

**DAMH-01**

**NHẬN DẠNG KÝ SỐ VIẾT TAY VỚI MÔ HÌNH  
MẠNG NEURAL NHÂN TẠO**

**BÁO CÁO BÀI TẬP MÔN HỌC  
CÁC HỆ CƠ SỞ TRI THỨC**

**Người thực hiện**

Huỳnh Gia Bảo - 1412036

Võ Phương Hòa - 1412192

**Giáo viên phụ trách**

Lê Hoàng Thái

Nguyễn Hoàng Khai

## **Mục lục**

I. Đôi nét về mạng neural nhân tạo .....	1
1. Mạng neural nhân tạo .....	1
2. Mạng truyền thẳng đa lớp .....	1
II. Chương trình nhận dạng ký số .....	2
1. Mô tả chương trình .....	2
1.1. Tiền xử lý dữ liệu.....	2
1.2. Chương trình nhận dạng .....	3
1.3. Chương trình thực thi .....	4
2. Các hàm chính .....	4
2.1. Tiền xử lý dữ liệu.....	4
2.2. Chương trình nhận dạng .....	5
3. Một số kết quả thu được .....	6
3.1. Nhận diện ký số qua một ảnh .....	6
3.2. Tính hiệu suất của chương trình huấn luyện .....	6
III. Tài liệu tham khảo.....	7
IV. Nội dung phân công.....	8

## I. Đôi nét về mạng neural nhân tạo

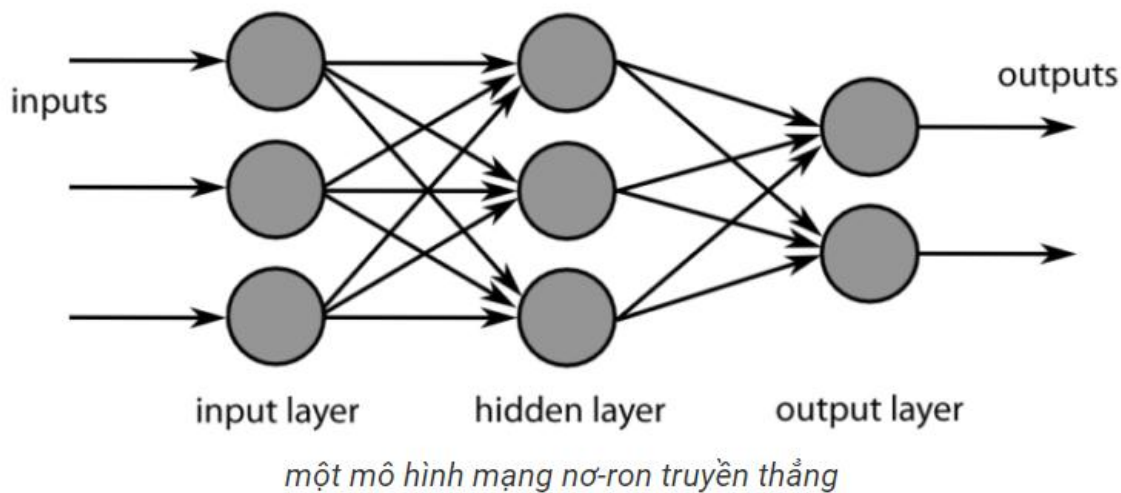
### 1. Mạng neural nhân tạo

Mạng nơ-ron nhân tạo hay thường gọi ngắn gọn là mạng nơ-ron là một mô hình toán học hay mô hình tính toán được xây dựng dựa trên các mạng nơ-ron sinh học. Nó gồm có một nhóm các nơ-ron nhân tạo (nút) nối với nhau, và xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút (cách tiếp cận connectionism đối với tính toán). Trong nhiều trường hợp, mạng nơ-ron nhân tạo là một hệ thống thích ứng (adaptive system) tự thay đổi cấu trúc của mình dựa trên các thông tin bên ngoài hay bên trong chảy qua mạng trong quá trình học.

Trong thực tế sử dụng, nhiều mạng nơ-ron là các công cụ mô hình hóa dữ liệu thống kê phi tuyến. Chúng có thể được dùng để mô hình hóa các mối quan hệ phức tạp giữa dữ liệu vào và kết quả hoặc để tìm kiếm các dạng/mẫu trong dữ liệu.

### 2. Mạng truyền thẳng đa lớp

Mô hình đơn giản nhất của mạng nơ-ron chính là mạng nơ-ron truyền thẳng. Các nơ-ron trong mạng truyền thẳng liên kết với nhau mà không hình thành chu trình nên tín hiệu sẽ truyền thẳng từ đầu vào qua các lớp ẩn và đến đầu ra.



Mạng nơ-ron truyền thẳng đa lớp có nhiều ứng dụng trong thực tế như trong các bài toán nhận dạng hình ảnh, nhận dạng tiếng nói, xử lý ngôn ngữ tự nhiên, ....

Trong báo cáo này, chúng tôi sử dụng mạng nơ-ron truyền thẳng đa lớp cho bài toán nhận dạng chữ số viết tay.

## II. Chương trình nhận dạng ký số

Ngôn ngữ: Python

Version: 2.7

Cấu trúc bài nộp: chứa các thư mục sau

+ Report: 1412036\_1412192.pdf

+ Source:

- mnist.py, neural\_nets.py, test.py
- data (chứa dữ liệu mnist)
- image (chứa một số ảnh (\*.bmp) lấy từ mnist)

Dữ liệu: MNIST (<http://yann.lecun.com/exdb/mnist/>)

### 1. Mô tả chương trình

#### 1.1. Tiền xử lý dữ liệu

Quá trình này áp dụng với dữ liệu đầu vào (MNIST).

Dữ liệu mnist là tập các database gồm train\_images, train\_labels, test\_images và test\_labels. Trong đó, dữ liệu huấn luyện gồm 60000 mẫu đã được gán nhãn đúng, dữ liệu test gồm 10000 mẫu với nhãn là dữ liệu dùng để kiểm tra hiệu suất của chương trình nhận dạng.

Cấu trúc các tập tin :

- **train\_images.idx3-ubyte** và **test\_images.idx3-ubyte** :  
4 bytes đầu tiên : magic number  
4 bytes thứ hai : số lượng images  
4 bytes thứ ba: số dòng của images  
4 bytes thứ tư: số cột của images  
các bytes còn lại, mỗi bytes mô tả một pixel.
- **train\_labels.idx1-ubyte** và **test\_labels.idx1-ubyte**:  
4 bytes đầu tiên: magic number  
4 bytes thứ hai: số lượng images  
các bytes còn lại, mỗi bytes mô tả một output (nhãn đúng của ảnh)

Tóm lại, ta có 60000 mẫu huấn luyện, kích thước 28\*28 và 10000 mẫu test, cùng kích thước 28\*28.

Chương trình tiền xử lý đọc dữ liệu trong các tệp tin trên dưới dạng nhị phân, loại bỏ các bytes không cần thiết, gom cụm mỗi 28\*28 pixel thành một ảnh (với data) và gom các nhãn thành một mảng (với label).

Bản thân chương trình được thiết kế như một thư viện để phục vụ cho chương trình thực thi. Các hàm chính sẽ được mô tả cụ thể trong phần sau.

## 1.2. Chương trình nhận dạng

Chương trình mô tả class Neural\_nets gồm 4 thuộc tính và 6 phương thức đóng vai trò huấn luyện mô hình học neural và 2 hàm khác dùng để tính sigmoid và đạo hàm của sigmoid.

4 thuộc tính:

- layers: mô tả số lớp của network
- sz: mô tả số neural của từng lớp, có dạng [a, b, c... , n] với a, b, c, n lần lượt là số neural của lớp input, lớp ẩn thứ nhất, lớp ẩn thứ hai và lớp output.
- biases: ma trận các bias
- weights: ma trận trọng số

6 phương thức:

- \_\_init\_\_(): khởi tạo lớp Neural\_nets .
- training\_model(): hàm huấn luyện ra một mô hình để nhận dạng ký số.
- mini\_training\_model(): hỗ trợ cho hàm training\_model
- learning\_model(): học từ mô hình thu được ở hàm training\_model
- result(): trả về kết quả là số lượng mẫu test được phân nhãn đúng
- backpropagation(): là hàm quan trọng nhất, mô phỏng thuật toán lan truyền ngược, huấn luyện trực tiếp dữ liệu mẫu (x, y) (có thể coi x là image, y là nhãn của image đó).

Cũng như chương trình tiền xử lý, chương trình này được thiết kế như một thư viện để phục vụ chương trình thực thi.

### 1.3. Chương trình thực thi

Chương trình chạy dưới dạng tham số dòng lệnh

Cú pháp:

**<đường dẫn>\python <tên chương trình> <type> <list> <loop\_number>  
<learning\_rate> <image\_name>**

trong đó:

- tên chương trình mặc định là **test.py**
- **<type>**: là 1 hoặc 2
  - 1: đọc vào 1 ảnh và cho biết nhãn của ảnh đó.
  - 2: cho biết hiệu suất của chương trình huấn luyện trên dữ liệu test.
- **<list>**: có dạng [a,b,c, ...z], trong đó số phần tử thể hiện số layers của nets, còn a, b, c... cho biết số neural của mỗi layers.  
Trong đó, a mặc định là 784 (= 28\*28) nếu type là 2, còn nếu type là 1 thì a là số pixel của ảnh cần xác định nhãn, còn z mặc định là 10.  
Giữa các phần tử ngăn cách nhau bởi dấu ',' (không có khoảng trắng).
- **<loop\_number>**: số lần huấn luyện cho tập huấn luyện, loop\_number càng lớn thì hiệu suất huấn luyện càng cao, nhưng tốc độ càng chậm dần.
- **<learning\_rate>**: là số thực, nên thuộc đoạn [1.0, 10.0], nếu quá nhỏ hoặc quá lớn sẽ ảnh hưởng đến hiệu suất huấn luyện.
- **<image\_name>**: chỉ áp dụng nếu type là 1, nhập vào tên của ảnh cần xác định nhãn.

## 2. Các hàm chính

### 2.1. Tiền xử lý dữ liệu

**load\_data**(data\_name, label\_name, \_type, size, sign): load dữ liệu từ các file 'data\_name' chứa mẫu (image) và 'label\_name' chứa nhãn của mẫu.

Các tham số đầu vào:

data\_name, label\_name: tên file dữ liệu

\_type: mode, ở đây là 'rb'

size : 60000 nếu là dữ liệu huấn luyện, 10000 nếu là dữ liệu test

sign : 1 nếu là dữ liệu huấn luyện, 0 nếu là dữ liệu test

Kết quả trả về là 'load\_data' là một list với các phần tử kiểu tuple (x, y). Trong đó, x là vector biểu diễn image với 784 chiều (784 phần tử), các phần tử đã được chuẩn hóa min-max về đoạn [0,1], còn y là output đúng của x (nếu x là dữ liệu huấn luyện, y là vector 10 chiều với kiểu dữ liệu là boolean, vd : nếu x là image mô tả số 6, thì  $y = [0,0,0,0,0,0,1,0,0,0]$  (tức  $y[7] = 1$ ), nếu x là dữ liệu test thì y là số nguyên biểu diễn ký số mô tả dữ liệu test, vd : nếu x mô tả số 6, thì  $y = 6$ ).

## 2.2. Chương trình nhận dạng

**training\_model**(self, training\_data, epoch, lrate, test\_data = None): huấn luyện mô hình học cho dữ liệu.

'self' cho biết đây là 1 phương thức trong class Neural\_nets.

Các tham số đầu vào:

training\_data: tập các mẫu huấn luyện, mỗi mẫu là một tuple (x, y) như mô tả ở hàm load\_data

epoch: số lần huấn luyện tập training\_data

lrate: learning rate

test\_data (=None): mặc định là None, tức không bắt buộc phải có tham số này lúc gọi hàm, ở đây test\_data được gọi khi muốn biết hiệu suất của chương trình huấn luyện trên dữ liệu test.

Hàm này không có kết quả trả về, hay đúng hơn, những thay đổi của hàm này được lưu lại trong các thuộc tính của class Neural\_nets.

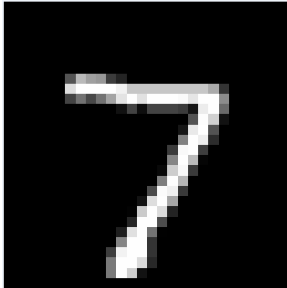
**backpropagation**(self, x, y): hàm mô phỏng thuật toán lan truyền ngược, nó trực tiếp huấn luyện trên mẫu dữ liệu (x, y)

Các tham số đầu vào: x, y có định dạng như mô tả ở hàm load\_data

Kết quả trả về là các vector init\_b, init\_w (có định dạng như các thuộc tính biases và weights của class Neural\_nets), các giá trị trong hai vector này thể hiện sự thay đổi sẽ được cập nhật với biases và weights.

### 3. Một số kết quả thu được

#### 3.1. Nhận diện ký số qua một ảnh



Hình bên mô phỏng số 7, size là 28\*28, tên ảnh là 'img1.bmp', ta gọi thực thi chương trình như sau:

```
..\python test.py 1 [784,30,10] 5 3.0 image/img1.bmp
```

Kết quả là:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Phuong Hoa>cd desktop\1412036_1412192\source
C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>python 1 test.py [784,30,10]
5 3.0 image/img1.bmp
python: can't open file '1': [Errno 2] No such file or directory

C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>python test.py 1 [784,30,10]
5 3.0 image/img1.bmp
loading...
training...
C:\Users\Phuong Hoa\Desktop\1412036_1412192\source\neural_nets.py:171: RuntimeWarning: overflow encountered in exp
  sigm = 1.0/(1.0+np.exp(-val))
The number is 7.

C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>
```

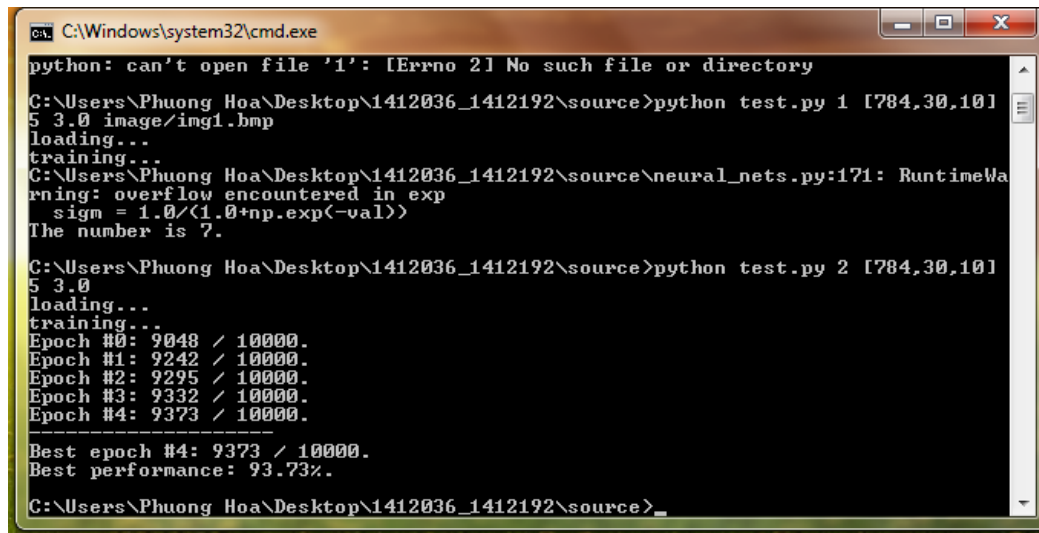
#### 3.2. Tính hiệu suất của chương trình huấn luyện

Như đã nói ở trên, chương trình sử dụng 60000 mẫu huấn luyện, 10000 mẫu test, chương trình huấn luyện mô hình thông qua các ma trận (w, b) (w là weights, còn b là biases). Mục đích của việc huấn luyện là tìm được ma trận w và b phù hợp nhất để tìm ra được kết quả là ký số tương ứng với mẫu.



Dưới đây là một mô tả thể hiện hiệu suất của chương trình.

..\python test.py 2 [784,30,10] 5 3.0



```
C:\Windows\system32\cmd.exe
python: can't open file '1': [Errno 2] No such file or directory

C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>python test.py 1 [784,30,10]
5 3.0 image/img1.bmp
loading...
training...
C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>python test.py 1 [784,30,10]
5 3.0 image/img1.bmp
loading...
training...
C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>python test.py 2 [784,30,10]
5 3.0
loading...
training...
Epoch #0: 9048 / 10000.
Epoch #1: 9242 / 10000.
Epoch #2: 9295 / 10000.
Epoch #3: 9332 / 10000.
Epoch #4: 9373 / 10000.
-----
Best epoch #4: 9373 / 10000.
Best performance: 93.73%.
C:\Users\Phuong Hoa\Desktop\1412036_1412192\source>
```

Trong các lệnh thực thi ở trên, như đã nói tham số liền kề ‘test.py’ là 1 hoặc 2 tùy thuộc vào yêu cầu người dùng, [784,30,10] mô tả mạng có 3 layers, input layer có 784 neurals, hidden input (ở đây chỉ có 1 hidden layer) có 30 neurals, và output layer có 10 neural. Lý do input layers có đến 784 (là con số mặc định) bởi vì nó mô tả dữ liệu của một image có kích thước 28\*28, output layers gồm 10 neurals tương ứng với số lớp (từ 0 đến 9), còn hidden layers, nó có từ 1 lớp trở lên, số hidden layers càng lớn thì hiệu suất phân lớp càng cao, tuy nhiên điều đó cũng tỉ lệ thuận với sự phức tạp mà chương trình gặp phải. Các tham số 5 và 3.0 mô tả số lần huấn luyện trên tập training và hệ số learning rate. Số lần huấn luyện càng nhiều thì hiệu suất chương trình càng cao, tất nhiên đi kèm với vấn đề thời gian và độ phức tạp tăng dần. Còn hệ số learning rate nên thuộc đoạn [1.0, 10.0] . Nếu hệ số này quá nhỏ hoặc quá lớn sẽ ảnh hưởng tới hiệu quả của chương trình.

### III. Tài liệu tham khảo

1. <http://neuralnetworksanddeeplearning.com/chap1.html>
2. <https://theprosperousheartblog.wordpress.com/2016/12/05/artificial-neural-network-ep-002/>
3. slide môn học ‘Các hệ cơ sở tri thức’

#### IV. Nội dung phân công

STT	Họ và tên	MSSV	Nhiệm vụ	Kết quả
1	Huỳnh Gia Bảo	1412036	Chương trình nhận dạng	Hoàn thành
2	Võ Phương Hòa	1412192	1. Tiền xử lý dữ liệu 2. Báo cáo 3. Hiệu chỉnh code	Hoàn thành