[pylearn2安装及测试](http://blog.csdn.net/lucktroy/article/details/9466369)

分类： [Theano](http://blog.csdn.net/lucktroy/article/category/1505471)2013-07-25 11:55 5667人阅读 [评论](http://blog.csdn.net/lucktroy/article/details/9466369#comments)(17) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lucktroy/article/details/9466369#report)

Pylearn2 是建立在Theano之上的一个机器学习库，可以参考：http://deeplearning.net/software/pylearn2/

早前我已经用《[Install Theano in Linux without root authority (sodu)](http://blog.csdn.net/lucktroy/article/details/9336477)》，所以现在也只需要仿照着安装即可。

1. Download:

1. $ git clone git://github.com/lisa-lab/pylearn2.git

2. 配置环境变量，我一般用.bash\_profile, 也可以放在.bashrc里

1. export PYLEARN2\_DATA\_PATH=/path/to/data

3. 安装，进入pylearn2

1. $ python setup.py build
2. $ python setup.py install --prefix=$PYTHON\_HOME

4. 测试，可以看http://deeplearning.net/software/pylearn2/tutorial/index.html#tutorial，首先下载数据集cifar-10: http://www.cs.toronto.edu/~kriz/cifar.html

1. $ wget http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
2. $ tar -zxvf cifar-10\*

4.1. 生成数据子集，进入$PYLEARN2\_HOME/pylearn2/scripts/tutorials/grbm\_smd/

1. $ python make\_dataset.py

4.2 训练模型

1. $ python ../../train.py cifar\_grbm\_smd.yaml

4.3 视察模型

1. $ python ../../show\_weights.py cifar\_grbm\_smd.pkl

# [使用Pylearn2构建一个简单的分类器](http://blog.csdn.net/tangchenyi/article/details/25076093)

分类： [Machine Learning](http://blog.csdn.net/tangchenyi/article/category/2243145)2014-05-05 22:22 536人阅读 [评论](http://blog.csdn.net/tangchenyi/article/details/25076093#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/tangchenyi/article/details/25076093#report)

[Pylearn2](http://www.csdn.net/tag/Pylearn2)[Theano](http://www.csdn.net/tag/Theano)[分类器](http://www.csdn.net/tag/%e5%88%86%e7%b1%bb%e5%99%a8)[Deep Learning](http://www.csdn.net/tag/Deep%20Learning)

久闻Pylearn大名，最近由于某些原因，想拿Pylearn2来做做实验。Python这门神奇的语言几年前学过点皮毛，对于一个习惯使用C++/JAVA的人来说，使用Python简直就是一种折磨，特别是阅读别人的代码。这种弱类型语言最让人纠结的就是不好跟踪代码（应该是我不会跟踪）。不喜欢Python还有一个原因，Python虽有多线程，但是此多线程非彼多线程，不管你开多少个线程，它只使用CPU的一个核，多核对它来说完全没用，而且线程越多速度反而越慢。好了，不废话了，回归正题。

Pylearn2的安装及环境搭建这里就不赘述了，网上资料很多的。对Pylearn2的使用说明中文基本没有（反正我没搜到），英文的还是有一写的。为了节省新手（我也是偷笑）的时间，这里简单介绍一下使用Pylearn构建一个简单的分类器的方法，会了一个应该其他的就好办了，入门是关键。

本文基于[pylearn2-practice](https://github.com/zygmuntz/pylearn2-practice)进行介绍（之所以不根据Quick-start example来讲是因为它结构比较复杂，而且只有训练，没有预测部分，看了后还是不知道如何使用自己的数据，可能是本人智商不够委屈）。

pylearn2-practice文件组织如下

├── adult  
│   ├── test.csv  
│   ├── test\_v.csv  
│   └── train\_v.csv  
├── adult\_dataset.py  
├── adult.yaml  
├── predict.py  
└── README.md

如何使用见README.md

我们先看看训练的配置文件adult.yaml

!obj:pylearn2.train.Train {   #使用类pylearn2.train.Train的实例来训练  
    dataset: &train !obj:adult\_dataset.AdultDataset { #使用adult\_dataset.AdultDataset来提供训练数据  
        path: 'adult/train\_v.csv',  #训练数据文件路径  
        one\_hot: 1  
    },  
      
    model: !obj:pylearn2.models.softmax\_regression.SoftmaxRegression { #选择模型, 可以选择的模型包括RBM, softmax\_regression, SVM等等

        #具体参看<http://deeplearning.net/software/pylearn2/features.html#features>中的Models  
        n\_classes: 2,  
        irange: 0.,  
        nvis: 123,  
    },  
  
  
    algorithm: !obj:pylearn2.training\_algorithms.bgd.BGD {#选择训练算法,具体参看

      #<http://deeplearning.net/software/pylearn2/features.html#features>中的Training algorithms  
        batch\_size: 10000,  
        line\_search\_mode: 'exhaustive',  
        conjugate: 1,  
        monitoring\_dataset:  
            {  
                'train' : \*train,  
                'valid' : !obj:adult\_dataset.AdultDataset {  
                              path: 'adult/test\_v.csv',  
                              one\_hot: 1  
                },  
                'test'  : !obj:adult\_dataset.AdultDataset {  
                              path: 'adult/test.csv',  
                              one\_hot: 1  
                }  
            },  
        termination\_criterion: !obj:pylearn2.termination\_criteria.MonitorBased {  
            channel\_name: "valid\_y\_misclass"  
        }  
    },  
      
    extensions: [  
        !obj:pylearn2.train\_extensions.best\_params.MonitorBasedSaveBest {  
             channel\_name: 'valid\_y\_misclass',  
             save\_path: "softmax\_regression\_best.pkl"  
        },  
    ],  
    save\_path: "softmax\_regression.pkl", #训练得到的模型保存到softmax\_regression.pkl这个文件  
    save\_freq: 1  
      
}

细心的你可能会发现上面的配置文件其实是给类的构造函数指定参数, 事实上Pylearn2就是使用PyYAML根据上面配置信息实例化一个Train对象.

我们再看看adult\_dataset.py

**[python]** [view plaincopy](http://blog.csdn.net/tangchenyi/article/details/25076093)[在CODE上查看代码片](https://code.csdn.net/snippets/328646)

1. **import** csv
2. **import** numpy as np
3. **import** os
5. **from** pylearn2.datasets.dense\_design\_matrix **import** DenseDesignMatrix
6. **from** pylearn2.utils **import** serial
7. **from** pylearn2.utils.string\_utils **import** preprocess
9. **class** AdultDataset( DenseDesignMatrix ):
11. **def** \_\_init\_\_(self,
12. path = 'train.csv',
13. one\_hot = False,
14. with\_labels = True,
15. start = None,
16. stop = None,
17. preprocessor = None,
18. fit\_preprocessor = False,
19. fit\_test\_preprocessor = False):

22. self.no\_classes = 2
24. # won't work TODO
25. self.test\_args = locals()
26. self.test\_args['which\_set'] = 'test'
27. self.test\_args['fit\_preprocessor'] = fit\_test\_preprocessor
28. **del** self.test\_args['start']
29. **del** self.test\_args['stop']
30. **del** self.test\_args['self']
32. path = preprocess(path)
33. X, y = self.\_load\_data( path, with\_labels )

36. **if** start **is** **not** None:
37. **assert** which\_set != 'test'
38. **assert** isinstance(start, int)
39. **assert** isinstance(stop, int)
40. **assert** start >= 0
41. **assert** start < stop
42. **assert** stop <= X.shape[0]
43. X = X[start:stop, :]
44. **if** y **is** **not** None:
45. y = y[start:stop, :]

48. super(AdultDataset, self).\_\_init\_\_(X=X, y=y)
50. **if** preprocessor:
51. preprocessor.apply(self, can\_fit=fit\_preprocessor)
53. **def** \_load\_data(self, path, expect\_labels):
55. **assert** path.endswith('.csv')
57. data = np.loadtxt( path, delimiter = ',', dtype = 'int' )
59. **if** expect\_labels:
60. y = data[:,0]
61. X = data[:,1:]
63. # TODO: if one\_hot
64. # 10 is number of possible y values
65. one\_hot = np.zeros((y.shape[0], self.no\_classes ),dtype='float32')
66. **for** i **in** xrange( y.shape[0] ):
67. label = y[i]
68. **if** label == 1:
69. one\_hot[i,1] = 1.
70. **else**:
71. one\_hot[i,0] = 1.
73. y = one\_hot
74. **else**:
75. X = data
76. y = None
78. **return** X, y

Adultataset继承于DenseDesignMatrix, 通过函数\_load\_data将训练/测试数据加载到X, y中, x基本没有变化, 但是y进行了相应的变换, 类别数决定了y的维数, 每一维代表一种类别,假如某条数据属于第N类,则y的第N维为1, 其他维为0(这里第N维从1开始计数,而不是从0开始计数). 然后将X, y赋给父类的X,y, 如果设置了preprocessor还需要执行preprocessor.

再看看predict.py.

**[python]** [view plaincopy](http://blog.csdn.net/tangchenyi/article/details/25076093)[在CODE上查看代码片](https://code.csdn.net/snippets/328646)

1. **import** sys
2. **import** os
3. **from** pylearn2.utils **import** serial
4. **from** pylearn2.config **import** yaml\_parse
5. **from** adult\_dataset **import** AdultDataset
7. **try**:
8. model\_path = sys.argv[1]
9. test\_path = sys.argv[2]
10. out\_path = sys.argv[3]
11. **except** IndexError:
12. **print** "Usage: predict.py <model file> <test file> <output file>"
13. quit()
15. **try**:
16. model = serial.load( model\_path )
17. **except** Exception, e:
18. **print** model\_path + "doesn't seem to be a valid model path, I got this error when trying to load it: "
19. **print** e

22. #dataset = yaml\_parse.load( model.dataset\_yaml\_src )
23. #dataset = dataset.get\_test\_set()
25. # or maybe specify test in yaml
26. dataset = AdultDataset( path = test\_path, one\_hot = True )
28. # use smallish batches to avoid running out of memory
29. batch\_size = 100
30. model.set\_batch\_size(batch\_size)
32. # dataset must be multiple of batch size of some batches will have
33. # different sizes. theano convolution requires a hard-coded batch size
34. m = dataset.X.shape[0]
35. extra = batch\_size - m % batch\_size
36. **assert** (m + extra) % batch\_size == 0
37. **import** numpy as np
38. **if** extra > 0:
39. dataset.X = np.concatenate((dataset.X, np.zeros((extra, dataset.X.shape[1]),
40. dtype=dataset.X.dtype)), axis=0)
41. **assert** dataset.X.shape[0] % batch\_size == 0

44. X = model.get\_input\_space().make\_batch\_theano()
45. Y = model.fprop(X)
47. **from** theano **import** tensor as T
49. y = T.argmax(Y, axis=1)
51. **from** theano **import** function
53. f = function([X], y)

56. y = []
58. **for** i **in** xrange(dataset.X.shape[0] / batch\_size):
59. x\_arg = dataset.X[i\*batch\_size:(i+1)\*batch\_size,:]
60. **if** X.ndim > 2:
61. x\_arg = dataset.get\_topological\_view(x\_arg)
62. y.append(f(x\_arg.astype(X.dtype)))
64. y = np.concatenate(y)
65. **assert** y.ndim == 1
66. **assert** y.shape[0] == dataset.X.shape[0]
67. # discard any zero-padding that was used to give the batches uniform size
68. y = y[:m]
70. class\_mapping = { 0: -1, 1: 1 }
72. out = open(out\_path, 'w')
73. **for** i **in** xrange(y.shape[0]):
74. p = y[i]
75. p = class\_mapping[p]
76. out.write( '%d\n' % ( p ))
77. out.close()

第一步从测试数据文件中获取数据dataset = AdultDataset( path = test\_path, one\_hot = True ),

然后就是对测试数据分批次(如果测试数据不是特别多, 内存够大的话就没必要分批次),

接着是构建 Theano function(深入了解见Theano说明文档)

X = model.get\_input\_space().make\_batch\_theano()  #X为输入的特征  
Y = model.fprop(X)  #fprop(x)使用模型预测

from theano import tensor as T

y = T.argmax(Y, axis=1)  
from theano import function  
  
f = function([X], y)

 然后就是调用 f 对 x 进行预测了

到这里,分类器构建完成, 这么看来,其实还是蛮简单的, 虽然简单,但是我可是花了大半天的功夫, 一开始花了大量时间看Quick-start example(结果还是一头雾水).由此说明一篇好的文档非常重要, 不然你就慢慢去摸索吧.

感谢zygmuntz的无私奉献

# [Deep learning：四十四(Pylearn2中的Quick-start例子)](http://www.cnblogs.com/tornadomeet/p/3424520.html)

**前言：**

　　听说Pylearn2是个蛮适合搞深度学习的库，它建立在Theano之上，支持GPU(估计得以后工作才玩这个，现在木有这个硬件条件)运算，由DL大牛Bengio小组弄出来的，再加上Pylearn2里面已经集成了一部分常见的DL算法，本着很想读读这些算法的源码和细节这一想法，打算学习下Pylearn2的使用. 网上这方面的中文资料简直是太少了，虽然本博文没什么实质内容，但也写贴出来，说不定可以帮到一些初学者。

　　从Bengio的一篇paper: [Pylearn2: a machine learning research library](http://arxiv.org/abs/1308.4214)可以看出，Pylearn2主要是针对机器学习开发者而设计的（说明使用该库的人需要有一定的机器学习背景知识），利用Pylearn2可以灵活设计自己的机器学习模型和算法，可扩展性较强（具体怎么弄？？）。而根据[Pylearn2库的特征(官网)](http://deeplearning.net/software/pylearn2/features.html#features)上的介绍可知，在pylearn2里，有一些常见的数据模块、模型模块、训练算法模块。数据模块中有常见的MNIST, CIFAR10, CIFAR100, STL10, NORB等。DL模型模块包含：RBM系列，AutoEncoder系列，LCC, maxout等。训练算法模块主要是SGD系列。

**Pylearn2安装简单介绍：**

　　好吧，进入正题。首先是库的安装，我是运行在64bit-ubuntu13.10上的。

　　1. 在此之前还需安装Theano(python下进行符号运算的库，类似Numpy，但在多维矩阵处理上功能更强)，安装Theano的方法请参考：[Installing Theano（Bleeding-edge install instruction）](http://deeplearning.net/software/theano/install.html#bleeding-edge-install-instructions)，里面有Ubuntu下安装的链接，按照里面的步骤一步步进行下去就行（期间遇到的各种问题多google吧！）。需要提一下的是，安装成功后，我们需要将Theano升级到开发版本Bleeding-edge下，因为后面的Pylearn2用到了开发版Theano的新特征。具体的升级方法参考网页中的Bleeding-edge install instructions小节。

　　2. Pylearn2的安装可以参考博文[pylearn2安装及测试（lucktroy的csdn博客）](http://blog.csdn.net/lucktroy/article/details/9466369)。主要有3步：

　　a. 在想安装Pylearn2的目录下打开vim,输入命令：

　　git clone git://github.com/lisa-lab/pylearn2.git

　　b. 配置Pylearn2所用数据目录的环境变量（做一些标准实验时，可将数据放入该目录），即在vim里输入命令行:vim ~/.bashrc ,然后在打开的.bashrc文件最后一行加入语句:export PYLEARN2\_DATA\_PATH=YourPath/data 保存后退出。其中的YourPath为你想放入数据的目录全称。接着在vim里执行source ~/.vimrc命令

　　c. 进入pylearn2目录(刚用git下载后会有该文件的)，执行命令:python setup.py build.

**运行Quick-start例子：**

　　安装完Pylearn2后就想弄个sample爽一把，选的是GRBM算法例子，可参考官网的[Quick-start example](http://deeplearning.net/software/pylearn2/tutorial/index.html)教程。这个例子中主要有3个步骤（如果实验过程中出现一些问题，可以参考下本博文的附录，看能否提供一些帮助）：

　　步骤一：创建数据。

　　在YourPath/pylearn2/scripts/tutorials/grbm\_smd/ 目录下执行下列命令：python make\_dataset.py

　　从make\_dataset.py的源码中可以看出，这里使用的是CIFAR10图片库（[http://www.cs.toronto.edu/~kriz/cifar.html（CIFAR10数据库）](http://www.cs.toronto.edu/~kriz/cifar.html)），为32\*32大小的彩色图片，共5w个训练样本和1w个测试样本。训练grbm的patch大小为8\*8的，有15w个patch。当然还对该图片库进行了一些预处理，比如ZCA白化等等。最后将预处理好的结果保存为pickle文件（pickle是python中用于序列处理的模块，保存数据为.pkl格式到硬盘，下次要使用该数据时可重新加载）：cifar10\_preprocessed\_train.pkl.

　　步骤二：GRBM模型参数的训练。

　　使用的命令（还是在原来的目录下）为：python ../../ train.py cifar\_grbm\_smd.yaml

　　其中的cifar\_grbm\_smd.yaml文件是该实验的配置文件，需要配置数据，模型，算法3个模块的一些参数，yaml文件是我们与pylearn2打交道的文件，如果是使用常见的深度学习模型和常见的优化算法来做实验的话，则只需把配置好这个.yaml文件就可以了。这可以简化不少工作。下面来看看这个cifar\_grbm\_smd.yaml的代码及一些注释，关于yaml语法的简单介绍可参考：[YAML for Pylearn2](http://deeplearning.net/software/pylearn2/yaml_tutorial/index.html#yaml-tutorial). 另外，如果想了解GRBM，则可参考网友博文：[DeepLearning（深度学习）原理与实现（四）](http://blog.csdn.net/marvin521/article/details/8906278)，写得很不错。

[复制代码](javascript:void(0);)

# pylearn2 tutorial example: cifar\_grbm\_smd.yaml by Ian Goodfellow

#

# Read the README file before reading this file

#

# This is an example of yaml file, which is the main way that an experimenter

# interacts with pylearn2.

#

# A yaml file is very similar to a python dictionary, with a bit of extra

# syntax.

# The !obj tag allows us to create a specific class of object. The text after

# the : indicates what class should be loaded. This is followed by a pair of

# braces containing the arguments to that class's \_\_init\_\_ method.

#

# Here, we allocate a Train object, which represents the main loop of the

# training script. The train script will run this loop repeatedly. Each time

# through the loop, the model is trained on data from a training dataset, then

# saved to file.

!obj:pylearn2.train.Train {

# The !pkl tag is used to create an object from a pkl file. Here we retrieve

# the dataset made by make\_dataset.py and use it as our training dataset.

dataset: !pkl: "cifar10\_preprocessed\_train.pkl",

# Next we make the model to be trained. It is a Binary Gaussian RBM

model: !obj:pylearn2.models.rbm.GaussianBinaryRBM {

# The RBM needs 192 visible units (its inputs are 8x8 patches with 3

# color channels)

nvis : 192,

# We'll use 400 hidden units for this RBM. That's a small number but we

# want this example script to train quickly.

nhid : 400,

# The elements of the weight matrices of the RBM will be drawn

# independently from U(-0.05, 0.05)

irange : 0.05,

# There are many ways to parameterize a GRBM. Here we use a

# parameterization that makes the correspondence to denoising

# autoencoders more clear.

energy\_function\_class : !obj:pylearn2.energy\_functions.rbm\_energy.grbm\_type\_1 {},

# Some learning algorithms are capable of estimating the standard

# deviation of the visible units of a GRBM successfully, others are not

# and just fix the standard deviation to 1. We're going to show off

# and learn the standard deviation.

learn\_sigma : True,

# Learning works better if we provide a smart initialization for the

# parameters. Here we start sigma at .4 , which is about the same

# standard deviation as the training data. We start the biases on the

# hidden units at -2, which will make them have fairly sparse

# activations.

init\_sigma : .4,

init\_bias\_hid : -2.,

# Some GRBM training algorithms can't handle the visible units being

# noisy and just use their mean for all computations. We will show off

# and not use that hack here.

mean\_vis : False,

# One hack we will make is we will scale back the gradient steps on the

# sigma parameter. This way we don't need to worry about sigma getting

# too small prematurely (if it gets too small too fast the learning

# signal gets weak).

sigma\_lr\_scale : 1e-3

},

# Next we need to specify the training algorithm that will be used to train

# the model. Here we use stochastic gradient descent.

algorithm: !obj:pylearn2.training\_algorithms.sgd.SGD {

# The learning rate determines how big of steps the learning algorithm

# takes. Here we use fairly big steps initially because we have a

# learning rate adjustment scheme that will scale them down if

# necessary.

learning\_rate : 1e-1,

# Each gradient step will be based on this many examples

batch\_size : 5,

# We'll monitor our progress by looking at the first 20 batches of the

# training dataset. This is an estimate of the training error. To be

# really exhaustive, we could use the entire training set instead,

# or to avoid overfitting, we could use held out data instead.

monitoring\_batches : 20,

monitoring\_dataset : !pkl: "cifar10\_preprocessed\_train.pkl",

# Here we specify the objective function that stochastic gradient

# descent should minimize. In this case we use denoising score

# matching, which makes this RBM behave as a denoising autoencoder.

# See

# Pascal Vincent. "A Connection Between Score Matching and Denoising

# Auutoencoders." Neural Computation, 2011

# for details.

cost : !obj:pylearn2.costs.ebm\_estimation.SMD {

# Denoising score matching uses a corruption process to transform

# the raw data. Here we use additive gaussian noise.

corruptor : !obj:pylearn2.corruption.GaussianCorruptor {

stdev : 0.4

},

},

# We'll use the monitoring dataset to figure out when to stop training.

#

# In this case, we stop if there is less than a 1% decrease in the

# training error in the last epoch. You'll notice that the learned

# features are a bit noisy. If you'd like nice smooth features you can

# make this criterion stricter so that the model will train for longer.

# (setting N to 10 should make the weights prettier, but will make it

# run a lot longer)

termination\_criterion : !obj:pylearn2.termination\_criteria.MonitorBased {

prop\_decrease : 0.01,

N : 1,

},

# Let's throw a learning rate adjuster into the training algorithm.

# To do this we'll use an "extension," which is basically an event

# handler that can be registered with the Train object.

# This particular one is triggered on each epoch.

# It will shrink the learning rate if the objective goes up and increase

# the learning rate if the objective decreases too slowly. This makes

# our learning rate hyperparameter less important to get right.

# This is not a very mathematically principled approach, but it works

# well in practice.

},

extensions : [!obj:pylearn2.training\_algorithms.sgd.MonitorBasedLRAdjuster {}],

#Finally, request that the model be saved after each epoch

save\_freq : 1

}

[复制代码](javascript:void(0);)

　　由上面的yaml文件可知，yaml中的内容有点类似python中的字典：一个关键字key对应一个值value。而这些key都是对应类的构造函数\_\_init\_\_()中的参数，也就是说将这些value传入到这些构造函数中，并由其对象接收。上面yaml代码中data来源于步骤一的cifar10\_preprocessed\_train.pkl文件。model来源于pylearn2库下的pylearn2.models.rbm.GaussianBinaryRBM类，而algorithm来源于pylearn2库下的pylearn2.training\_algorithms.sgd.SGD类。

　　当.yaml文件都配置好后，我们就需要启动对应的程序来训练参数了，train.py就是执行的这个功能的，其代码为：

[复制代码](javascript:void(0);)

#!/usr/bin/env python

"""

Script implementing the logic for training pylearn2 models.

This is intended to be a "driver" for most training experiments. A user

specifies an object hierarchy in a configuration file using a dictionary-like

syntax and this script takes care of the rest.

For example configuration files that are consumable by this script, see

pylearn2/scripts/train\_example

pylearn2/scripts/autoencoder\_example

"""

\_\_authors\_\_ = "Ian Goodfellow"

\_\_copyright\_\_ = "Copyright 2010-2012, Universite de Montreal"

\_\_credits\_\_ = ["Ian Goodfellow", "David Warde-Farley"]

\_\_license\_\_ = "3-clause BSD"

\_\_maintainer\_\_ = "Ian Goodfellow"

\_\_email\_\_ = "goodfeli@iro"

# Standard library imports

import argparse

import gc

import logging

import os

# Third-party imports

import numpy as np

# Local imports

from pylearn2.utils import serial

from pylearn2.utils.logger import (

CustomStreamHandler, CustomFormatter, restore\_defaults

)

class FeatureDump(object):

def \_\_init\_\_(self, encoder, dataset, path, batch\_size=None, topo=False):

self.encoder = encoder

self.dataset = dataset

self.path = path

self.batch\_size = batch\_size

self.topo = topo

def main\_loop(self):

if self.batch\_size is None:

if self.topo:

data = self.dataset.get\_topological\_view()

else:

data = self.dataset.get\_design\_matrix()

output = self.encoder.perform(data)

else:

myiterator = self.dataset.iterator(mode='sequential',

batch\_size=self.batch\_size,

topo=self.topo)

chunks = []

for data in myiterator:

chunks.append(self.encoder.perform(data))

output = np.concatenate(chunks)

np.save(self.path, output)

def make\_argument\_parser():

parser = argparse.ArgumentParser(

description="Launch an experiment from a YAML configuration file.",

epilog='\n'.join(\_\_doc\_\_.strip().split('\n')[1:]).strip(),

formatter\_class=argparse.RawTextHelpFormatter

) #parser是用来接收参数的

parser.add\_argument('--level-name', '-L',

action='store\_true',

help='Display the log level (e.g. DEBUG, INFO) '

'for each logged message')

parser.add\_argument('--timestamp', '-T',

action='store\_true',

help='Display human-readable timestamps for '

'each logged message')

parser.add\_argument('--verbose-logging', '-V',

action='store\_true',

help='Display timestamp, log level and source '

'logger for every logged message '

'(implies -T).')

parser.add\_argument('--debug', '-D',

action='store\_true',

help='Display any DEBUG-level log messages, '

'suppressed by default.')

parser.add\_argument('config', action='store', #按照格式输入参数，比如这里的输入的参数会保存在config中

choices=None,

help='A YAML configuration file specifying the '

'training procedure')

return parser

if \_\_name\_\_ == "\_\_main\_\_":

parser = make\_argument\_parser()

args = parser.parse\_args() #读取传入进来的参数，这里是直接在命令行读取该文件,参数放入args.config中

train\_obj = serial.load\_train\_file(args.config) #serial.load\_train\_file()函数最后返回的是：  
　　　　　　　　　　# return yaml\_parse.load\_path(args.config) 也就是说调用的是ymal\_parse.load\_path()函数。返回的是一个train类的对象。  
　　　　　　　　　　# 其中的ymal\_parse是pylearn2.config中的函数。

return yaml\_parse.load\_path(config\_file\_path)

try:

iter(train\_obj) #iter()是个迭代器函数

iterable = True

except TypeError as e:

iterable = False

# Undo our custom logging setup.

restore\_defaults()

# Set up the root logger with a custom handler that logs stdout for INFO

# and DEBUG and stderr for WARNING, ERROR, CRITICAL.

root\_logger = logging.getLogger() #logging主要是python中用于处理日志的模块,这里是返回一个logger实例，由于没有指定name，所以是root logger

if args.verbose\_logging:

formatter = logging.Formatter(fmt="%(asctime)s %(name)s %(levelname)s "

"%(message)s")

handler = CustomStreamHandler(formatter=formatter)

else:

if args.timestamp:

prefix = '%(asctime)s '

else:

prefix = '' #这里为空

formatter = CustomFormatter(prefix=prefix, only\_from='pylearn2')

handler = CustomStreamHandler(formatter=formatter)

root\_logger.addHandler(handler) #给root\_lgger添加handler来帮助处理日志

# Set the root logger level.

if args.debug:

root\_logger.setLevel(logging.DEBUG)

else:

root\_logger.setLevel(logging.INFO) #给root\_logger设置级别，为INFO级别，因为每个日志消息都会关联一个级别

if iterable: #enumerate()为对一个list或者array既要遍历索引又要遍历元素时使用

for number, subobj in enumerate(iter(train\_obj)):#train\_obj里面装的是ymal文件内容，类似字典

# Publish a variable indicating the training phase.

phase\_variable = 'PYLEARN2\_TRAIN\_PHASE'

phase\_value = 'phase%d' % (number + 1)

os.environ[phase\_variable] = phase\_value

os.putenv(phase\_variable, phase\_value)

# Execute this training phase.

subobj.main\_loop()

# Clean up, in case there's a lot of memory used that's

# necessary for the next phase.

del subobj

gc.collect()

else:

train\_obj.main\_loop() #因为train\_obj中已经包含了数据，模型，算法，所以调用main\_loop()后表示采用对应算法用对应数据在对应的模型上训练

#直到满足迭代终止条件

[复制代码](javascript:void(0);)

　　其中最核心的就是main\_loop()函数了，在调用main\_loop()后，程序会自动用algorithm对象使用model对象在data上来训练参数了。至于具体该函数是怎样将data, model, algorithm联系起来的呢？我们可以试着去读一下源码：

　　首先是由train\_obj.main\_loop()函数将data, model, algorithm联系起来的。从名字train\_obj可以看出它是一个某个类的对象，猜测应该是Pylearn2下的Train类对象，因为在库Pylearn2的子目录下有个model为train.py，该文件有个Train类，并且这个Train类有一个方法：main\_loop()。看来一切符合猜测，那么是否真是的呢？

　　首先来看看train\_obj从哪里来的(因为main\_loop()是由train\_obj来调用的)。由上面的程序可知：train\_obj = serial.load\_train\_file(args.config), 需要跟踪serial, 找到serial.load\_train\_file()的源代码，最后一句为：return yaml\_parse.load\_path(args.config). 继续跟踪发现load\_path()函数里面调用了load()函数，而里面最调用的是yaml.load()函数，由源码中的注释可知它是将.yaml配置文件转换成一个graph, 而这个graph应该就是一个Train对象...

　　好吧，到了该看main\_loop()的内容了：

[复制代码](javascript:void(0);)

def main\_loop(self):

"""

Repeatedly runs an epoch of the training algorithm, runs any

epoch-level callbacks, and saves the model.

"""

if self.algorithm is None:

self.model.monitor = Monitor.get\_monitor(self.model)

self.setup\_extensions()

self.run\_callbacks\_and\_monitoring()

while True:

rval = self.model.train\_all(dataset=self.dataset)

if rval is not None:

raise ValueError("Model.train\_all should not return anything. Use Model.continue\_learning to control whether learning continues.")

self.model.monitor.report\_epoch()

if self.save\_freq > 0 and self.model.monitor.epochs\_seen % self.save\_freq == 0:

self.save()

continue\_learning = self.model.continue\_learning()

assert continue\_learning in [True, False, 0, 1]

if not continue\_learning:

break

else:

self.algorithm.setup(model=self.model, dataset=self.dataset) #这一句将data,model, dataset联系起来了

self.setup\_extensions() #和.yaml文件中的extensions项联系起来了

if not hasattr(self.model, 'monitor'):

# TODO: is this really necessary? I just put this error here

# to prevent an AttributeError later, but I think we could

# rewrite to avoid the AttributeError

raise RuntimeError("The algorithm is responsible for setting"

" up the Monitor, but failed to.")

if len(self.model.monitor.\_datasets)>0:

# This monitoring channel keeps track of a shared variable,

# which does not need inputs nor data.

self.model.monitor.add\_channel(name="monitor\_seconds\_per\_epoch",

ipt=None,

val=self.monitor\_time,

data\_specs=(NullSpace(), ''),

dataset=self.model.monitor.\_datasets[0])

self.run\_callbacks\_and\_monitoring()

while True: #循环中，直到满足终止条件

with log\_timing(log, None, final\_msg='Time this epoch:',

callbacks=[self.monitor\_time.set\_value]):

rval = self.algorithm.train(dataset=self.dataset) #算法训练的核心函数

if rval is not None:

raise ValueError("TrainingAlgorithm.train should not return anything. Use TrainingAlgorithm.continue\_learning to control whether learning continues.")

self.model.monitor.report\_epoch()

self.run\_callbacks\_and\_monitoring()

if self.save\_freq > 0 and self.model.monitor.\_epochs\_seen % self.save\_freq == 0:

self.save()

continue\_learning = self.algorithm.continue\_learning(self.model) #终止条件测试

assert continue\_learning in [True, False, 0, 1]

if not continue\_learning:

break

self.model.monitor.training\_succeeded = True

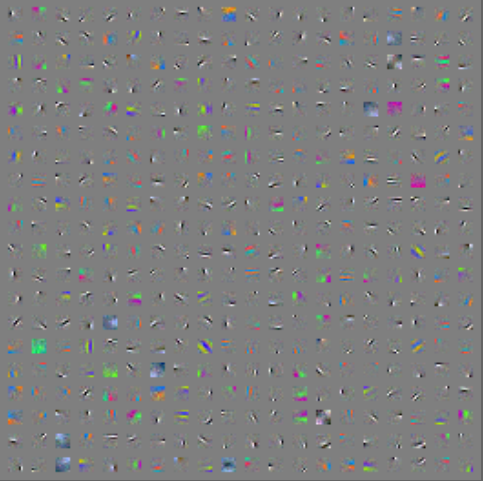
if self.save\_freq > 0:

self.save()

[复制代码](javascript:void(0);)

　　步骤三:

　　这部分就是看结果显示了，执行命令：python ../../show\_weights.py cifar\_grbm\_smd.pkl 比如我这里执行后的结果显示如下：



　　当然了你还可以使用plot\_monitor.py来看一些对应结果。

**总结：**

　　当使用Pylearn2中已有的一些DL模型，采用其中已有的一些优化算法来做实验时，我们只需要配置好实验的.yaml文件即可，调参过程就是不断更改.ymal中的配置。但是如果需要采用自己新提出来的DL模型，或者采用自己新提出的目标函数及优化方法，则还需要自己写出对应的类，具体这部分该怎么做（比如说怎样去实现这个类，接口怎样设计，.ymal文件需要更改哪些地方等），本人暂时没任何经验。希望懂这些的可以大家可贡献贡献下想法，交流交流下。网上有个教程是把Pylearn2当做通常的python库来用，实现了一个异或网络，很不错，见：[Neural network example using Pylearn2.](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

　　另外，分析Pylearn2的源码可知，每个algorithm中，必须有下面4个函数:\_\_init(), setup(), train(),  continue\_training(), 作用分别为构造函数, 根据model建立网络的结构，模型参数的训练，模型训练终止处理。model模块中，应该也有一些统一的函数。

**附录：**

　　我实验过程中可能出现的一些错误处理：

**A:**

　　如果执行 python make\_dataset.py后出现错误：

   　raise IOError("permission error creating %s" % filepath) IOError: permission error creating cifar10\_preprocessed\_train.pkl

　　看错误提示应该是权限问题，这时改为命令：

　　sudo python make\_dataset.py

　　如果继续出现错误：

　　pylearn2.datasets.exc.NoDataPathError: You need to define your PYLEARN2\_DATA\_PATH environment variable. If you are using a computer at LISA, this should be set to /data/lisa/data.

　　说明PYLEARN2\_DATA\_PATH环境变量没有设置，但是前面却是设置了啊！为什么呢？有可能是你设置环境变量时用的是root权限，而执行该命令只是普通用户。如果切换到root下再执行 root#:python make\_dataset.py成功！生成了cifar10\_preprocessed\_train.pkl

　　但是后面执行：../../train.py cifar\_grbm\_smd.yaml出现错误：ImportError: Could not import pylearn2.models but could import pylearn2. Original exception: No module named compat.python2x

　　到这里基本可以确定是权限问题，解决方法是：重新用普通用户安装了下pylearn2,设置好环境变量，放着好下载的数据后，执行（普通用户下）：

　　python make\_dataset

　　则成功生成了cifar10\_preprocessed\_train.pkl 可恶的是后续的../../train.py cifar\_grbm\_smd.yaml还是会出现刚刚的错误。

　　当然了这个问题主要是因为Theano的版本不对，在使用pylearn2时，应该使用development版本的Theano，按照本文前面的方法更新下Theano即可。

 B.

　　如果在显示权值阶段，当执行下面命令后：sudo python ../../show\_weights.py cifar\_grbm\_smd.pkl.可能会出现下面提示：

　　You need to choose an image viewer program that pylearn2 should use. Then tell pylearn2 to usethat image viewer program by defining your PYLEARN2\_VIEWER\_COMMAND environment variable.You need to choose PYLEARN\_VIEWER\_COMMAND such that running ${PYLEARN2\_VIEWER\_COMMAND} image.png

in a command prompt on your machine will do the following:

    -open an image viewer in a new process.

    -not return until you have closed the image.

Acceptable commands include:

    gwenview

    eog --new-instance

This is assuming that you have gwenview or a version of eog that supports --new-instance

......

……

　　这说明pylearn2中没有指定图片显示的软件。首先安装gwenview软件：sudo apt-get Install gwenview.

　　然后设置一下PYLEARN2\_VIEWER\_COMMAND环境变量。vim ~/.bashrc 在最后一行加入gwenview的安装目录，比如我按照默认的安装目录加入的为：

　　export PYLEARN2\_VIEWER\_COMMAND=/usr/bin/gwenview

　　保存好后执行source ~/.bashrc

**参考资料：**

[Pylearn2: a machine learning research library](http://arxiv.org/abs/1308.4214)

[Pylearn2库的特征(官网)](http://deeplearning.net/software/pylearn2/features.html#features)

[Installing Theano（Bleeding-edge install instruction）](http://deeplearning.net/software/theano/install.html#bleeding-edge-install-instructions)

[pylearn2安装及测试（lucktroy的csdn博客）](http://blog.csdn.net/lucktroy/article/details/9466369)

[Quick-start example](http://deeplearning.net/software/pylearn2/tutorial/index.html)

[http://www.cs.toronto.edu/~kriz/cifar.html（CIFAR10数据库）](http://www.cs.toronto.edu/~kriz/cifar.html)

[YAML for Pylearn2](http://deeplearning.net/software/pylearn2/yaml_tutorial/index.html#yaml-tutorial)

[DeepLearning（深度学习）原理与实现（四）](http://blog.csdn.net/marvin521/article/details/8906278)

[Neural network example using Pylearn2.](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

# [关于pylearn2的一些。](http://blog.csdn.net/silence1214/article/details/41312249)

2014-11-20 14:20 750人阅读 [评论](http://blog.csdn.net/silence1214/article/details/41312249#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/silence1214/article/details/41312249#report)

pylearn2是一个类似caffe的库，是建立在python语言之上，使用了theano的，由于我之前使用sgd总是达不到paper上的效果，所以我想学习下这个玩意，上午拿到之后，在win8下使用了下，发现入门有点简单，我就阅读了在axriv上的论文，关于pylearn2的，来大概总结一下吧：

历史：

它是LISP实验室的，这个实验室我就不多说了吧，bengio领导的。pylearn2是LISP实验室的第三个做此类开发的平台了，之前的就不说了。

架构：

包含了Model，learningAlgorithm和Datesets

Model：用来存储参数的，实现了很多成熟的模型，比如RBM，CNN,AUTOENCODER等，尤其是LISP实验室的paper中的模型，它全都实现了的。其实嘛，就是我们理解的模型。人家说的也很简单，模型里面主要是线性操作（非线性包括在里面了吧）

学习算法：调整Model中的参数的，并且还有别的功能，比如建立Monitor，来检测学习过程中的一些变化吗，比如精度曲线啥的，里面有几个实现好的类，比如SGD。我看用的最多的也就是SGD了吧，实现的还有BGD

数据集：就是我们训练算法用的数据，只是对原始数据和模型之间做了个接口吧，让模型对数据透明，因为不同的实现，毕竟数据类型很多，理论上支持任何类型的数据输入格式。如果数据时矩阵直接用DenseDesignmatrix类，如果数据在Numpy或者pickle format的直接用就是了，更大的数据也支持HDFS格式，同时这个模块还可以对数据进行ZCA,PCA的预处理。

其实大概也就这些了，具体使用的话，是用配置文件来实现的，配置的主要模块也就是上面3个地方，每个地方会涉及到一些参数，具体的还要具体在看。

啰嗦完了。。

# Neural network example using Pylearn2

[7 Replies](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/#comments)

I was recently looking into using a neural network for a project so I started looking into some of the available Python libraries. The one I ended up using was [Pylearn2](http://deeplearning.net/software/pylearn2/) which is a fast and powerful library for machine learning that is mainly built upon [Theano](http://deeplearning.net/software/theano/).

Pylearn2 is under development and is still a bit rough around the edges and the documentation is limited and in some instances not correct. The recommended way of using it is by writing YAML scripts and if you are ok with that you can probably manage with the existing documentation. But if you, like me, want to use it as a standard Python library you have better be prepared to read the code. One thing that would have saved me some time was a complete example of how to use Pylearn2 as a standalone library, so what follows is a simple example of creating a neural network for solving the XOR problem.

The XOR problem is stated as follows, create a neural network that given two binary inputs, 0 or 1, the output should be a 1 if exactly one of the inputs are 1 and 0 otherwise.

Pylearn2 has a dataset implementation that in its simplest form needs a collection of datapoints in a 2D Numpy array named X and a 2D array named y containing the answers. We can create a dataset by creating a new class that inherits from DenseDesignMatrix:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| class XOR(DenseDesignMatrix):      def \_\_init\_\_(self):          self.class\_names = ['0', '1']          X = [[randint(0, 1), randint(0, 1)] for \_ in range(1000)]          y = []          for a, b in X:              if a + b == 1:                  y.append([0, 1])              else:                  y.append([1, 0])          X = np.array(X)          y = np.array(y)          super(XOR, self).\_\_init\_\_(X=X, y=y)    ds = XOR() |

Note that we are using two columns in the target variable y, a 1 in the first column signifies a output of 0 and a 1 in the second columns signifies a output of 1.

Next we need to create the layers in the neural net. To be able to solve the XOR problem we need a hidden layer with at least two neurons:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| hidden\_layer = mlp.Sigmoid(layer\_name='hidden', dim=2, irange=.1, init\_bias=1.) |

The hidden layer uses a standard [sigmoid](https://en.wikipedia.org/wiki/Sigmoid_function) activation function and the weights are initialized in the range -0.1 to 0.1 (using the irange argument). We also add a bias to the two neurons with value 1.0.

We use a [softmax](https://en.wikipedia.org/wiki/Softmax_activation_function) layer with two nodes as output layer. The output from the two nodes is between 0 and 1 and the sum of the output from all nodes in the layer is 1.

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| output\_layer = mlp.Softmax(2, 'output', irange=.1) |

To train the network we use a [Stochastic Gradient Descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent)(SGD) method which we initialize like this:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| trainer = sgd.SGD(learning\_rate=.05, batch\_size=10, termination\_criterion=EpochCounter(400)) |

We use a simple termination criterion that runs for 400 epochs, more advanced termination criteria are of course available.

To initialize the neural network and setup the training we do like this:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| layers = [hidden\_layer, output\_layer]  ann = mlp.MLP(layers, nvis=2)  trainer.setup(ann, ds) |

We put the layers in the Multi-Layer Perceptron class with two inputs and then setup the trainer with the class and the dataset.

We then train the neural network until the termination criteria is reached:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| while True:      trainer.train(dataset=ds)      ann.monitor.report\_epoch()      ann.monitor()      if not trainer.continue\_learning(ann):          break |

After the training is complete we of course wants to test that it works. We do this by using the fprop-method that takes the inputs as Theano variables:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| inputs = np.array([[0, 1]])  print ann.fprop(theano.shared(inputs, name='inputs')).eval() |

This should yield a answer like this:

[?](http://www.arngarden.com/2013/07/29/neural-network-example-using-pylearn2/)

|  |
| --- |
| [[ 0.00526688  0.99473312]] |

Meaning that the network correctly predicts that the output should be a 1.

See [here](https://gist.github.com/arngarden/6087798) for the complete source code of the example.