

# 인공 신경망과 딥 러닝 실습

원 중 호

서울대학교 통계학과

Dec. 2015

# 목차

- 실습 목표
- 필요한 소프트웨어 및 라이브러리
- 참고 코드
- 실습

# 실습 목표

- Numpy, Theano 라이브러리 실습
- Keras 라이브러리를 활용한 deep convolutional network 구현

# 필요한 소프트웨어 및 라이브러리

- 리눅스, 맥 등에서 실습할 경우, 다음을 설치

① Python (2.7 버전)

② Numpy library / 설치 / Nonofficial Windows binaries 설치

- N차원 배열을 다루기 편리
- 선형대수, 푸리에 변환, 난수 생성 등의 유용한 기능을 제공

③ Scipy library / 설치 / Nonofficial Windows binaries 설치

- Numpy를 이용한 좀 더 고차원의 다양한 함수를 제공

④ Theano library / 설치

- Numpy와 잘 결합되어 있어, 다차원 배열을 포함한 수학적 표현이 쉬움
- GPU 사용이 편리
- 속도와 안정성에 대해 최적화

⑤ Keras library / 설치

- Theano 혹은 TensorFlow를 기반으로 하는 딥 뉴럴 네트워크 라이브러리
- 쉽고 직관적이며, 새로운 네트워크를 구성해 실험하기 좋은 환경을 제공
- Convolutional networks와 recurrent networks 및 다양한 병합 네트워크를 지원
- GPU 사용이 편리

# 필요한 소프트웨어 및 라이브러리 (Cont'd)

- 윈도우 환경에서 실습할 경우, 다음을 설치

## ① WinPython을 설치

- Python 기반의 완전한 기능을 갖춘 윈도우용 수치 계산 환경
- Keras, Theano, Numpy, Scipy 등의 라이브러리가 내장되어 있어 윈도우 환경에서 설치가 편리
- 반드시 2.7.x 버전을 설치

# 필요한 소프트웨어 및 라이브러리 (Cont'd)

- 윈도우 환경에서 실습할 경우, **WinPython 다운로드** 링크에서 운영체제(64bit/32bit)에 맞는 실행 파일 다운로드

Home / Browse / Development / Software Development / WinPython / Files

## WinPython

Portable Scientific Python 2/3 32/64bit Distribution for Windows  
Brought to you by: [praybaut](#), [stonebig](#)

Summary | **Files** | Reviews | Support | Wiki | Discussion | Mailing Lists | Tickets | Mercurial

Looking for the latest version? [Download WinPython-64bit-3.4.3.7.exe \(275.3 MB\)](#)

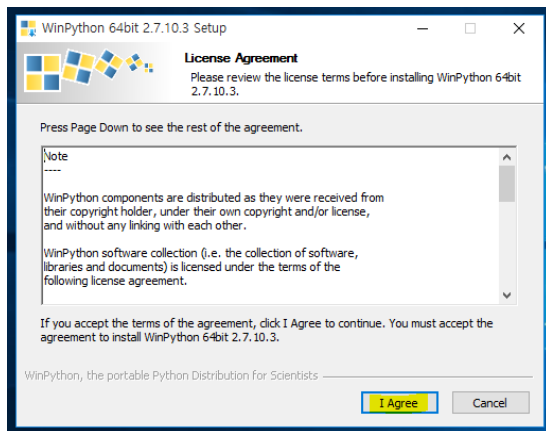
Home / WinPython\_2.7 / 2.7.10.3

Name ▾	Modified ▾	Size ▾	Downloads / Week ▾
↑ Parent folder			
betas	2015-10-26		1
WinPython-64bit-2.7.10.3.exe	2015-10-29	275.8 MB	861
WinPython-32bit-2.7.10.3.exe	2015-10-29	220.5 MB	568
Totals: 3 Items		496.3 MB	1,429

- 해당 링크에서 다운로드가 안 될 경우 **32bit 링크**/**64bit 링크**를 이용

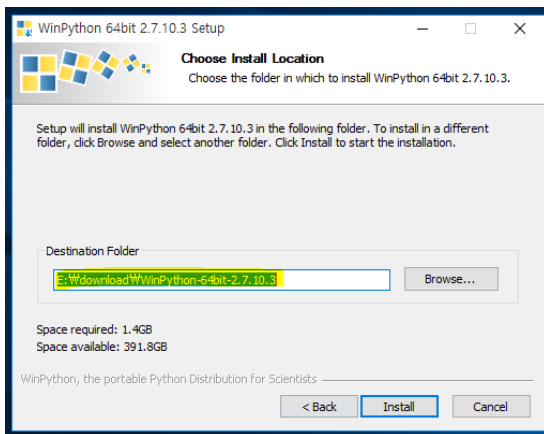
# 필요한 소프트웨어 및 라이브러리 (Cont'd)

- WinPython 설치



# 필요한 소프트웨어 및 라이브러리 (Cont'd)

- 설치 경로 지정

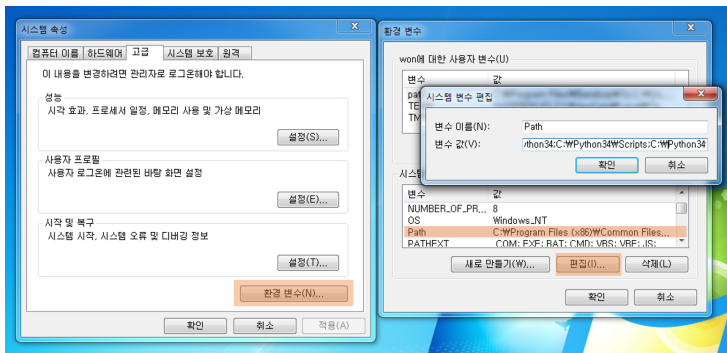




# 필요한 소프트웨어 및 라이브러리 (Cont'd)


















- 윈도우 환경 변수 설정

- ▶ 제어판 → 시스템 및 보안 → 시스템 → 고급 시스템 설정 → 환경 변수의 path에 WinPython 설치 경로 추가



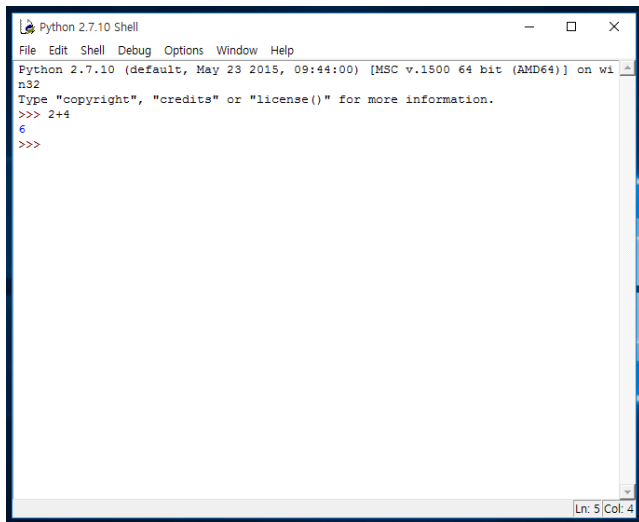
# WinPython 사용 방법

- IDLE(Integrated Development Environmnet : 통합 개발 환경)
  - ▶ Window, Mac, Unix 등의 다양한 환경에 적용할 수 있는 cross-platform 제공

 notebooks	2015-12-27 오후...	피
 python-2.7.10.amd64	2015-12-27 오후...	피
 scripts	2015-12-27 오후...	피
 settings	2015-12-27 오후...	피
 tools	2015-12-27 오후...	피
 IDLE (Python GUI)	2015-10-26 오후...	응
 IPython Qt Console	2015-10-26 오후...	응
 Jupyter Notebook	2015-10-26 오후...	응
 Qt Assistant	2015-10-26 오후...	응
 Qt Demo	2015-10-26 오후...	응
 Qt Designer	2015-10-26 오후...	응
 Qt Linguist	2015-10-26 오후...	응
 Spyder (light)	2015-10-26 오후...	응
 Spyder	2015-10-26 오후...	응
 WinPython Command Prompt	2015-10-26 오후...	응
 WinPython Control Panel	2015-10-26 오후...	응
 WinPython Interpreter	2015-10-26 오후...	응

# WinPython 사용 방법 (Cont'd)

- 인터프리터(Interactive interpreter)
  - ▶ 파이썬이 입력된 명령을 바로 번역해 실행



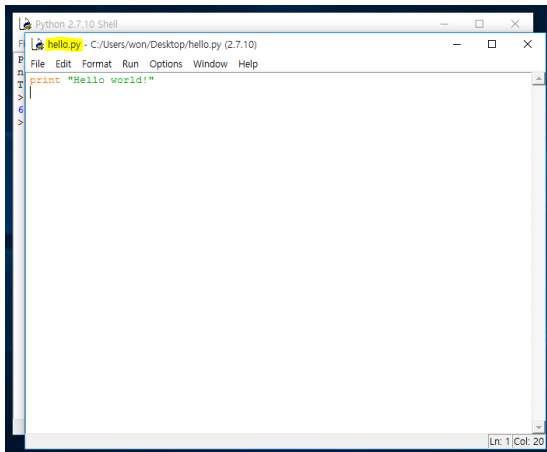
The screenshot shows a window titled "Python 2.7.10 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

```
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+4
6
>>>
```

The status bar at the bottom right indicates "Ln: 5 | Col: 4".

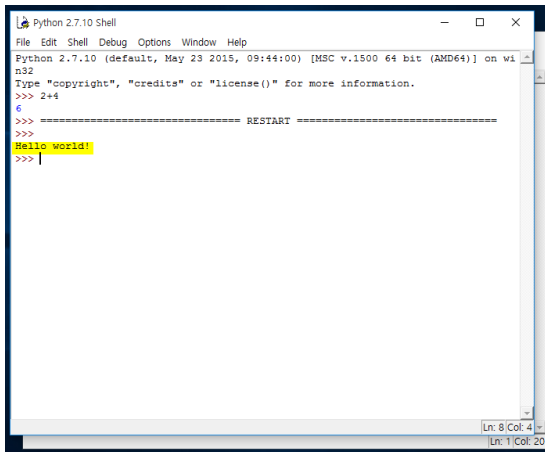
# WinPython 사용 방법 (Cont'd)

- 코드 작업을 위해 IDLE에서 소스 파일을 저장
  - ▶ File-New file(Ctrl+N)로 새 파일 창 열기
  - ▶ File-Save(Ctrl+S)로 저장



# WinPython 사용 방법 (Cont'd)

- IDLE에서 소스 파일 실행
  - ▶ Run-Run Module(F5)로 소스 파일 실행




















The screenshot shows a 'Python 2.7.10 Shell' window. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The status bar at the bottom indicates 'Ln: 8 Col: 4' and 'Ln: 1 Col: 20'. The command prompt shows the following sequence of commands and outputs:

```
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+4
6
>>> ===== RESTART =====
>>>
>>> Hello world!
>>> |
```

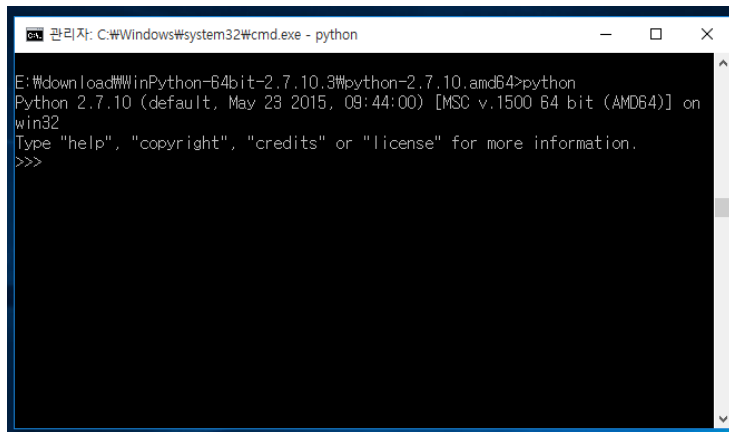
# WinPython 사용 방법 (Cont'd)

- WinPython Command Prompt
  - ▶ 명령 프롬프트에서 직접 python 실행

 notebooks	2015-12-24 오후...	파일 폴더
 python-2.7.10.amd64	2015-12-24 오후...	파일 폴더
 scripts	2015-12-24 오후...	파일 폴더
 settings	2015-12-24 오후...	파일 폴더
 tools	2015-12-24 오후...	파일 폴더
 IDLE (Python GUI)	2015-10-26 오후...	응용 프로그램
 IPython Qt Console	2015-10-26 오후...	응용 프로그램
 Jupyter Notebook	2015-10-26 오후...	응용 프로그램
 Qt Assistant	2015-10-26 오후...	응용 프로그램
 Qt Demo	2015-10-26 오후...	응용 프로그램
 Qt Designer	2015-10-26 오후...	응용 프로그램
 Qt Linguist	2015-10-26 오후...	응용 프로그램
 Spyder (light)	2015-10-26 오후...	응용 프로그램
 Spyder	2015-10-26 오후...	응용 프로그램
 WinPython Command Prompt	2015-10-26 오후...	응용 프로그램
 WinPython Control Panel	2015-10-26 오후...	응용 프로그램
 WinPython Interpreter	2015-10-26 오후...	응용 프로그램

# WinPython 사용 방법 (Cont'd)

- WinPython Command Prompt
  - ▶ 다음을 WinPython Command Prompt에 입력
    - > python
- WinPython Command Prompt내에서 python 실행 시



The screenshot shows a Windows Command Prompt window titled "관리자: C:\Windows\system32\cmd.exe - python". The command prompt is running the 'python' command, which has successfully launched the Python 2.7.10 interpreter. The output text is as follows:

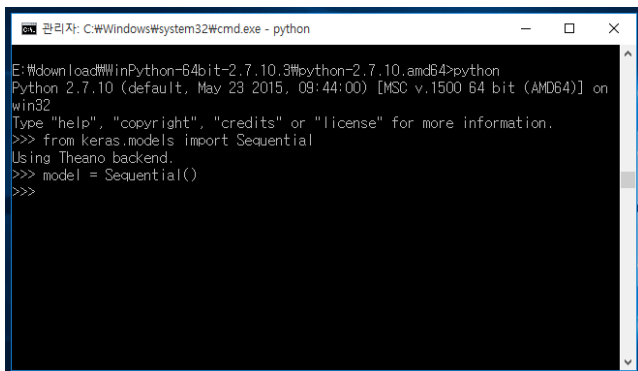
```
E:\download\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>python
Python 2.7.10 (default, May 23 2015, 08:44:00) [MSC v.1500 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## WinPython 사용 방법 (Cont'd)

- Winpython 및 Keras 설치가 잘 되었는지 확인하기 위해 python 내에서 다음을 입력

```
>>> from keras.models import Sequential  
>>> model = Sequential()
```

- 다음과 같이 뜨면 성공

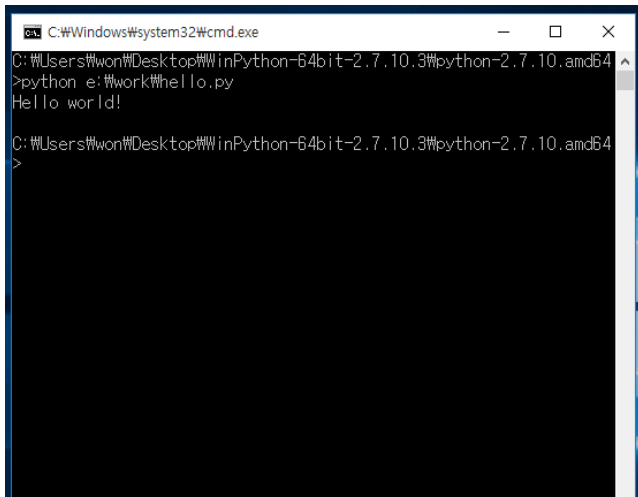


```
관리자: C:\Windows\system32\cmd.exe - python  
E:\download\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>python  
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)] on  
win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from keras.models import Sequential  
Using Theano backend.  
>>> model = Sequential()  
>>>
```



# WinPython 사용 방법 (Cont'd)

- WinPython Command Prompt
  - ▶ WinPython Command Prompt에서 직접 python 소스 파일 실행
- > python 파일 경로\파일명.py



```
C:\Windows\system32\cmd.exe
C:\Users\won\Desktop\WinPython-64bit-2.7.10.3\python-2.7.10.amd64
>python e:\work\hello.py
Hello world!


















C:\Users\won\Desktop\WinPython-64bit-2.7.10.3\python-2.7.10.amd64
>
```

# 간단한 numpy 소개

- 예제 : 다차원 배열 선언 및 형태 표현
  - ▶  $n$ 개 행과  $m$ 개 열을 갖는 행렬의 형태를  $(n,m)$ 으로 표현
- 유용한 함수
  - ▶ `a.reshape` : 배열의 형태를 변환
  - ▶ `a.random.rand` : 난수 생성
  - ▶ `a.zeros` : 0 값을 가지는 텐서 생성
  - ▶ `a.dot` : 내적 함수

# 간단한 numpy 소개 (Con't)

- 코드 작업을 위해 IDLE 실행

 notebooks	2015-12-24 오후...	파일 폴더
 python-2.7.10.amd64	2015-12-24 오후...	파일 폴더
 scripts	2015-12-24 오후...	파일 폴더
 settings	2015-12-24 오후...	파일 폴더
 tools	2015-12-24 오후...	파일 폴더
 IDLE (Python GUI)	2015-10-26 오후...	응용 프로그램
 IPython Qt Console	2015-10-26 오후...	응용 프로그램
 Jupyter Notebook	2015-10-26 오후...	응용 프로그램
 Qt Assistant	2015-10-26 오후...	응용 프로그램
 Qt Demo	2015-10-26 오후...	응용 프로그램
 Qt Designer	2015-10-26 오후...	응용 프로그램
 Qt Linguist	2015-10-26 오후...	응용 프로그램
 Spyder (light)	2015-10-26 오후...	응용 프로그램
 Spyder	2015-10-26 오후...	응용 프로그램
 WinPython Command Prompt	2015-10-26 오후...	응용 프로그램
 WinPython Control Panel	2015-10-26 오후...	응용 프로그램
 WinPython Interpreter	2015-10-26 오후...	응용 프로그램

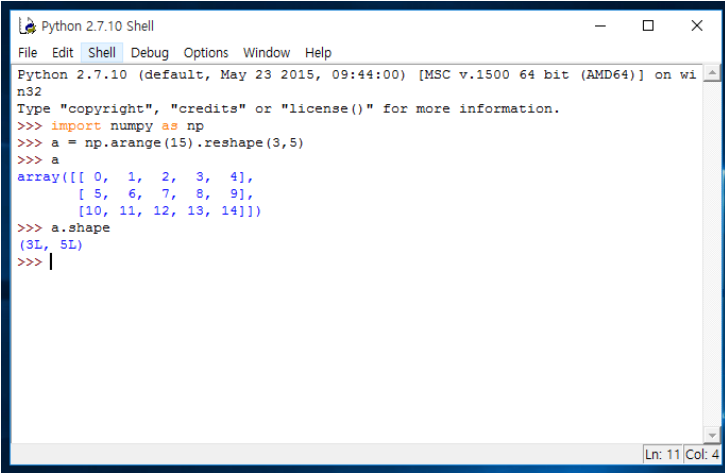
# 간단한 numpy 소개 (Con't)

- Numpy 예제 실습을 위해 IDLE 인터프리터에 다음을 입력해 실행

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
>>> a.shape
```

# 간단한 numpy 소개 (Con't)

- Numpy 예제 실습

A screenshot of a Python 2.7.10 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The title bar says 'Python 2.7.10 Shell'. The main text area shows the following code and output:

```
Python 2.7.10 (default, May 23 2015, 09:44:00) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import numpy as np
>>> a = np.arange(15).reshape(3,5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3L, 5L)
>>> |
```

The status bar at the bottom right indicates 'Ln: 11 Col: 4'.

# Theano 라이브러리

- 인터프리터와 컴파일러를 합침
  - ▶ 빠른 prototyping
  - ▶ 빠른 실행 속도 (C/CUDA 코드 생성기)
- Symbolic computation
  - ▶ 미분 자동화 (프로그래밍 감소)
- 역할 분담
  - ▶ 사람: 생성 공식을 주고, 조각들을 합침
  - ▶ Theano는 특정 추론/계산을 최적화
    - CPU/GPU 계산을 같은 코드로 할 수 있음
- 기존의 개발 환경 활용 가능
  - ▶ Python, NumPy, SciPy, Matplotlib

# Theano의 4 단계 사용 방법

- ① 입력 타입을 선언
- ② Symbolic 계산 : 수식 분석
- ③ 컴파일 : C/CUDA 코드를 만들고 컴파일
- ④ 계산 : 실제로 함수를 계산하고 모델을 훈련

# Compilation

- **표준화(canonicalization)**: symbolic 표현을 간단하게 함
- **수치 안정화(stabilization)**: 수치적 안정성을 향상시킴
  - ▶ 예.  $\log(1 + \exp(x))$
- **특수화(specialization)**: 더 빠른 계산 방법을 선택
  - ▶ 예.  $\text{pow}(x, 2) \rightarrow \text{sqr}(x)$ , 행렬 곱셈  $\rightarrow$  BLAS GEMM, 전치/부분 텐서  $\rightarrow$  상수 시간 메모리 에일리어싱
  - ▶ 만약 GPU가 사용되면, GPU에 적합한 것으로 선택
- **GPU 변환**: GPU 메모리로 float32형 데이터를 보냄
- **코드 생성**: 파이썬 모듈 동적 컴파일



# Theano 예제 : $Ax + b$

## ① Theano 변수를 정의

```
>>> import theano.tensor as T
>>> from theano import function
>>> A = T.matrix('A')
>>> x = T.vector('x')
>>> b = T.vector('b')
```

## ② $Ax + b$ 연산을 위한 symbolic 그래프 구성

```
>>> y = T.dot(A, x) + b
```

## ③ 컴파일

```
>>> f = function([A,x,b], y)
```

## ④ 계산

```
>>> f([[1,-1],[-1,1]],[2,3],[1,1])
array([0., 2.])
```

# Theano 예제 : Softmax classifier

# 1. Theano 입력 타입 선언

```
x = matrix()
y = vector(dtype='int32')
w = shared(numpy.random.randn(32,10))
b = shared(numpy.zeros(10))
```

# 2. Symbolic 계산

```
p_y = softmax(dot(x,w) + b)
loss = -log(p_y) [arange(y.shape[0]), y]
prediction = argmax(p_y)
gw , gb = grad(loss.mean(), [w,b])
```

# 3. 컴파일

```
train = function(inputs = [x,y],
                 outputs = [prediction, loss],
                 updates = {w:w-0.1 * gw, b:b-0.3 * gb})
```

# 4. 계산

```
for xdata, ydata in training_set:
    pred, err = train(xdata, ydata)
```

# Theano 예제 : Softmax classifier (Cont'd)

## ① Theano 입력 타입 선언

```
# 행렬 변수  $x$  선언
x = matrix();
# 정수형 벡터 변수  $y$  선언
y = vector(dtype='int32')
# numpy의 난수 생성 함수를 이용해 만든
#  $32 \times 10$  행렬 변수를 초기값으로 하는 shared 변수  $w$  선언
w = shared(numpy.random.randn(32,10))
# numpy의 0 벡터 생성 함수를 이용해 만든
# 크기 10의 벡터 변수를 초기값으로 하는 shared 변수  $b$  선언
b = shared(numpy.zeros(10))
```

- ▶ **theano.shared** : GPU의 shared memory에 저장되도록 변수를 선언

# Theano 예제 : Softmax classifier

## ② Symbolic 계산

```
#  $xw + b$  행렬의 각 원소에 softmax 함수를 취한  
# 행렬을 구해주는 그래프 p_y  
p_y = softmax(dot(x,w) + b)  
# p_y 그래프 결과의 각 원소에 log 함수를 취한  
# 결과 행렬의, 실제 레이블에 대응되는 값 벡터를  
# 구해주는 그래프 loss  
loss = -log(p_y) [arange(y.shape[0]), y]  
# p_y 그래프 결과의 최대값에 대응되는 레이블을  
# 구해주는 그래프  
prediction = argmax(p_y)  
# loss 그래프 결과의 평균값을 w, b 변수로 미분한  
# 기울기(gradient) 그래프 gw, gb  
gw , gb = grad(loss.mean(), [w,b])
```

# Theano 예제 : Softmax classifier

## 3 컴파일

```
# symbolic으로 구성한 그래프를 컴파일해 train 함수로 구성
# outputs : outputs 그래프에 input 값을 대입한 결과를 출력
# updates : updates 수식을 이용해 w, b 변수의 값을 업데이트
train = function(inputs = [x,y],
                 outputs = [prediction, loss],
                 updates = {w:w-0.1 * gw, b:b-0.3 * gb})
```

▶ `theano.function`을 이용

## 4 계산

```
# training_set 데이터 내의 원소 xdata, ydata를
# train 함수에 대입해 w, b를 학습하고,
# 이 때 prediction, loss를 각각 pred, err에 저장
for xdata, ydata in training_set:
    pred, err = train(xdata, ydata)
```

# 실습 - 딥 뉴럴 네트워크 코드

- 다운로드 / Github
- `mnist_shallow.py`
  - ▶ MNIST 데이터의 구분을 위한 얇은 네트워크(shallow network)를 구성해 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- `mnist_cnn_sigmoid_shallow.py`
  - ▶ CNN(Convolutional Neural Network) 네트워크 구조를 이용해 MNIST 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
  - ▶ 계산 유닛으로 sigmoid를 사용
- `mnist_cnn_sigmoid.py`
  - ▶ CNN(Convolutional Neural Network) 네트워크 구조를 더 쌓아 구성한 deep Convolutional Network를 이용해 MNIST 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
  - ▶ 계산 유닛으로 sigmoid를 사용

# 실습 - GPU 사용

- GPU 사용이 가능한 컴퓨터의 경우, Theano에서 제공하는 GPU 병렬화 기능을 사용하기 위해서는 Theano flag를 GPU 계산을 위해 셋팅해야 함.
    - ▶ 예를 들어 mnist\_shallow.py 프로그램을 GPU에서 실행시키려 할 경우, Command-line Prompt에 다음과 같이 입력해 Theano flag 셋팅 Theano flag를 GPU 계산을 위해 셋팅 후 코드를 실행
- ```
> THEANO_FLAGS=mode=FAST_RUN,device=gpu,floatX=float32
> python mnist_shallow.py
```

## ① mnist\_shallow.py : 얇은 네트워크 구조

- ▶ 12개의 epoch, mini-batch size = 128
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 mnist\_shallow.py로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import SGD
from keras.utils import np_utils
```



# 실습 (Cont'd)

## ① `mnist_shallow.py` : 얇은 네트워크 구조

- ▶ **MNIST database of handwritten digits** : MNIST 데이터 로드
- ▶ 데이터 변환 : 각각 784(28x28)개의 정수열(0~1)로 이루어진 60000개의 training set, 10000개의 test set 구성

```
batch_size = 128
nb_classes = 10
nb_epoch = 12
```

```
# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

# 실습 (Cont'd)

## ① `mnist_shallow.py` : 얇은 네트워크 구조

```
# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()
model.add(Dense(10, init='uniform', input_shape=(784,)))
model.add(Activation('softmax'))

model.compile(loss='mse', optimizer='sgd')

hist = model.fit(X_train, Y_train, batch_size=batch_size,
                 nb_epoch=nb_epoch, show_accuracy=True,
                 verbose=1, validation_split=0.2)

score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)

print('Test score:', score[0])
print('Test accuracy:', score[1])
```

# 실습 (Cont'd)

## ① `mnist_shallow.py` : 얇은 네트워크 구조

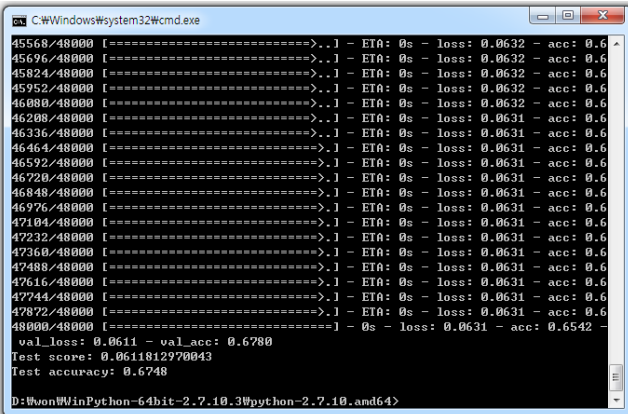
- ▶ 12개의 epoch, mini-batch size = 128
- ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행

> python 파일 경로\mnist\_shallow.py

# 실습 (Cont'd)

## ① mnist\_shallow.py : 얇은 네트워크 구조

- ▶ 12개의 epoch, mini-batch size = 128



```
C:\Windows\system32\cmd.exe
45568/48000 [=====>..] - ETA: 0s - loss: 0.0632 - acc: 0.6
45696/48000 [=====>..] - ETA: 0s - loss: 0.0632 - acc: 0.6
45824/48000 [=====>..] - ETA: 0s - loss: 0.0632 - acc: 0.6
45952/48000 [=====>..] - ETA: 0s - loss: 0.0632 - acc: 0.6
46080/48000 [=====>..] - ETA: 0s - loss: 0.0632 - acc: 0.6
46208/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46336/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46464/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46592/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46720/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46848/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
46976/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47104/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47232/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47360/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47488/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47616/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47744/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
47872/48000 [=====>..] - ETA: 0s - loss: 0.0631 - acc: 0.6
48000/48000 [=====] - 0s - loss: 0.0631 - acc: 0.6542 -
val_loss: 0.0611 - val_acc: 0.6780
Test score: 0.0611812970043
Test accuracy: 0.6748
D:\won\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>
```

## 실습 (Cont'd)

### ② mnist\_shallow2.py : 얇은 네트워크 구조 2

- ▶ 30개의 epoch, mini-batch size = 128
- ▶ 하나의 은닉층, 100개의 hidden neuron을 포함
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 mnist\_shallow2.py로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import SGD
from keras.utils import np_utils
```

## 실습 (Cont'd)

### ② `mnist_shallow2.py` : 얇은 네트워크 구조 2

```
batch_size = 128
nb_classes = 10
nb_epoch = 30

# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

## 실습 (Cont'd)

### ② `mnist_shallow2.py` : 얇은 네트워크 구조 2

```
# convert class vectors to binary class matrices
```

```
Y_train = np_utils.to_categorical(y_train, nb_classes)
```

```
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

```
model = Sequential()
```

```
model.add(Dense(100, init='uniform', input_shape=(784,)))
```

```
model.add(Activation('sigmoid'))
```

```
model.add(Dense(10, init='uniform', input_shape=(100,)))
```

```
model.add(Activation('softmax'))
```

```
model.compile(loss='mse', optimizer='sgd')
```

```
hist = model.fit(X_train, Y_train, batch_size=batch_size,  
                 nb_epoch=nb_epoch, show_accuracy=True,  
                 verbose=1, validation_split=0.2)
```

```
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
```

```
print('Test score:', score[0])
```

```
print('Test accuracy:', score[1])
```

# 실습 (Cont'd)

## ② `mnist_shallow2.py` : 얇은 네트워크 구조 2

- ▶ 30개의 epoch, mini-batch size = 128
- ▶ 하나의 은닉층, 100개의 hidden neuron을 포함
- ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행

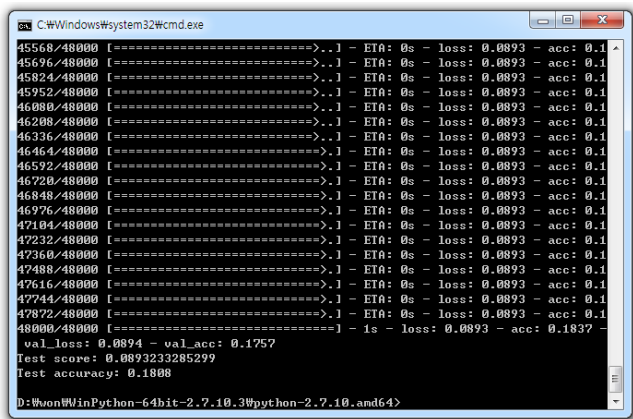
> python 파일 경로\mnist\_shallow2.py



# 실습 (Cont'd)

## ② mnist\_shallow2.py : 얇은 네트워크 구조 2

- ▶ 30개의 epoch, mini-batch size = 128
- ▶ 하나의 은닉층, 100개의 hidden neuron을 포함



```
C:\Windows\system32\cmd.exe
45568/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
45696/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
45824/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
45952/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46080/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46208/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46336/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46464/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46592/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46720/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46848/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
46976/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47104/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47232/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47360/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47488/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47616/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47744/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
47872/48000 [=====>..] - ETA: 0s - loss: 0.0893 - acc: 0.1
48000/48000 [=====] - 1s - loss: 0.0893 - acc: 0.1837 -
val_loss: 0.0894 - val_acc: 0.1757
Test score: 0.0893233285299
Test accuracy: 0.1808
D:\won\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>
```

## 실습 (Cont'd)

### ③ `mnist_cnn_shallow.py` : Convolutional network

- ▶ 12개의 epoch, mini-batch size = 128
- ▶ Convolutional layer : 3개의 필터,  $[3 \times 3]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 `mnist_cnn_shallow.py`로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

## 실습 (Cont'd)

### ③ mnist\_cnn\_shallow.py : Convolutional network

```
batch_size = 128
nb_classes = 10
nb_epoch = 12

# input image dimensions
img_rows, img_cols = 28, 28
# number of convolutional filters to use
nb_filters = 3
# convolution kernel size
nb_conv = 3
# size of pooling area for max pooling
nb_pool = 2

# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
```

## 실습 (Cont'd)

### ③ mnist\_cnn\_shallow.py : Convolutional network

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

print('X_train shape:', X_train.shape)
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()
```

## 실습 (Cont'd)

### ③ mnist\_cnn\_shallow.py : Convolutional network

```
model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
                        border_mode='valid',
                        input_shape=(1, img_rows, img_cols)))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adadelta')
```

## 실습 (Cont'd)

### ③ mnist\_cnn\_shallow.py : Convolutional network

```
hist = model.fit(X_train, Y_train, batch_size=batch_size,
                 nb_epoch=nb_epoch, show_accuracy=True,
                 verbose=1, validation_split=0.2)
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

## 실습 (Cont'd)

### ③ `mnist_cnn_shallow.py` : Convolutional network

- ▶ 12개의 epoch, mini-batch size = 128
- ▶ Convolutional layer : 3개의 필터,  $[3 \times 3]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool
- ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행

> python 파일 경로\mnist\_cnn\_shallow.py

# 실습 (Cont'd)

## ③ mnist\_cnn\_shallow.py : Convolutional network

- ▶ 12개의 epoch, mini-batch size = 128
- ▶ Convolutional layer : 3개의 필터,  $[3 \times 3]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool

```
C:\Windows\system32\cmd.exe
45568/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
45696/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
45824/48000 [=====>..] - ETA: 0s - loss: 0.2575 - acc: 0.9
45952/48000 [=====>..] - ETA: 0s - loss: 0.2576 - acc: 0.9
46080/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
46208/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
46336/48000 [=====>..] - ETA: 0s - loss: 0.2584 - acc: 0.9
46464/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46592/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46720/48000 [=====>..] - ETA: 0s - loss: 0.2582 - acc: 0.9
46848/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46976/48000 [=====>..] - ETA: 0s - loss: 0.2584 - acc: 0.9
47104/48000 [=====>..] - ETA: 0s - loss: 0.2588 - acc: 0.9
47232/48000 [=====>..] - ETA: 0s - loss: 0.2588 - acc: 0.9
47360/48000 [=====>..] - ETA: 0s - loss: 0.2589 - acc: 0.9
47488/48000 [=====>..] - ETA: 0s - loss: 0.2587 - acc: 0.9
47616/48000 [=====>..] - ETA: 0s - loss: 0.2587 - acc: 0.9
47744/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
47872/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
48000/48000 [=====] - 13s - loss: 0.2585 - acc: 0.9199
- val_loss: 0.1398 - val_acc: 0.9597
Test score: 0.134794600892
Test accuracy: 0.9584
D:\won\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>
```



# 실습 (Cont'd)

## ④ `mnist_cnn.py` : Convolutional network 2

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ 2개의 Convolutional layer : 3개의 필터,  $[5 \times 5]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 `mnist_cnn.py`로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

## 실습 (Cont'd)

### ④ `mnist_cnn.py` : Convolutional network 2

```
batch_size = 128
nb_classes = 10
nb_epoch = 20

# input image dimensions
img_rows, img_cols = 28, 28
# number of convolutional filters to use
nb_filters = 3
# convolution kernel size
nb_conv = 5
# size of pooling area for max pooling
nb_pool = 2

# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
```

## 실습 (Cont'd)

### ④ mnist\_cnn.py : Convolutional network 2

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print('X_train shape:', X_train.shape)
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()
```

## 실습 (Cont'd)

### ④ `mnist_cnn.py` : Convolutional network 2

```
model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
                        border_mode='valid',
                        input_shape=(1, img_rows, img_cols)))
model.add(Activation('sigmoid'))
model.add(Convolution2D(nb_filters, nb_conv, nb_conv))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adadelat')
```

# 실습 (Cont'd)

## ④ mnist\_cnn.py : Convolutional network 2

```
hist = model.fit(X_train, Y_train, batch_size=batch_size,
                 nb_epoch=nb_epoch, show_accuracy=True,
                 verbose=1, validation_split=0.2)
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

## 실습 (Cont'd)

### ④ `mnist_cnn.py` : Convolutional network 2

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ 2개의 Convolutional layer : 3개의 필터,  $[5 \times 5]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool
- ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행

> python 파일 경로\mnist\_cnn.py

# 실습 (Cont'd)

## ④ mnist\_cnn.py : Convolutional network 2

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ 2개의 Convolutional layer : 3개의 필터,  $[5 \times 5]$  국소 수용
- ▶ Maxpooling layer :  $[2 \times 2]$  영역에서 pool

```
C:\Windows\system32\cmd.exe
45568/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
45696/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
45824/48000 [=====>..] - ETA: 0s - loss: 0.2575 - acc: 0.9
45952/48000 [=====>..] - ETA: 0s - loss: 0.2576 - acc: 0.9
46080/48000 [=====>..] - ETA: 0s - loss: 0.2578 - acc: 0.9
46208/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
46336/48000 [=====>..] - ETA: 0s - loss: 0.2584 - acc: 0.9
46464/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46592/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46720/48000 [=====>..] - ETA: 0s - loss: 0.2582 - acc: 0.9
46848/48000 [=====>..] - ETA: 0s - loss: 0.2581 - acc: 0.9
46976/48000 [=====>..] - ETA: 0s - loss: 0.2584 - acc: 0.9
47104/48000 [=====>..] - ETA: 0s - loss: 0.2588 - acc: 0.9
47232/48000 [=====>..] - ETA: 0s - loss: 0.2588 - acc: 0.9
47360/48000 [=====>..] - ETA: 0s - loss: 0.2589 - acc: 0.9
47488/48000 [=====>..] - ETA: 0s - loss: 0.2587 - acc: 0.9
47616/48000 [=====>..] - ETA: 0s - loss: 0.2587 - acc: 0.9
47744/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
47872/48000 [=====>..] - ETA: 0s - loss: 0.2585 - acc: 0.9
48000/48000 [=====] - 13s - loss: 0.2585 - acc: 0.9199
- val_loss: 0.1398 - val_acc: 0.9597
Test score: 0.134794600892
Test accuracy: 0.9584
D:\won\WinPython-64bit-2.7.10.3\python-2.7.10.amd64>
```