Using Multi-layer Neural Network to recognize Hand-Writing Digit

장정문 2014920049

Why I choose this one to achieve?

I want to know how the Neural Network actually works, and how the Neural Network can be applied. After finished this project, I realized a lot, something that I even never considered. The first one, weather you can get the good training data determines your success or failure. A proper initial weight is vital. Choosing a good learning rate can accelerate the learning process, it can also improve the learning quality.

What's in the project folder?

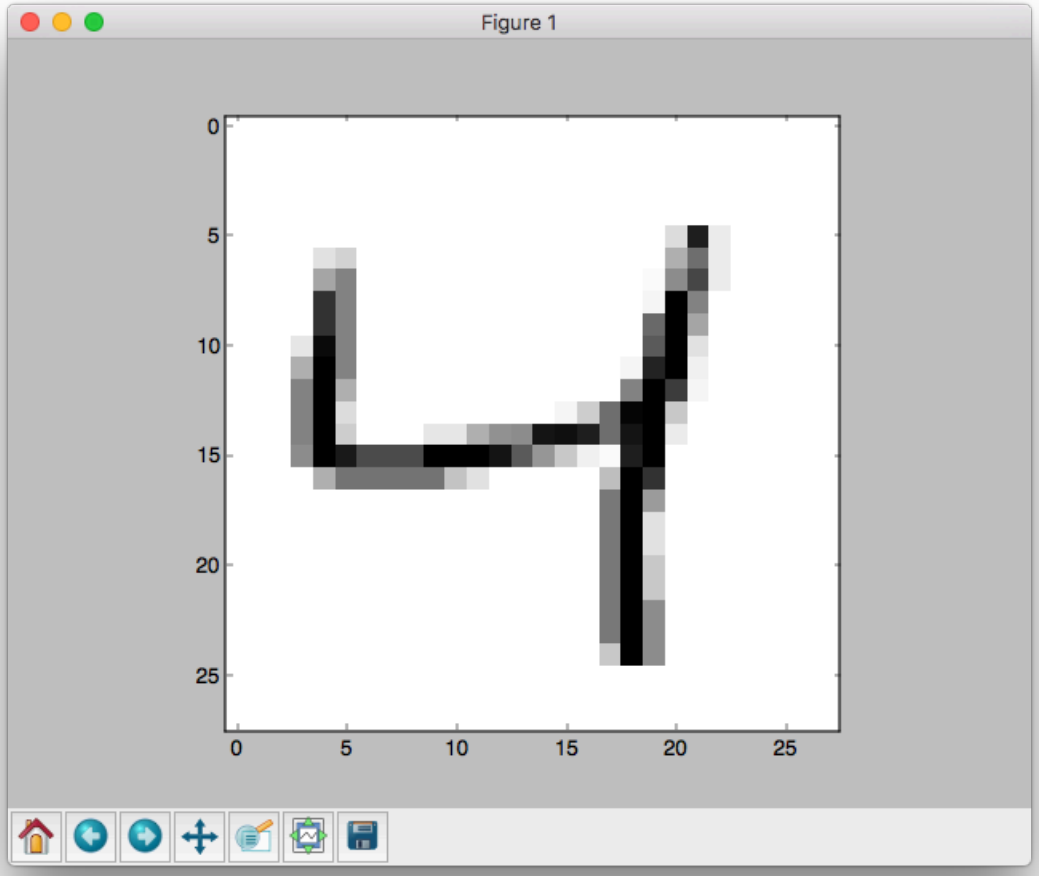
num_r.c -> Source File
num_r -> Executable Binary(x86)
show.py -> A tiny python script for showing the image in the data set
test.csv -> Data set for testing the Neural Network
train.csv -> Data set for training the Neural Network
Good_Weight -> Every time after the num_r execution, it stores its trained weight to Good_Weight.
gsl-2.2.tar.gz -> GNU Scientific Library

What's in the train.csv and test.csv?

Each line in the train.csv and test.csv is a 28*28 bmp file in the form of pixel's RGB value. The first number at the start of the line is the label, It tells what's the actual number of the bmp file.

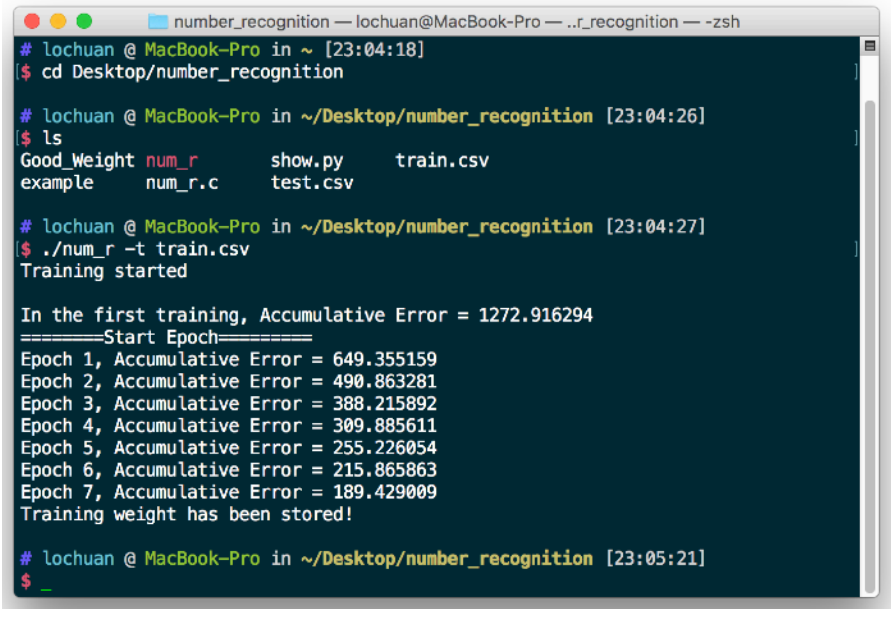
```
7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,84,185,159,151,60,36,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,222,254,254,254,254,241,198,198,198,198,198,198,198,198,170,52,0,0,0,0,0,0,0,0,0,0,0,0,
67,114,72,114,163,227,254,225,254,254,254,250,229,254,254,140,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,17,66,14,67,67,67,59,21,236,254,106,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,83,253
,209,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,233,255,83,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,129,254,238,44,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,59,249,25
4,62,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,254,187,5,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,9,205,248,58,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,126,254,182,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,75,251,240,57,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,19,221,254,166,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,203,254,219,35,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38,254,254,77,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,31,224,254,115,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,254,254,52,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,61,242,254,254,52,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,121,254,254,219,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,254,207,1
8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

In the picture above, the first number 7 is the label, the left are bmp file information. If we write a small python script to process the data, we can get the output below



How to use this program?

In the command line environment, ./num_c [-t, -r] [data_set], the -t stands for training, the -r stands for recognize. If you attach with x86, you may directly run the binary. If it doesn't, please compile the source code num_r.c



How to compile the source?

The source file depends on GNU Scientific Library.

1: Untar the gsl-2.2-tar.gz to somewhere
2. -I specifies the Include Path, and -L specifies the library path

$ gcc -I/somewhere/gsl/2.2.1/include num_r.c -L/somewhere/gsl/2.2.1/lib -lgsl -lcblas -lm

If the gsl has been installed.

$ gcc num_r.c -o num_r -lgsl -lcblas -lm

Some details:

train.csv contains 60000 images.
test.csv contains 10000 images.

The hidden layer has 250 nodes.
The default learning rate is 0.1
The default input size of training data is 5000
The default input size of testing data is 1000
The default EPOCH is 8

Before recognizing, please train the neural network first.
./num_r -t train.csv -> ./num_r -r test.csv