Chuck Anderson

Professor, Computer Science, Colorado State University

Numpy versus Pytorch

August 26, 2017 by anderson

Here we compare the accuracy and computation time of the training of simple fully-connected neural networks using numpy and pytorch implementations and applied to the MNIST data set. The Adam optimization algorithm in numpy and pytorch are compared, as well as the Scaled Conjugate Gradient optimization algorithm in numpy.

The original notebook is available here.

Additional comments and explanations will be added shortly. If you have suggestions or corrections, please write to chuck.anderson@colostate.edu.

In [ ]:

```
!nvidia-smi
```

Search

Recent Posts

Comparing Numpy, Pytorch, and autograd on CPU and GPU
October 13, 2017

Numpy versus Pytorch
August 26, 2017

```
Mon Jul 17 15:16:36 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375
|-------------------------------+----------------------+----------------------
| GPU  Name        Persistence-M| Bus-Id        Disp
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-
|===============================+======================+=======
|   0  NVS 315            Off  | 0000:02:00.0     N/A |
| 30%   42C    P8    N/A /  N/A |     35MiB /   964MiB
+-------------------------------+----------------------+----------------------
|   1  GeForce GTX 980    Off  | 0000:03:00.0     Of
| 26%   36C    P8    13W / 180W |    281MiB /  4038
+-------------------------------+----------------------+----------------------

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU M
|  GPU       PID  Type  Process name
|=============================================================================
|    0              Not Supported
|    1      8001    C   .../anderson/anaconda3/envs/i
+-----------------------------------------------------------------------------+
```

In [ ]:

```
import numpy as np
import pickle
import gzip
import json
import time
import sys
import subprocess

# For my numpy neural network implementation
import neuralnetworks as nn
```

Fast Reinforcement Learning After Pretraining
June 30, 2017

Javascript for Zooming and Panning in a Canvas
September 1, 2012

Record Your Own Lectures
September 1, 2011

Tags

C (1) climate (1) deep learning (2) GPU (1) graphics (1) image analysis (1)

```
# for pytorch
import os
import torch
import torch.nn as tnn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.autograd import Variable

# for reading and plotting results
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
#####################################
## Read mnist data into dataTrain

with gzip.open('mnist.pkl.gz', 'rb') as f:
    train_set, valid_set, test_set = pickle.load(f, enc
    Xtrain = np.array(train_set[0])
    Xval = np.array(valid_set[0])
    Xtest = np.array(test_set[0])
    Ttrain = np.array(train_set[1]).reshape((-1, 1))
    Tval = np.array(valid_set[1]).reshape((-1, 1))
    Ttest = np.array(test_set[1]).reshape((-1, 1))

# to match with main-gpu.py in res/pytorch...01...
Xtrain = np.vstack((Xtrain, Xval))
Ttrain = np.vstack((Ttrain, Tval))
dataTrain = np.hstack((Xtrain, Ttrain))  # so we ca
dataTrain = dataTrain.astype(np.float32)
nSamples = dataTrain.shape[0]
```

In [ ]:

javascript (1)

matlab (3)

neural networks (2)

python (2)

pytorch (2)

reinforcement learning (2)

Recent Comments

Categories

```
#####################################
## Write results

def writeResults(filename, timeAcc, label):
    if filename == 'stdout':
        f = sys.stdout
    else:
        f = open(filename, 'a')
    f.write(label+'\n')
    f.write(str(len(timeAcc)) + '\n')
    for ta in timeAcc:
        f.write('{:.2f} {:.3f}\n'.format(ta[0], ta[1]))
    if filename != 'stdout':
        f.close()
```

In [ ]:

```
def runnumpy(batchSize=None, numEpochs=10,

    label = 'Numpy '
    label += 'Adam' if useAdam else 'SCG'
    label += ' batch {} epochs {:d} hids {} nIter {:d}'.
    label += ' ReLU ' if useRelu else ' Tanh '
    label += time.strftime('%m/%d/17-%H:%M')

    Xtrain = dataTrain[:,:-1]
    Ttrain = dataTrain[:,-1:]

    nnet = nn.NeuralNetworkClassifier([Xtrain.shap
                                       np.arange(10), useRelu=use
    # NOT STANDARDIZING THE INPUTS!!!
    nnet.setStandardize(False)

    secsAcc = []
```

atmospheric science (1)
deep learning (2)
javascript (1)
lecture recording (1)
neural networks (2)
reinforcement learning (1)
reinforcement learning (1)
Uncategorized (1)

Archives

October 2017

```
    # numEpochs determines number of breaks du
    if batchSize is None:

        startTime = time.time()
        for i in range(numEpochs):
            Xtrain = dataTrain[:, :-1]
            Ttrain = dataTrain[:, -1:]
            nnet.train(Xtrain, Ttrain, nIterations=nItera
            ptest = nnet.use(Xtest)
            secsAcc.append([time.time() - startTime, np

    else:  # numpyg on batches

        nSamples = dataTrain.shape[0]
        if nSamples % batchSize != 0:
            print('WARNING: nSamples {} is not divisib
                nSampmles, batchSize))
        nBatches = nSamples // batchSize
        startTime = time.time()
        for epoch in range(numEpochs):
            np.random.shuffle(dataTrain)
            for traini in range(0, nSamples, batchSize):
                Xtrain = dataTrain[traini:traini+batchSize
                Ttrain = dataTrain[traini:traini+batchSize
                nnet.train(Xtrain, Ttrain, restart=True, nI
            ptest = nnet.use(Xtest)
            secsAcc.append([time.time() - startTime, np

    writeResults(resultsFilename, secsAcc, label)
    if numEpochs <= 10:
        writeResults('stdout', secsAcc, label)
```

In [ ]:

```
def runpytorch(batchSize=100, numEpochs=10, h
```

August 2017
June 2017
September 2012
September 2011
January 2007
October 2001
May 2001
June 1997
August 1996

Meta

Log in
Entries RSS
Comments RSS
WordPress.org

```
        learningRate=0.001, nIterations=100, use

if useGPU:
    os.environ['CUDA_VISIBLE_DEVICES'] = '0'
    print('torch.cuda.is_available() is', torch.cuda.

    if not torch.cuda.is_available():
        print('GPU is not available. Not running sg
        return

label = 'Pytorch '
if useGPU:
    label += 'GPU '
label += 'Adam batch {} epochs {:d} lr {:.6f} hids
label += ' ReLU ' if useRelu else ' Tanh '
label += time.strftime('%m/%d/17-%H:%M')

# Neural Network Model (1 hidden layer)
class Net(tnn.Module):
    def __init__(self, input_size, hidden_size, num
        self.hidden_size = hidden_size
        super(Net, self).__init__()
        self.fc1 = tnn.Linear(input_size, hidden_siz
        self.relu = tnn.ReLU() if useRelu else tnn.Ta
        if len(hidden_size) > 1:
            self.fc2 = tnn.Linear(hidden_size[0], hidd
            self.relu2 = tnn.ReLU() if useRelu else tnn
            self.fc3 = tnn.Linear(hidden_size[1], num
        else:
            self.fc3 = tnn.Linear(hidden_size[0], num
    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        if len(self.hidden_size) > 1:
            out = self.fc2(out)
            out = self.relu2(out)
```

```
        out = self.fc3(out)
        return out

train_dataset = dsets.MNIST(root='./data',
                            train=True,
                            transform=transforms.ToTensc
                            download=True)

test_dataset = dsets.MNIST(root='./data',
                           train=False,
                           transform=transforms.ToTenso

# Data Loader (Input Pipeline)
train_loader = torch.utils.data.DataLoader(data
                                           batch_size=batchSize,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(datas
                                          batch_size=batchSize,
                                          shuffle=False)

num_classes = 10
net = Net(784, hidden, num_classes)
if useGPU:
    net.cuda()

# Loss and Optimizer
criterion = tnn.CrossEntropyLoss()
optimizer = torch.optim.Adam(net.parameters(

global dataTrain, nSamples

if nSamples % batchSize != 0:
    print('WARNING: nSamples {} is not divisible
        nSampmles, batchSize))
nBatches = nSamples // batchSize
```

```
secsAcc = []
startTime = time.time()
for epoch in range(numEpochs):
    np.random.shuffle(dataTrain)
    for i, (images, labels) in enumerate(train_loac
        # Forward + Backward + Optimize
        if useGPU:
            images = Variable(images.view(-1, 28*28
            labels = Variable(labels).cuda()
        else:
            images = Variable(images.view(-1, 28*28
            labels = Variable(labels)
        # Forward + Backward + Optimize
        for iter in range(nIterations):
            optimizer.zero_grad()  # zero the gradier
            outputs = net(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

    correct = 0
    total = 0
    for images, labels in test_loader:
        if useGPU:
            images = Variable(images.view(-1, 28*28
        else:
            images = Variable(images.view(-1, 28*28
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        if useGPU:
            correct += (predicted.cpu() == labels).sur
        else:
            correct += (predicted == labels).sum()
    secsAcc.append([time.time() - startTime, (tota
```

```
    writeResults(resultsFilename, secsAcc, label)
    if numEpochs <= 10:
        writeResults('stdout', secsAcc, label)
```

In [ ]:

```
resultsFilename = 'test.results'
subprocess.call(['rm', resultsFilename])
hidden = [500, 500]  # can contain one or two ints
batchSize = 100
numEpochs = 50
```

In [ ]:

```
runpytorch(batchSize=batchSize, numEpochs=nu
        learningRate=0.001, nIterations=1, useRelu
```

```
torch.cuda.is_available() is True
```

In [ ]:

```
runpytorch(batchSize=batchSize, numEpochs=nu
        learningRate=0.001, nIterations=1, useRelu
```

In [ ]:

```
runpytorch(batchSize=batchSize, numEpochs=nu
        learningRate=0.001, nIterations=1, useRelu
```

In [ ]:

```
runpytorch(batchSize=batchSize, numEpochs=nu
      learningRate=0.001, nIterations=1, useRelu
```

In [ ]:

```
def plotFromFile(filename='test.results'):
    results = {}
    with open(filename,'r') as f:
        while True:
            label = f.readline()
            if label is None or label == '':
                break;
            n = int(f.readline())
            secsAcc = []
            for i in range(n):
                secsAcc.append([float(s) for s in f.readlir
            results[label] = secsAcc

    markers = ['s','8','>','^','<','v','o','X','P','d','h','*','p',
    mi = 0
    print(sorted(results))
    for key in sorted(results):
        value = results[key]
        value = np.array(value)
        plt.plot(value[:, 0], value[:, 1], '-',
                 marker=markers[mi], label=key, lw=4,
                 markersize=15)
        mi = (mi + 1) % len(markers)
    plt.xlabel('Seconds')
    plt.ylabel('Fraction of test samples incorrectly c
    plt.legend();
```

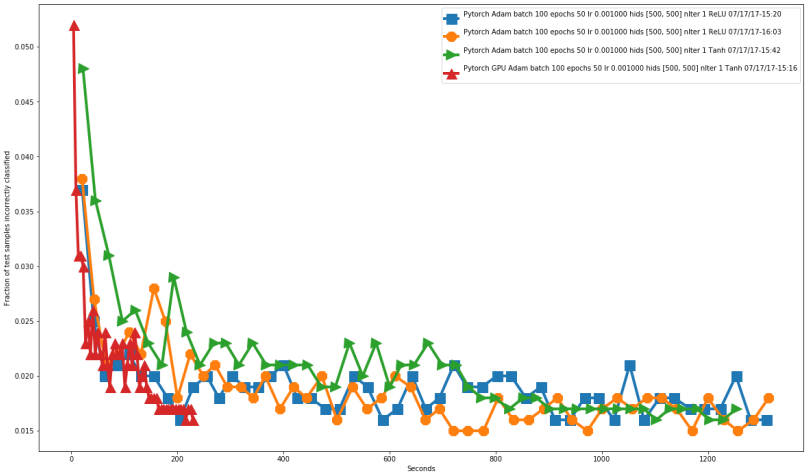
In [ ]:

```
# cat test.results
```

In [ ]:

```
plt.figure(figsize=(20, 12))
plotFromFile('test.results')
```

```
['Pytorch Adam batch 100 epochs 50 lr 0.001000 h
```



In [ ]:

```
runnumpy(batchSize=batchSize, numEpochs=nur
         useRelu=False, useAdam=True)
runnumpy(batchSize=batchSize, numEpochs=nur
         useRelu=False, useAdam=False)

runnumpy(batchSize=batchSize, numEpochs=nur
         useRelu=True, useAdam=True)
runnumpy(batchSize=batchSize, numEpochs=nur
         useRelu=True, useAdam=False)
```

In [ ]:

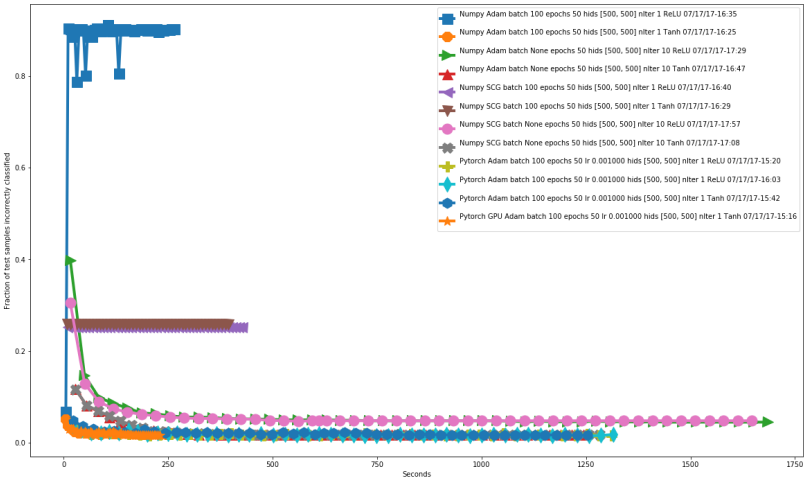
```
runnumpy(batchSize=None, numEpochs=50, hidd
      useRelu=False, useAdam=True)
runnumpy(batchSize=None, numEpochs=50, hidd
      useRelu=False, useAdam=False)

runnumpy(batchSize=None, numEpochs=50, hidd
      useRelu=True, useAdam=True)
runnumpy(batchSize=None, numEpochs=50, hidd
      useRelu=True, useAdam=False)
```

In [ ]:

```
plt.figure(figsize=(20, 12))
plotFromFile('test.results')
```

```
['Numpy Adam batch 100 epochs 50 hids [500, 50
```



In [ ]:

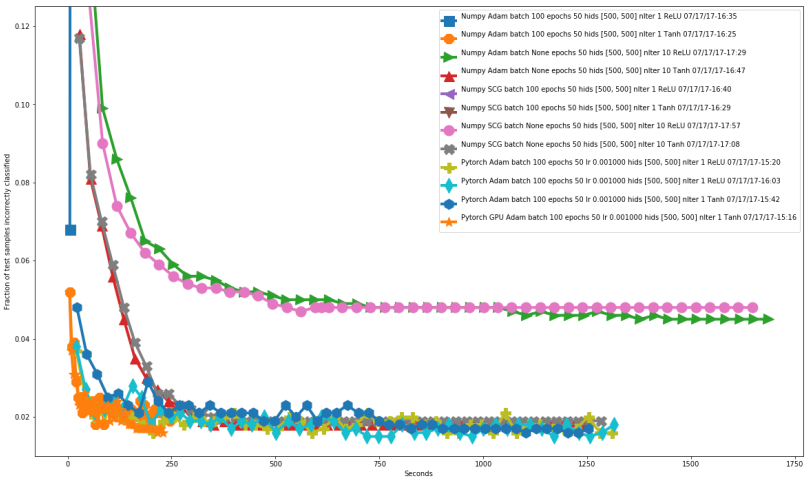
```
plt.figure(figsize=(20, 12))
```

```
plotFromFile('test.results')
plt.ylim(0.01,0.125)
# plt.xlim(0,40)
```

```
['Numpy Adam batch 100 epochs 50 hids [500, 50
```

Out[ ]:

```
(0.01, 0.125)
```



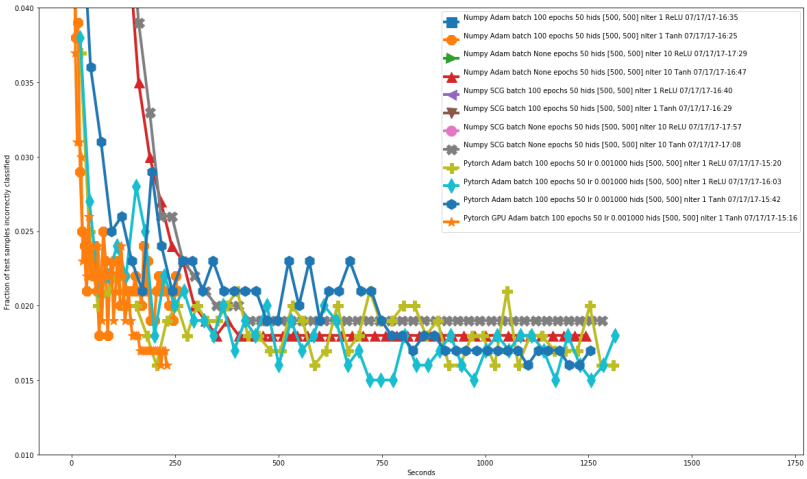
In [ ]:

```
plt.figure(figsize=(20, 12))
plotFromFile('test.results')
plt.ylim(0.01,0.04)
# plt.xlim(0,40)
```

```
['Numpy Adam batch 100 epochs 50 hids [500, 50
```

Out[ ]:

(0.01, 0.04)



In [ ]:

deep learning

deep learning, python, pytorch

‹ Fast Reinforcement Learning After Pretraining

› Comparing Numpy, Pytorch, and autograd on CPU and GPU

Leave a Comment

Name *

Email *

Website

Post Comment