(https://www.facebook.com/AnalyticsVidhya) | (https://twitter.com/analyticsvidhya) | (https://plu

(https://www.linkedin.com/groups/Analytics-Vidhya-Learn-everything-about-5057165)

Home (https://www.analyticsvidhya.com/) | Blog (https://www.analyticsvidhya.com/blog/) | Jobs (https:

Trainings (https://www.analyticsvidhya.com/trainings/)

Learning Paths (https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-

Datahack Summit 2017 (https://www.analyticsvidhya.com/datahacksummit/)

(https://www.analyticsvidhya.com)

Practical Guide to implementing Neural Networks in Python (using Theano)

MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNING/) PYTHON (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-2/)

//www.facebook.com/sharer.php?u=https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-
:al%20Guide%20to%20implementing%20Neural%20Networks%20in%20Python%20(using%20Theano)) (https://twitter.com/home?
20Guide%20to%20implementing%20Neural%20Networks%20in%20Python%20(using%20Theano)+https://www.analyticsvidhya.com/blog/
theano/) (https://plus.google.com/share?url=https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/)
:om/pin/create/button/?url=https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/&media=https://www.analytics
2016/04/1-1.jpg&description=Practical%20Guide%20to%20implementing%20Neural%20Networks%20in%20Python%20(using%20Theano)

(http://events.upxacademy.com/infosession-bd?utm_source=Infosession-AVBA&utm_medium=Banner&utm_campaign=infosession)

Introduction

In my last article, I discussed the fundamentals of deep (https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamenta networks/), where I explained the basic working of a artificial neural network. If you've bee this series, today we'll become familiar with practical process of implementing neural Python (using Theano package).

I found various other packages also such as Caffe, Torch, TensorFlow etc to do this job. But no less than and satisfactorily execute all the tasks. Also, it has multiple benefits wh enhances the coding experience in Python.

In this article, I'll provide a comprehensive practical guide to implement Neural Netw Theano. If you are here for just python codes, feel free to skip the sections and learn at And, if you are new to Theano, I suggest you to follow the article sequentially to gair knowledge.

Note:

1. *This article is best suited for users with knowledge of neural network & deep learning.*
2. *If you don't know python, start here (https://www.analyticsvidhya.com/blog/2016/01 tutorial-learn-data-science-python-scratch-2/).*
3. *If you don't know deep learning, start (https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamenta networks/).*

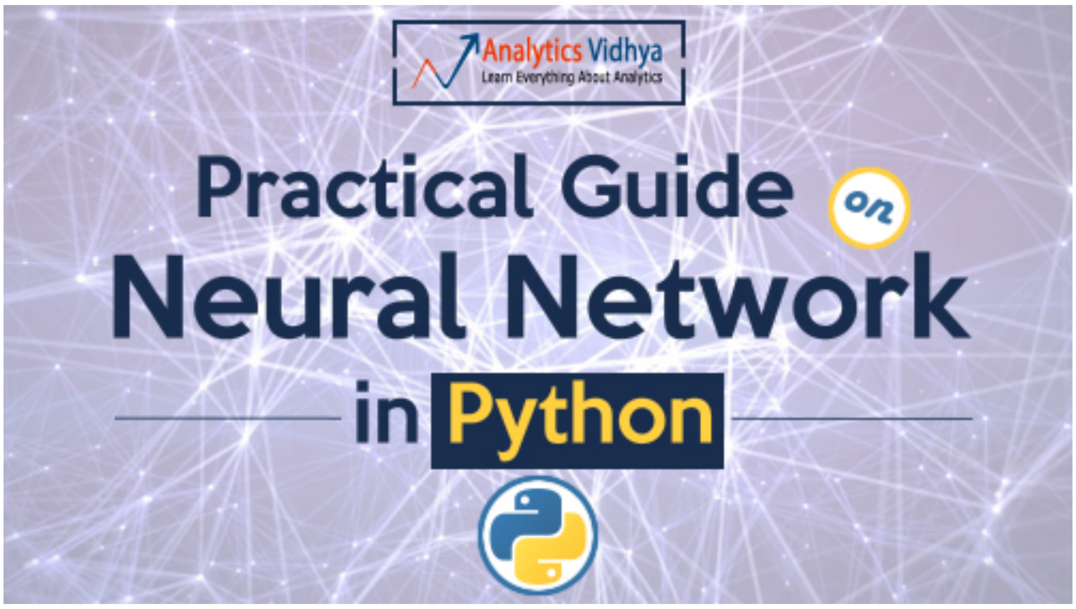


Table of Contents

1. Theano Overview

In short, we can define Theano as:

- A programming language which runs on top of Python but has its own data structure whic
  integrated with numpy
- A linear algebra compiler with defined C-codes at the backend
- A python package allowing faster implementation of mathematical expressions

As popularly known, Theano was developed at the University of Montreal in 2008. It
defining and evaluating mathematical expressions in general.

Theano has several features which optimize the processing time of expressions. For instance it modifies the symbolic expressions we define before converting them to C codes
Examples:

- It makes the expressions faster, for instance it will change { (x+y) + (x+y) } to { 2*(x+y) }
- It makes expressions more stable, for instance it will change { exp(a) / exp(a).sum(axis=1) } tc softmax(a) }

Below are some powerful advantages of using Theano:

1. It defines C-codes for different mathematical expressions.
2. The implementations are much faster as compared to some of the python's default implem
3. Due to fast implementations, it works well in case of high dimensionality problems.
4. It allows GPU implementation which works blazingly fast specially for problems like deep le

Let's now focus on Theano (with example) and try to understand it as a programming langu

2. Implementing Simple Expressions

Lets start by implementing a simple mathematical expression, say a multiplication in Thea how the system works. In later sections, we will take a deep dive into individual compc general structure of a Theano code works in 3 steps:

1. Define variables/objects
2. Define a mathematical expression in the form of a function
3. Evaluate expressions by passing values

Lets look at the following code for simply multiplying 2 numbers:

Step 0: Import libraries

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

Here, we have simply imported 2 key functions of theano – tensor and function.

Step 1: Define variables

```
a = T.dscalar('a')
b = T.dscalar('b')
```

Here 2 variables are defined. Note that we have used Theano tensor object type here. Also, arguments passed to dscalar function are just name of tensors which are useful while debu They code will work even without them.

Step 2: Define expression

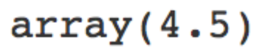
```
c = a*b
f = function([a,b],c)
```

Here we have defined a function f which has 2 arguments:

1. Inputs [a,b]: these are inputs to system
2. Output c: this has been previously defined

Step 3: Evaluate Expression

```
f(1.5,3)
```



```
array(4.5)
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/1.-ou 1.png)

Now we are simply calling the function with the 2 inputs and we get the output as a mul two. In short, we saw how we can define mathematical expressions in Theano and eval Before we go into complex functions, lets understand some inherent properties of Theanc be useful in building neural networks.

3. Theano Variable Types

Variables are key building blocks of any programming language. In Theano, the objects are tensors. A tensor can be understood as a generalized form of a vector with dimension dimensions are analogous to different types:

- t = 0: scalar
- t = 1: vector
- t = 2: matrix
- and so on..

Watch this (https://www.youtube.com/watch?v=f5liqUk0ZTw) interesting video to get a dee of intuition into vectors and tensors.

These variables can be defined similar to our definition of 'dscalar' in the above code. T keywords for defining variables are:

- **byte:** bscalar, bvector, bmatrix, brow, bcol, btensor3, btensor4
- **16-bit integers:** wscalar, wvector, wmatrix, wrow, wcol, wtensor3, wtensor4
- **32-bit integers:** iscalar, ivector, imatrix, irow, icol, itensor3, itensor4
- **64-bit integers:** lscalar, lvector, lmatrix, lrow, lcol, ltensor3, ltensor4
- **float:** fscalar, fvector, fmatrix, frow, fcol, ftensor3, ftensor4
- **double:** dscalar, dvector, dmatrix, drow, dcol, dtensor3, dtensor4
- **complex:** cscalar, cvector, cmatrix, crow, ccol, ctensor3, ctensor4

Now you understand that we can define variables with different memory allocations and c But this is not an exhaustive list. We can define dimensions higher than 4 using a generic T class. You'll find more details (http://deeplearning.net/software/theano/library/tensor/basic.html#libdoc-tensor-creatio

Please note that variables of these types are just symbols. They don't have a fixed valu passed into functions as symbols. They only take values when a function is called. But, we variables which are constants and which we need not pass in all the functions. For th provides shared variables. These have a fixed value and are not of the types discussed a can be defined as numpy data types or simple constants.

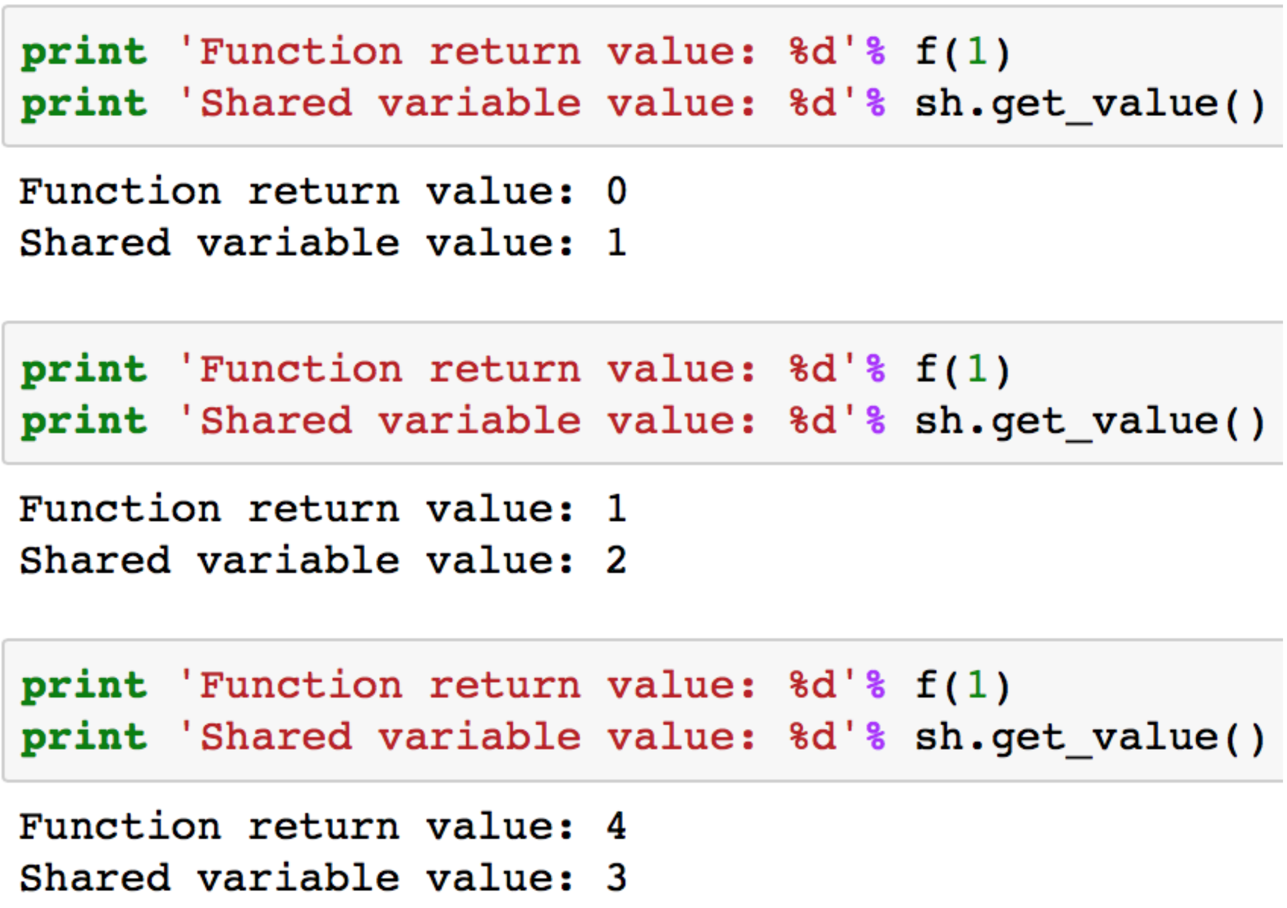
Lets take an example. Suppose, we initialize a shared variable as 0 and use a function whicl

- takes an input
- adds the input to the shared variable
- returns the square of shared variable

This can be done as:

```
from theano import shared
x = T.iscalar('x')
sh = shared(0)
f = function([x], sh**2, updates=[(sh,sh+x)])
```

Note that here function has an additional argument called updates. It has to be a list of list each containing 2 elements of form (shared_variable, updated_value). The output for 3 s runs is:



```
print 'Function return value: %d'% f(1)
print 'Shared variable value: %d'% sh.get_value()
```

```
Function return value: 0
Shared variable value: 1
```

```
print 'Function return value: %d'% f(1)
print 'Shared variable value: %d'% sh.get_value()
```

```
Function return value: 1
Shared variable value: 2
```

```
print 'Function return value: %d'% f(1)
print 'Shared variable value: %d'% sh.get_value()
```

```
Function return value: 4
Shared variable value: 3
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/2.-shared.png)

You can see that for each run, it returns the square of the present value, i.e. the value befor After the run, the value of shared variable gets updated. Also, note that shared variab functions "get_value()" and "set_value()" which are used to read and modify the value variables.

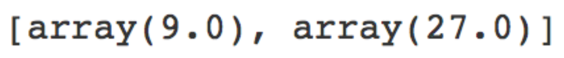
4. Theano Functions

Till now we saw the basic structure of a function and how it handles shared variables. forward and discuss couple more things we can do with functions:

Return Multiple Values

We can return multiple values from a function. This can be easily done as shown in followir example:

```
a = T.dscalar('a')
f = function([a],[a**2, a**3])
f(3)
```



```
[array(9.0), array(27.0)]
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/3.-multiple-output.png)

We can see that the output is an array with the square and cube of the number passed into function.

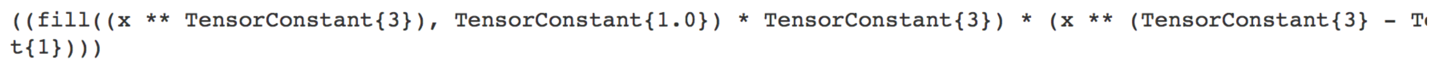
Computing Gradients

Gradient computation is one of the most important part of training a deep learning model. done easily in Theano. Let's define a function as the cube of a variable and determine its gr

```
x = T.dscalar('x')
y = x**3
qy = T.grad(y,x)
f = function([x],qy)
f(4)
```

This returns 48 which is $3x^2$ for x=4. Lets see how Theano has implemented this derivativ pretty-print feature as following:

```
from theano import pp  #pretty-print
print(pp(qy))
```



```
((fill((x ** TensorConstant{3}), TensorConstant{1.0}) * TensorConstant{3}) * (x ** (TensorConstant{3} - Te
t{1})))
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/4.-pp.png)

In short, it can be explained as: **fill(x^3,1)*3*x^{3-1}** You can see that this is exactly the **derivative**
that fill(x^3,1) simply means to make a matrix of same shape as x^3 and fill it with 1. This is use
high dimensionality input and can be ignored in this case.

We can use Theano to compute Jacobian and Hessian matrices as well which you can find
(http://deeplearning.net/software/theano/tutorial/gradients.html).

There are various other aspects of Theano like conditional and looping constructs. You c
further detail using following resources:

1. Theano Conditional Constructs (http://deeplearning.net/software/theano/tutorial/conditio
2. Theano Looping Statements (http://deeplearning.net/software/theano/tutorial/loop.html)
3. Handling Shape Information (http://deeplearning.net/software/theano/tutorial/shape_info

5. Modeling a Single Neuron

Lets start by modeling a single neuron.

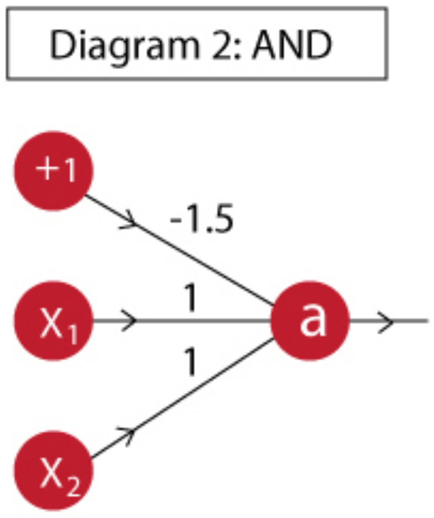
Note that I will take examples from my previous article on neuron networks here. If you w
the detail of how these work, please read this
(https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamenta
networks/). For modeling a neuron, lets adopt a 2 stage process:

1. Implement Feed Forward Pass
   - take inputs and determine output
   - use the fixed weights for this case
2. Implement Backward Propagation
   - calculate error and gradients
   - update weights using gradients

Lets implement an AND gate for this purpose.

Feed Forward Pass

An AND gate can be implemented as:

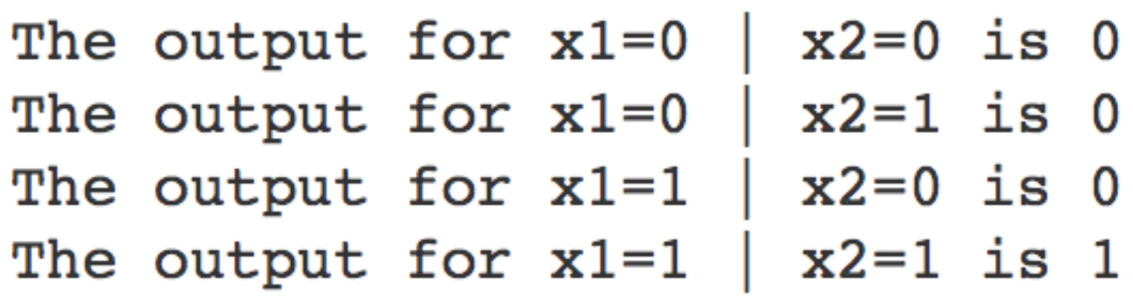


Now we will define a feed forward network which takes inputs and uses the shown determine the output. First we will define a neuron which computes the output a.

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

#Define variables:
x = T.vector('x')
w = T.vector('w')
b = T.scalar('b')

#Define mathematical expression:
z = T.dot(x,w)+b
a = ifelse(T.lt(z,0),0,1)

neuron = theano.function([x,w,b],a)
```

I have simply used the steps we saw above. If you are not sure how this expression works, p to the neural networks article I have referred above. Now let's test out all values in the trutl see if the AND function has been implemented as desired.

```
#Define inputs and weights
inputs = [
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
]
weights = [ 1, 1]
bias = -1.5

#Iterate through all inputs and find outputs:
for i in range(len(inputs)):
    t = inputs[i]
    out = neuron(t,weights,bias)
    print 'The output for x1=%d | x2=%d is %d' % (t[0],t[1],out)
```



```
The output for x1=0 | x2=0 is 0
The output for x1=0 | x2=1 is 0
The output for x1=1 | x2=0 is 0
The output for x1=1 | x2=1 is 1
```

(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/5.-single-neuron.png)

Note that, in this case we had to provide weights while calling the function. However, required to update them while training. So, its better that we define them as a shared va following code implements *w* as a shared variable. Try this out and you'll get the same outp

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

#Define variables:
x = T.vector('x')
w = theano.shared(np.array([1,1]))
b = theano.shared(-1.5)

#Define mathematical expression:
z = T.dot(x,w)+b
a = ifelse(T.lt(z,0),0,1)

neuron = theano.function([x],a)

#Define inputs and weights
inputs = [
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
]

#Iterate through all inputs and find outputs:
for i in range(len(inputs)):
    t = inputs[i]
    out = neuron(t)
    print 'The output for x1=%d | x2=%d is %d' % (t[0],t[1],out)
```

Now the feedforward step is complete.

Backward Propagation

Now we have to modify the above code and perform following additional steps:

1. Determine the cost or error based on true output
2. Determine gradient of node
3. Update the weights using this gradient

Lets initialize the network as follow:

```
#Gradient
import theano
import theano.tensor as T
from theano.ifelse import ifelse
import numpy as np
from random import random

#Define variables:
x = T.matrix('x')
w = theano.shared(np.array([random(),random()]))
b = theano.shared(1.)
learning_rate = 0.01

#Define mathematical expression:
z = T.dot(x,w)+b
a = 1/(1+T.exp(-z))
```

Note that, you will notice a change here as compared to above program. I have defined x as here and not a vector. This is more of a vectorized approach where we will determine all the together and find the total cost which is required for determining the gradients.

You should also keep in mind that I am using the full-batch gradient descent here, i.e. we training observations to update the weights.

Let's determine the cost as follows:

```
a_hat = T.vector('a_hat') #Actual output
cost = -(a_hat*T.log(a) + (1-a_hat)*T.log(1-a)).sum()
```

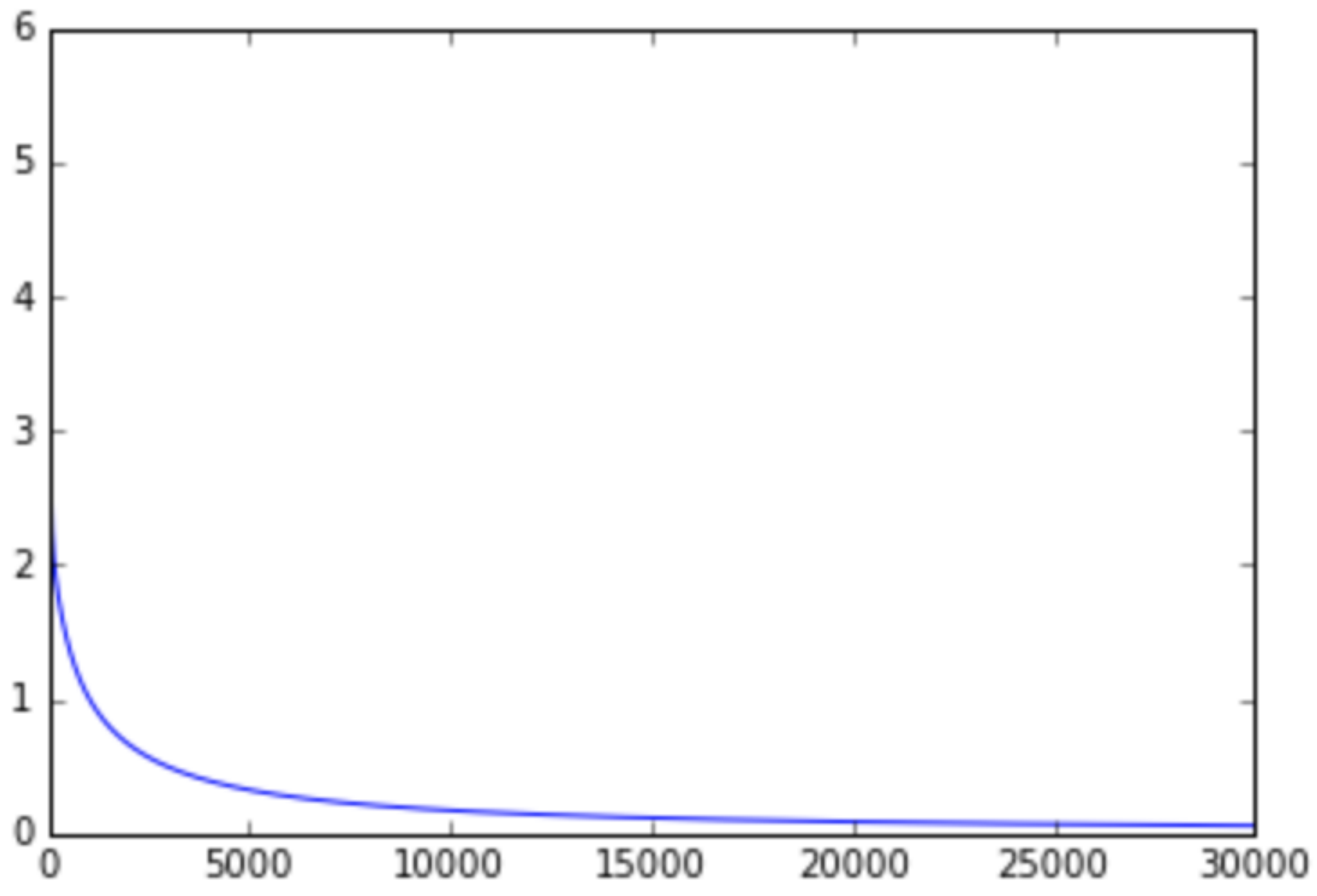
In this code, we have defined a_hat as the actual observations. Then we determine the c simple logistic cost function since this is a classification problem. Now lets compute the gra define a means to update the weights.

```
dw,db = T.grad(cost,[w,b])

train = function(
    inputs = [x,a_hat],
    outputs = [a,cost],
    updates = [
        [w, w-learning_rate*dw],
        [b, b-learning_rate*db]
    ]
)
```

In here, we are first computing gradient of the cost w.r.t. the weights for inputs and bias uni train function here does the weight update job. This is an elegant but tricky approach weights have been defined as shared variables and the updates argument of the functior update them every time a set of values are passed through the model.

```
#Define inputs and weights
inputs = [
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
]
outputs = [0,0,0,1]

#Iterate through all inputs and find outputs:
cost = []
for iteration in range(30000):
    pred, cost_iter = train(inputs, outputs)
    cost.append(cost_iter)

#Print the outputs:
print 'The outputs of the NN are:'
for i in range(len(inputs)):
    print 'The output for x1=%d | x2=%d is %.2f' % (inputs[i][0],inputs[i][1],pred[i

#Plot the flow of cost:
print '\nThe flow of cost during model run is as following:'
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(cost)
```

```
The outputs of the NN are:
The output for x1=0 | x2=0 is 0.00
The output for x1=0 | x2=1 is 0.02
The output for x1=1 | x2=0 is 0.02
The output for x1=1 | x2=1 is 0.98

The flow of cost during model run is as follow

[<matplotlib.lines.Line2D at 0x111159f50>]
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/6.-output-single-updat

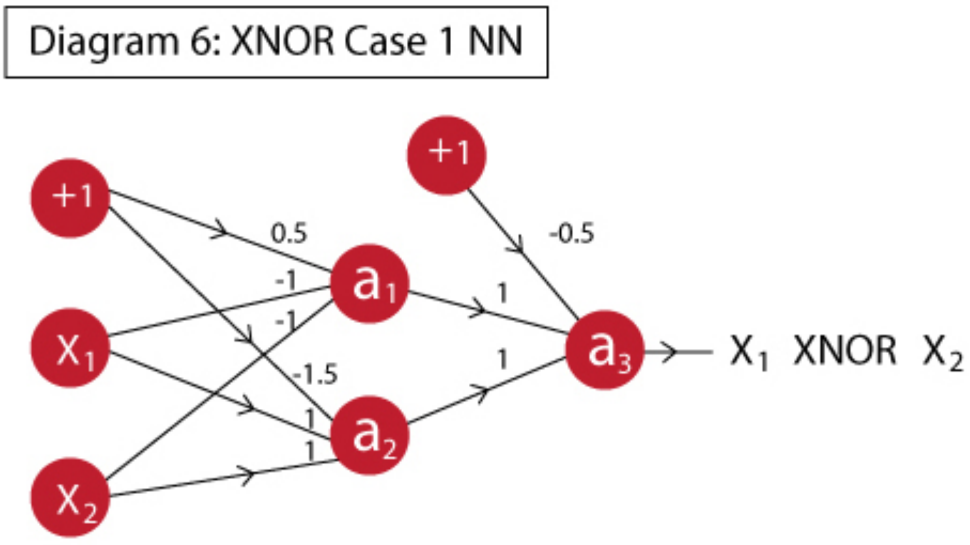
Here we have simply defined the inputs, outputs and trained the model. While training, w recorded the cost and its plot shows that our cost reduced towards zero and then finally s a low value. The output of the network also matched the desired output closely. Henc successfully implemented and trained a single neuron.

6. Modeling a Two-Layer Neural Network

I hope you have understood the last section. If not, please do read it multiple times and this section. Along with learning Theano, this will enhance your understanding of neural n the whole.

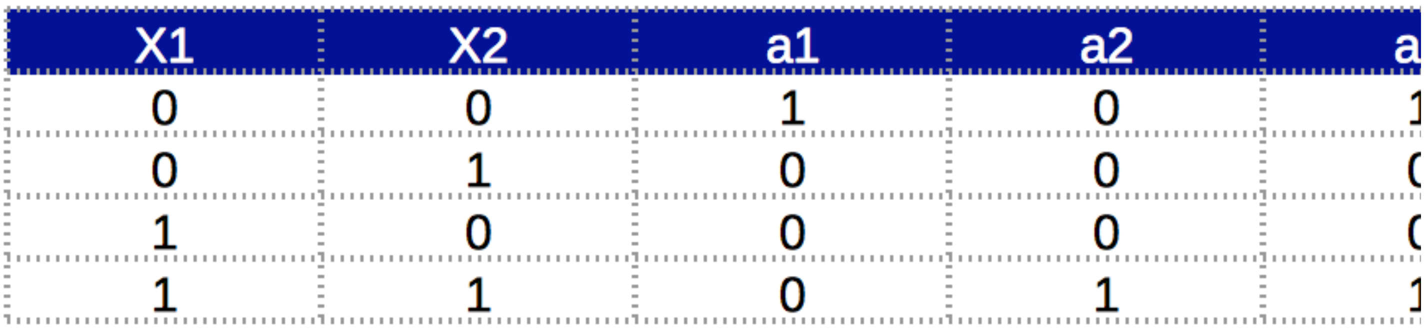
Lets consolidate our understanding by taking a 2-layer example. To keep things simple, XNOR example like in my previous article. If you wish to explore the nitty-gritty of how recommend reading the previous (https://www.analyticsvidhya.com/blog/2016/03/introduction-deep-learning-fundamenta networks/).

The XNOR function can be implemented as:



(https://www.analyticsvidhya.com/wp-content/uploads/2016/03/6.jpg)

As a reminder, the truth table of XNOR function is:



X1	X2	a1	a2	a
0	0	1	0	1
0	1	0	0	(
1	0	0	0	(
1	1	0	1	1

(https://www.analyticsvidhya.com/wp-content/uploads/2016/03/8.-tt-xnor-case-1.png)

Now we will directly implement both feed forward and backward at one go.

Step 1: Define variables

```
import theano
import theano.tensor as T
from theano.ifelse import ifelse
import numpy as np
from random import random

#Define variables:
x = T.matrix('x')
w1 = theano.shared(np.array([random(),random()]))
w2 = theano.shared(np.array([random(),random()]))
w3 = theano.shared(np.array([random(),random()]))
b1 = theano.shared(1.)
b2 = theano.shared(1.)
learning_rate = 0.01
```

In this step we have defined all the required variables as in the previous case. Note that nc
3 weight vectors corresponding to each neuron and 2 bias units corresponding to 2 layers.

Step 2: Define mathematical expression

```
a1 = 1/(1+T.exp(-T.dot(x,w1)-b1))
a2 = 1/(1+T.exp(-T.dot(x,w2)-b1))
x2 = T.stack([a1,a2],axis=1)
a3 = 1/(1+T.exp(-T.dot(x2,w3)-b2))
```

Here we have simply defined mathematical expressions for each neuron in sequence. Not
an additional step was required where x2 is determined. This is required because we want t
of a1 and a2 to be combined into a matrix whose dot product can be taken with the weights

Lets explore this a bit further. Both a1 and a2 would return a vector with 4 units. So if we sim
array [a1, a2] then we'll obtain something like [[a11,a12,a13,a14], [a21,a22,a23,a24]]. Howeve
this to be [[a11,a21], [a12,a22], [a13,a23], [a14,a24]]. The stacking function of Theano does this

Step 3: Define gradient and update rule

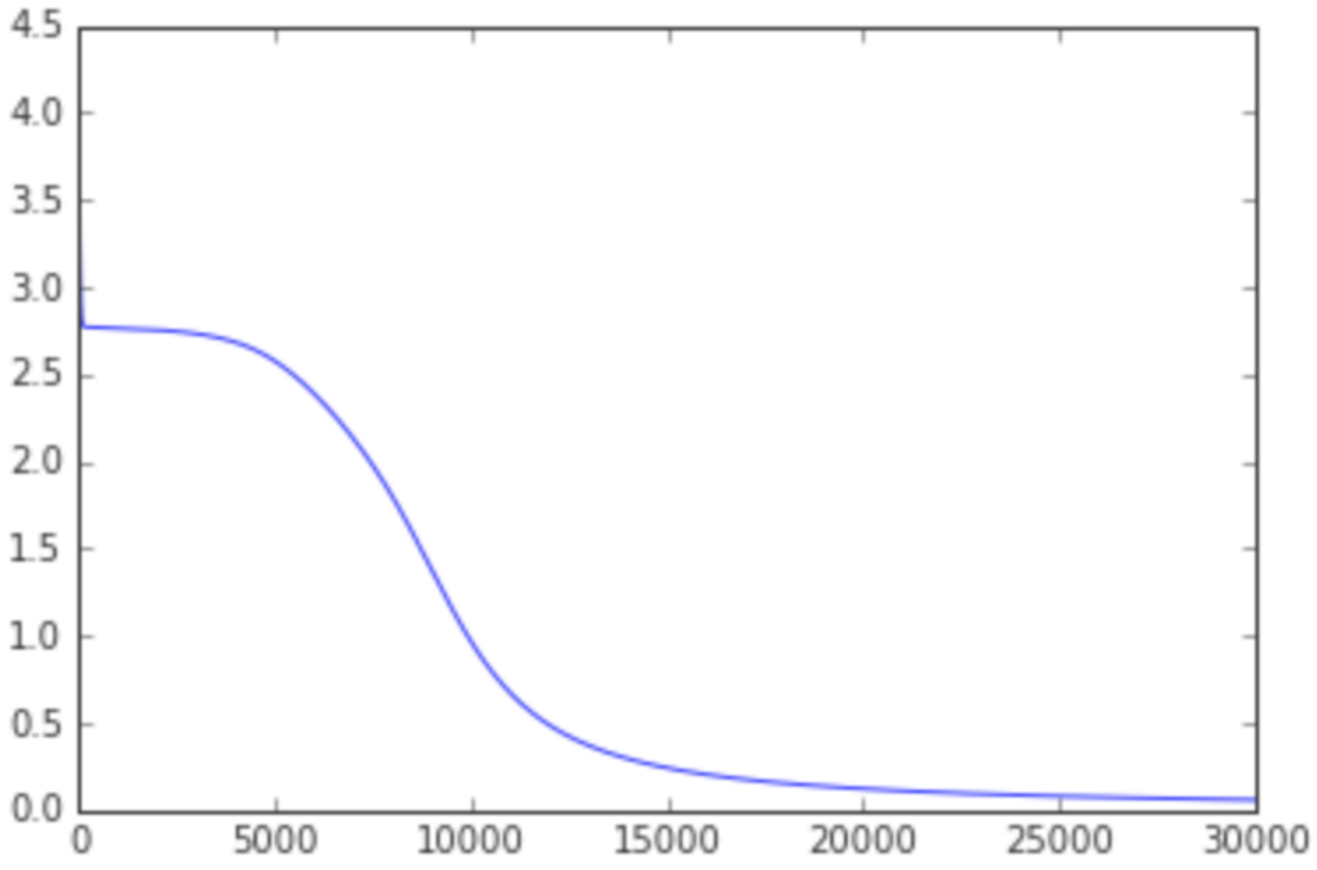
```
a_hat = T.vector('a_hat') #Actual output
cost = -(a_hat*T.log(a3) + (1-a_hat)*T.log(1-a3)).sum()
dw1,dw2,dw3,db1,db2 = T.grad(cost,[w1,w2,w3,b1,b2])

train = function(
    inputs = [x,a_hat],
    outputs = [a3,cost],
    updates = [
        [w1, w1-learning_rate*dw1],
        [w2, w2-learning_rate*dw2],
        [w3, w3-learning_rate*dw3],
        [b1, b1-learning_rate*db1],
        [b2, b2-learning_rate*db2]
    ]
)
```

This is very similar to the previous case. The key difference being that now we have to det
gradients of 3 weight vectors and 2 bias units and update them accordingly.

Step 4: Train the model

```
inputs = [
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
]
outputs = [1,0,0,1]

#Iterate through all inputs and find outputs:
cost = []
for iteration in range(30000):
    pred, cost_iter = train(inputs, outputs)
    cost.append(cost_iter)

#Print the outputs:
print 'The outputs of the NN are:'
for i in range(len(inputs)):
    print 'The output for x1=%d | x2=%d is %.2f' % (inputs[i][0],inputs[i][1],pred[i

#Plot the flow of cost:
print '\nThe flow of cost during model run is as following:'
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(cost)
```

```
The outputs of the NN are:
The output for x1=0 | x2=0 is 0.98
The output for x1=0 | x2=1 is 0.01
The output for x1=1 | x2=0 is 0.01
The output for x1=1 | x2=1 is 0.98

The flow of cost during model run is as followin

[<matplotlib.lines.Line2D at 0x116d081d0>]
```



(https://www.analyticsvidhya.com/wp-content/uploads/2016/04/7.-2-layer-op.png)

We can see that our network has successfully learned the XNOR function. Also, the cost of
has reduced to reasonable limit. With this, we have successfully implemented a 2-layer net

End Notes

In this article, we understood the basics of Theano package in Python and how it programming language. We also implemented some basic neural networks using Theanc that implementing Neural Networks on Theano will enhance your understanding of NN on

If hope you have been able to follow till this point, you really deserve a pat on your understand that Theano is not a traditional plug and play system like most of sklearn's But the beauty of neural networks lies in their flexibility and an approach like this will allow degree of customization in models. Some high-level wrappers of Theano do exist like Lasagne which you can check out. But I believe knowing the core of Theano will help y them.

Did you find this article useful ? Please feel free to share your feedback and questions belo waiting to interact with you!

Check out Live Competitions (http://datahack.analyticsvidhya.com/contest/a compete with best Data Scientists around the world.

Share this:

(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=linkedin&nb=1)
613

(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=facebook&nb=1)

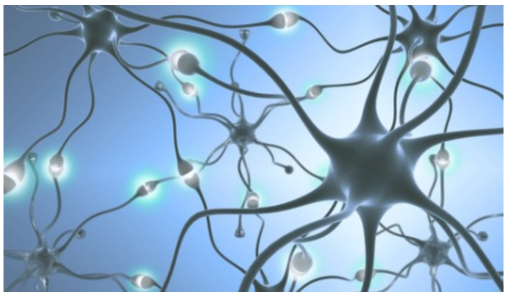
(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=google-plus-1&nb=1)

(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=twitter&nb=1)

(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=pocket&nb=1)

(https://www.analyticsvidhya.com/blog/2016/04/neural-networks-python-theano/?share=reddit&nb=1)

RELATED



(https://www.analyticsvidhya.com/blog/2016/11/fine-tuning-a-keras-model-using-theano-trained-neural-network-introduction-to-transfer-learning/)
Fine-tuning a Keras model using Theano trained Neural Network & Introduction to Transfer Learning (https://www.analyticsvidhya.com/blog/2016/11/fine-tuning-a-keras-model-using-theano-trained-neural-network-introduction-to-transfer-learning/)
November 21, 2016
In "Machine Learning"

Analytics Vidhya
Learn Everything About Analytics
Free Online Learning Resources
Deep Learning & Neural Network

(https://www.analyticsvidhya.com/blog/2015/11/free-resources-beginners-deep-learning-neural-network/)
Free Resources for Beginners on Deep Learning and Neural Network (https://www.analyticsvidhya.com/blog/2015/11/free-resources-beginners-deep-learning-neural-network/)
November 3, 2015
In "Deep Learning"

(https://www.analyticsvidh
g/2015/07/top-youtube-vi
machine-learning-neural-n
deep-learning/)
My playlist - Top YouTube V
Machine Learning, Neural N
Deep Learning
(https://www.analyticsvidh
g/2015/07/top-youtube-vi
machine-learning-neural-n
deep-learning/)
July 8, 2015
In "Deep Learning"

TAGS: BACKWARD PROPAGATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/BACKWARD-PROPAGATION/), DEEP LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/DEEP-LEARNING/), FORWARD PROPOGATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/FORV
PROPOGATION/), GRADIENT DESCENT (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/GRADIENT-DESCENT/), MACHINE LEARNING (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/MACHINE-LEARNING/), MATRIX FACTORIZATION (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/M
FACTORIZATION/), NEURAL NETWORK (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/NEURAL-NETWORK/), THEANO (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/THEANO/)

Previous Article
Case Study For Freshers (Level : Medium) – Call Center Optimization (https://www.analyticsvidhya.com/blog/2016/04/case-study-level-medium-call-center-optimization/)

Next Articl
It's our 3rd Birthday – Come & Celebrat
(https://www.analyticsvidhya.com/blog/2016/0
3-years-analytics-vidhyas-success/

(https://www.analyticsvidhya.com/blog/author/aarshay/)

Author

Aarshay Jain (https://www.analyticsvidhya.com/blog/author/aarshay/)

Aarshay is a ML enthusiast, pursuing MS in Data Science at Columbia Univer graduating in Dec 2017. He is currently exploring the various ML techniques writes articles for AV to share his knowledge with the community.

(mailto:aarshayjain@gmail.com) (https://in.linkedin.com/in/aarshayja

(https://github.com/aarshayj) (aarshay)

This is article is quiet old now and you might not get a prompt response from the author. We w request you to post this comment on Analytics Vidhya **Discussion portal** (https://discuss.analyticsvidhya.com/) to get your queries resolved.

14 COMMENTS

Gianni says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
APRIL 18, 2016 AT 9:57 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/ 109642)

Thanks Aarshay really usefull.

There's a big big step from sklearn fit models and build a theano functioning model tha confusion..

In beetwen there's keras, that permit to easily build neural network models, so maybe i first step from sklearn to theano.

Do you have plans for keras full tutorial ? I this case can I suggest to develop well the m
part, from the neural network graph to the keras code with a lot of examples ?

However, good job Aarshay.

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
APRIL 18, 2016 AT 10:03 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-P
THEANO/#COMMENT-109643)

Hi Gianni,

Yes you are right. Keras and Lasagne are somewhere in between.

I don't have immediate plans of doing Keras but if you are interested, you car
blog on Keras. Lets discuss this further offline. Please drop me a note on
aarshayjain@gmail.com (mailto:aarshayjain@gmail.com).

Regards,
Aarshay

allen kennedy says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
APRIL 20, 2016 AT 7:37 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/
109723)

Hi

Thank you very much for this great tutorial.

I just have one question. you are using the object a_hat as the actual outputs, but i cant
understand where the outputs are initialized in this vector? I'm probably missing somet

Can you please explain where they are been initialized?

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
APRIL 21, 2016 AT 2:58 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PY
THEANO/#COMMENT-109733)

Hi,

Theano works a bit differently. Theano variables are just objects which don't h
permanent memory. You can understand them as functions. So they get a va
when the function is called.

Thus Variables are not initialized here. Whenever we call the Theano functior
arguments which go into the variables being used in the function.

Hope this makes sense.

allen kennedy says:

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=

APRIL 22, 2016 AT 10:27 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL
PYTHON-THEANO/#COMMENT-109870)

Thanks you for the quick reply. But im still not understanding where
been called from. I understand that it is just a function and holds n
it is called,

I can see it created as a vector above – a_hat. Then the cost calcula
it, and at this stage it is still a blank vector. Then the function train c
as part of the input for inputs. But im struggling to see where it pas
values to it? Its probably because i do not know python too well, i r
R. But is the input parameter in the train function, taking in matrix x
returning a_hat?

Aarshay Jain says:

REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?
REPLYTOCOM=109885#RESPOND)

APRIL 23, 2016 AT 5:33 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016
NETWORKS-PYTHON-THEANO/#COMMENT-109885)

Lets take the last section as an example. Step 3 has train
train = function(
inputs = [x,a_hat],
outputs = [a3,cost],
updates = [
[w1, w1-learning_rate*dw1],
[w2, w2-learning_rate*dw2],
[w3, w3-learning_rate*dw3],
[b1, b1-learning_rate*db1],
[b2, b2-learning_rate*db2]

]
)
This takes 2 inputs – x and a_hat.

In the step 4 of last section, when we call train as:
pred, cost_iter = train(inputs, outputs)
It has 2 arguments – inputs, outputs. Inputs go into x and
into a_hat.

Hope this makes sense. Please feel free to discuss furthe
needed. I understand its not super intuitive like sklearn.

keerthisuresh (http://www.datawaretools.in) says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
MAY 5, 2016 AT 6:59 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/#C
110463)

keep sharing such ideas in the future as well.this was actually what i was looking for,an
to came here you keep up the fantastic work!my weblog..

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
MAY 5, 2016 AT 7:01 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTH
THEANO/#COMMENT-110464)

thank you keerthi.. i'll try my best 🙂

Babalola, Rotimi (http://plaindata.blogspot.com) says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=
MAY 8, 2016 AT 3:38 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/#C
110626)

Thanks for a wonderful post. My question is – For the neural network with one layer, yo
bias term in the mathematical expression i.e. z = T.dot(x,w) + b.

But for the multiple layer case you subtracted the bias term i.e.
a1 = 1/(1+T.exp(-T.dot(x,w1)-b1))
a2 = 1/(1+T.exp(-T.dot(x,w2)-b1))
a3 = 1/(1+T.exp(-T.dot(x2,w3)-b2))

I don't understand why it is different in both cases. Please can you explain? Thanks

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=

MAY 9, 2016 AT 5:17 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTH
THEANO/#COMMENT-110663)

You are welcome.

Regarding the mathematical expression, if you observe carefully, both the T.c
and b1 are negative. This is because the sigmoid function is 1/(1+e(-x). There x
"T.dot(x,w1) + b1". Hope this makes sense.

Babalola, Rotimi (http://plaindata.blogspot.com) says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=

MAY 9, 2016 AT 3:47 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NET
PYTHON-THEANO/#COMMENT-110699)

Yes, it makes sense. The minus in the sigmoid function turns the m
plus. I hope that is right. Thank you once again. Your website has he
lot. Keep it up

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?
REPLYTOCOM=110702#RESPOND)

MAY 9, 2016 AT 4:25 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/0
NETWORKS-PYTHON-THEANO/#COMMENT-110702)

Yup you got it! Thanks!

Poornachandra Sandur says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=

JUNE 5, 2016 AT 6:05 PM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/#
111897)

Superb article…

Aarshay Jain says: REPLY (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTHON-THEANO/?REPLYTOCOM=

JUNE 6, 2016 AT 9:00 AM (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/2016/04/NEURAL-NETWORKS-PYTH
THEANO/#COMMENT-111917)

Glad you liked it 🙂

LEAVE A REPLY

Your email address will not be published.

Comment

Name (required)

Email (required)

Website

SUBMI