrent Neural Networks
Installation

Linux, OS X or Windows

Requirements:

- Python >= 2.6
- g++, python-dev
- NumPy, SciPy
- BLAS
Installation

Basic instructions:

    pip install Theano
    easy_install Theano

Advanced instructions (for bleeding edge installs):

    git clone git://github.com/Theano/Theano.git
    cd Theano
    python setup.py develop --user
Configuration

Three ways to configure Theano:

`~/.theanorc`: settings you always want

`THEANO_FLAGS`: setting for one job

`theano.config`: mid-code settings changes
GPU acceleration

Three ways to invoke:

Add `device=gpu` to `.theanorc`
Add `device=gpu` to `THEANO_FLAGS`
Set `theano.config.device='gpu'` in code

cuDNN acceleration (including v4) is automatic if installed on system
Deep Learning Lab Series Schedule

developer.nvidia.com/deep-learning-courses

- Review the other seminars in series

Seminar #2 - Introduction to DIGI Ts
Seminar #3 - Getting Started with the Caffe Framework
Seminar #5 - Getting Started with the Torch Framework
Hands-on Lab

1. Create an account at nvidia.qwiklab.com
2. Go to “Getting started with Theano” lab at bit.ly/dlnvlab4
3. Start the lab and enjoy!

Only requires a supported browser, no NVIDIA GPU necessary!
Lab is free until end of Deep Learning Lab series