**­­­­­­­­­BỘ GIÁO GIỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

# ­­

# ĐỒ ÁN TỐT NGHIỆP

**XÂY DỰNG ỨNG DỤNG HỌC TIẾNG ANH DÙNG GIẢI THUẬT PHÂN LỚP DỮ LIỆU**

**Giảng viên hướng dẫn: Phạm Thị Kim Ngoan**

**Sinh viên thực hiện: NGUYỄN TẤT CHỦ**

**55133917 – 55.CNTT1**

*Nha Trang –2017*

LỜI CẢM ƠN

Lời cảm ơn đầu tiên em xin gửi đến quý thầy cô Khoa Công nghệ Thông tin trường Đại học Nha Trang đã truyền dạy cho em những kiến thức cho em trong thời gian qua để em có thể hoàn thành quá trình nghiên cứu và thực hiện đề tài đồ án. Và hơn hết em xin chân thành cảm ơn cô Phạm Thị Kim Ngoan, người đã nhiệt tình hướng dẫn cho em trong suốt quá trình làm đề tài.

Do điều kiện thời gian thực hiện có hạn, khả năng nghiên cứu và kinh nghiệm thực tế còn hạn chế nên bài báo cáo sẽ có nhiều thiếu sót. Em rất mong nhận được sự đóng góp quý báu của quý Thầy cô và các bạn để em có điều kiện bổ sung, nâng cao kiến thức tốt hơn cho việc học tập, nghiên cứu và công việc sau này.

Cuối cùng em kính chúc quý Thầy cô dồi dào sức khỏe, niềm tin để tiếp tục sự nghiệp cao quý của mình là truyền đạt kiến thức cho thế hệ mai sau.

*Nha Trang,* ngày … tháng 06 năm 2017

Sinh viên thực hiện

***Nguyễn Tất Chủ***

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT 4

DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNHLỜI MỞ ĐẦU 5

CHƯƠNG 1: TỔNG QUAN 7

1.1. Giới thiệu 7

1.2. Đối tượng và phạm vi nghiên cứu 7

1.2.1. Lý thuyết 7

1.2.2. Thực tiễn 8

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 9

2.1. Nền tảng xây dựng ứng dụng NodeJS 9

2.1.1. Đặc điểm của NodeJS 9

2.1.2. Khi nào nên sử dụng NodeJS? 10

2.1.3. Cài đặt môi trường NodeJS và các công cụ phát triển 10

2.1.4. Phát triển ứng dụng với NodeJS 10

2.2. Angular Framework 12

2.3. NoSQL và hai hệ quản trị CSDL MongoDB, Redis 14

2.3.1. Cơ sở dữ liệu NoSQL 14

2.3.2. Hệ quản trị CSDL MongoDB 16

2.3.3. Hệ thống lưu trữ cache máy chủ Redis 17

2.4. Thuật toán phân lớp dữ liệu bằng Neural Network 17

2.4.1. Ý tưởng xây dựng mạng Neural nhân tạo 18

2.4.2. Cấu trúc Neural nhận tạo 20

2.4.3. Mạng Neural nhân tạo và luật học 21

2.4.4. Một số cấu trúc mạng Neural điển hình 22

2.4.5. Các yếu tố ảnh hưởng đến độ chính xác của mạng Neural 23

2.5. Bài toán nhận dạng chữ viết tay 23

CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG 24

3.1. Xây dựng bộ dữ liệu sử dụng 24

3.2. Xây dựng mạng 24

3.3. Xây dựng ứng dụng 24

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 25

# DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Ký hiệu viết tắt** | **Từ gốc** | **Diễn giải** |
| 1 | I/O | Input/Ouput |  |
| 2 | API | Application Programming Interface |  |
| 3 | JSON | JavaScript Object Noattion |  |
| 4 | IDE | Integrated Development Environment |  |
| 5 | MVC | Model – View – Controller |  |
| 6 | MVVM | Model – View – View Model |  |
| 7 | MVW | Model – View – Whatever | Mô hình kết hợp |
| 8 | CSDL | Cơ sở dữ liệu |  |

# DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNHLỜI MỞ ĐẦU

Ngày nay, với sự xu hướng hội nhập toàn cầu, tiếng Anh đã trở thành yếu tố thiết yếu cho sự phát triển của mỗi cá nhân và của toàn xã hội, theo đó nhu cầu học tiếng Anh ngày càng phát triển, bằng chứng là tiếng Anh đã được đưa vào chương trình giáo dục từ lớp 1. Cùng với đó các ứng dụng giúp các các em học sinh có thể cải thiện tiếng Anh của mình cũng được xây dựng rất nhiều, tuy nhiên những những dụng bám sát nội dung cách học theo chương trình đào tạo đặc biệt là kỹ năng luyện viết của các em gần như chưa có.

Từ khi ra đời, máy tính đã nhanh chóng phát triển và đóng một vai trò rất quan trọng trong nghiên cứu khoa học kỹ thuật cũng như trong đời sống. Nhưng một máy tính dù có mạnh đến đâu chăng nữa, cũng chỉ có thể làm việc theo một chương trình đã được hoạch định sẵn bởi lập trình viên. Nó vẫn không có khả năng liên tưởng, kết nối sự việc này với sự việc khác, và quan trọng hơn hết là khả năng sáng tạo như con người.

Trong những năm gần đây trí tuệ nhân tạo đã len lỏi vào hầu hết các lĩnh vực, giúp máy tính có khả năng nhận biết, phân biệt các đối tượng với nhau từ đó có thể xây dựng được những ứng dụng vô cùng đa dạng.

Nổi lên trong lĩnh vực trí tuệ nhân tạo đó là mô hình mạng Neural nhân tạo có thể xây dựng được những hệ thống thông minh với độ chính xác cao. Đồ án tốt nghiệp này đi vào nghiên cứu mạng Neural nhân tạo và tích hợp vào ứng dụng trên nền tảng NodeJS và các hệ quản trị CSDL thế hệ mới để xây dựng ứng dụng giúp các em học sinh cải thiện khả năng học tiếng Anh theo chương trình đào tạo của bộ Giáo dục.

Nội dung báo cáo gồm 4 chương:

* Chương 1: Tổng quan và đặt vấn đề
* Chương 2: Cơ sở lý thuyết xây dựng ứng dụng
* Chương 3: Xây dựng ứng dụng
* Chương 4: Kết quả thực hiện, nghiên cứu và hướng phát triển

# CHƯƠNG 1: TỔNG QUAN

## Giới thiệu

Mạng neural nhân tạo (Artificial Neural Networks : ANN) ra đời xuất phát từ ý tưởng mô phỏng hoạt động của bộ não con người.

Mạng noron nhân tạo là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người với vô số các neural được liên kết truyền thông với nhau qua mạng. Giống như con người, ANN được học bởi kinh nghiệm, lưu những kinh nghiệm đó và sử dụng trong các tình huống

Mạng neural trong một vài năm trở lại đây đã được nhiều người quan tâm và đã áp dụng thành công trong nhiều lĩnh vực khác nhau, như tài chính, y tế, địa chất và vật lý. Thật vậy, bất cứ ở đâu có vấn đề về dự báo, phân loại và điều khiển, mạng neural đều có thể ứng dụng được. Ví dụ như khả năng nhận dạng mặt người trong các hệ thống quản lý thông tin liên quan đến con người (quản lý nhân sự ở các công sở, doanh nghiệp; quản lý học sinh, sinh viên trong các trường trung học, đại học và cao đẳng;… ); các ngành khoa học hình sự, tội phạm; khoa học tướng số, tử vi,…

Kết hợp chặt chẽ với logic mờ, mạng neural nhân tạo đã tạo nên cuộc cách mạng thực sự trong việc thông minh hóa và vạn năng hóa các bộ điều khiển kỹ thuật cao cho cả hiện nay và trong tương lai. Ví dụ như ứng dụng tự động điều khiển hệ thống lái tàu, hệ thống dự báo sự cố,…

Mạng neural dựa trên việc mô phỏng cấp thấp hệ thống neural sinh học. Trong tương lai với sự phát triển mô phỏng neural sinh học, chúng ta có thể có loại máy tính thông minh thật sự.

Rất nhiều nghiên cứu đã cho thấy khả năng đặc biệt của mạng Neural nhân tạo, nhưng việc tích hợp vào các ứng dụng thực tế để sử dụng rỗng rãi vẫn còn rất hạn chế. Căn cứ vào thực tế đó cùng nhu cầu học tiếng Anh bám sát chương trình đào tạo, đề tài này sẽ tập trung nghiên cứu một nền tảng phát triển ứng dụng NodeJS cùng với các kỹ thuật có liên quan về CSDL, khả năng phát triển ứng dụng trên nền tảng này trong lĩnh vực trí tuệ nhân tạo để xây dựng ứng dụng hỗ trợ học tiếng Anh.

## Đối tượng và phạm vi nghiên cứu

### Lý thuyết

Nhằm xây dựng một ứng dụng phù hợp với vấn đề đặt ra, về mặt lý thuyết đề tài sẽ tìm hiểu và nghiên cứu các lĩnh vực sau:

* Tìm hiểu nền tảng NodeJS, ngôn ngữ lập trinh JavaScript và các thư viện trong xây dựng ứng dụng website
* Nghiên cứu CSDL NoSQL và hệ quản trị CSDL MongoDB, Redis
* Nghiên cứu kỹ thuật phân lớp dữ liệu bằng mạng Neural và các ứng dụng
* Tìm hiểu các kỹ thuật xử lý ảnh, trích chọn đặc trưng trong quá trình xây dựng mạng Neural

### Thực tiễn

Từ cơ sở lý thuyết đề tài sẽ xây dựng ứng dụng hỗ trợ học từ vựng tiếng Anh bằng NodeJS với các tính năng:

* Thông qua phương pháp nhận dạng chữ viết tay xây dựng tính năng học từ vựng giúp các em học sinh dễ nhớ dễ thuộc từ vựng
* Hỗ trợ tra cứu từ điển, phát âm từ điển

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## Nền tảng xây dựng ứng dụng NodeJS

Node.js là một nền tảng dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các phần phát sinh các sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

NodeJS là một mã nguồn mở, đa nền tảng được thiết kế để xây dựng các ứng dụng Internet nhanh, có độ lớn, có khả năng mở rộng phía máy chủ dựa trên Chrome Javascript Engine V8 được phát triển bởi Ryan Dahl năm 2009 dưới sự bảo trợ của Joyent.

Những môi trường tương tự được viết trong các ngôn ngữ khác bao gồm Twisted cho Python, Perl Object Environment cho Perl, libevent cho C và EventMachine cho Ruby. Khác với hầu hết các chương trình Javascript, Nodejs không chạy trên một trình duyệt mà chạy trên Server. Node.js sử dụng nhiều chi tiết kỹ thuật của CommonJS. Nó cung cấp một môi trường REPL cho kiểm thử tương tác.

NodeJS cung cấp cho nhà phát triển các module JavaScript đa dạng, giúp cho việc phát triển ứng dụng dễ dàng hơn, tuy nhiên NodeJS chỉ là môi trường, điều này có nghĩa là nhà phát triển phải làm mọi thứ. Không có một máy chủ nào mặc định, một đoạn script sẽ xử lý tất cả các kết nối từ máy khách. Điều này làm giảm được đáng kể tài nguyên được sử dụng trong ứng dụng.

### Đặc điểm của NodeJS

Một vài đặc điểm quan trọng của NodeJS khuyến cho NodeJS trở thành lựa chọn hàng đầu cho các nhà phát triển phần mềm:

**Không đồng bộ và Phát sinh sự kiện (Event Driven):** Khác với ngôn ngữ C#, PHP hay Java, tất các các APIs của thư viện Node.js đều bất đồng bộ (non-blocking). Điều này rất cần thiết vì Node.js không bao giờ đợi một API trả về dự liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về Sự kiện của Node.js giúp Server nhận đựa phản hồi từ các API gọi trước đó.

**Chạy rất nhanh:** Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực thi các đoạn code.

**Các tiến trình đơn giản nhưng hiệu năng cao:** Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.

**Không đệm:** Ứng dụng NodeJS không lưu trữ các dữ liệu buffer.

**Có giấy phép:** mã nguồn mở NodeJS (<https://github.com/nodejs/node>) được phát hành dựa vào MIT License.

Với những đặc điểm này NodeJS là một sự lựa chọn hoàn hảo cho các lĩnh vực gồm:

* Các ứng dụng về I/O
* Các ứng dựng về luồng dữ liệu
* Các ứng dụng về dữ liệu hướng đến thời gian thực
* Các ứng dụng dựa vào JSON APIs
* Các ứng dụng Single Page Application

Hiện tại có rất nhiều hệ thống lớn hiện tại đang sử dụng NodeJS, trong số này có thể kể đến như: eBay, GE, GoDaddy, Microsoft, Yahoo, Linkedin, Paypal,Uber,…

### Khi nào nên sử dụng NodeJS?

### Cài đặt môi trường NodeJS và các công cụ phát triển

Để cài đặt môi trường phát triển NodeJS, có hai cách là tải phiên bản cài đặt từ trang chủ <https://nodejs.org/en/download/> theo hệ điều hành tương ứng hoặc sử dụng các công cụ hỗ trợ cài đặt gói thông qua dòng lệnh:

* Trên Windows sử dụng Chocolatey(<https://chocolatey.org)>: *choco install nodejs*
* Trên MacOS sử dụng Homebrew(<https://brew.sh)>: *brew install node*
* Trên Linux sử dụng trình cài đặt gói mặc định: *sudo apt-get install nodejs*

Sau khi cài đặt sử dụng lệnh *node –*v, để kiểm tra đã cài đặt thành công hay chưa, nếu thành công dòng lệnh này sẽ in ra màn hình console phiên bản NodeJS đã cài trên hệ thống.

Ngoài điều kiện tiên quyết là cài đặt môi trường, ngoài các ứng dụng soạn thảo mặc định của hệ đièu hành để thuận tiện cho quá trình phát triển cần có thêm những trình soạn thảo code (Editor) hoặc IDE, các IDE miễn phí và nổi bật thường được các lập trình viên trên thế giới sử dụng bao gồm: Visual Studio Code, Sublime Text, Notepad++, …

### Phát triển ứng dụng với NodeJS

Bất kỳ lập trình viên nào muốn xây dựng ứng dụng với NodeJS cũng cần phải biết và hiểu được các khái niệm như module, lập trình không đồng độ, function callback, promise, các khái niệm về giao thức giữa máy khách với máy chủ (client – server), cách giao tiếp của chúng với nhau, cũng như các đối tượng, các lớp, các hàm được cung cấp sẵn của framework này. Phần dưới đây sẽ đưa ra những định nghĩa và ví dụ minh hoạ.

**Modules** trong NodeJS giống như các thư viện trong C, C#, Java, … mỗi module là một đối tượng (Object) chứa các hàm, các hằng số, … Điều này làm cho việc xây dựng các ứng dụng phức tạp trở nên đơn giản hơn. Mỗi module đóng gói tập các phương thức, thông tin, … liên quan đến một đối tượng, chức năng của Module. Ví dụ **fs** là Module chứa các hàm, các hằng số cụ thể liên quan đến việc đọc ghi dữ liệu hệ thống. Ngoài những module được NodeJs cung cấp sẵn để thao tác với hệ thống và một số tiện ích khác. NodeJs cho phép bất kỳ ai cũng có thể tạo ra module của riêng mình, hơn thế nữa NodeJs đi kèm với trình quản lý gói **npm** (<https://www.npmjs.com)> giúp cho việc cài đặt tích hợp các module vào ứng dụng của mình trở nên vô cùng đơn giản.

Các thao tác cần thiết để sử dụng một module đó là cài đặt module và tham chiếu tới module.

* Cú pháp cài đặt bằng npm: **npm install** *<tên package>.*Ví dụ: *npm install http*
* Cú pháp gọi tham chiếu module: **var** <*tên biến> =* **require(‘***<tên module>***’)**.   
  Ví dụ: *var http = requỉe(‘http’)*

Trong ví dụ trên hàm *require* trả về tham chiếu tới module *http* được cài đặt từ internet về thông qua lệnh *npm install*. Để tham chiếu tới một Module cục bộ do trên hệ thống phải chỉ rõ đường dẫn tới tập tin của Module đó. Ví dụ: *var myModule = requỉe(‘./moduleName.js’)*

Có thê xem mỗi module là một đoạn mã được đóng gói lại với nhau, mã lệnh bên trong module có phạm vi là *private*. Gần giống như việc public một hàm của một Class, bằng việc sử dụng đối tượng exports, sẽ giúp các hàm, các hằng private có thể để đưa ra ngoài sử dụng cho trên ứng dụng. Xem xét ví dụ dưới đây:

const PI = **Math**.PI;

**exports**.dienTich = function (r) {

return PI \* r \* r;

};

**exports**.chuVi = function (r) {

return 2 \* PI \* r;

};

Đoạn mã trên tạo ra hằng số *PI* và hai function, hằng số *PI* chỉ có thể sử dụng trong nội bộ module đang định nghĩa ở trên. Qua việc sử dụng đối tượng ***exports***, hai hàm *dienTich* và *chuVi* có thể sử dụng khi tham chiếu tới Module này.

**Global Object**: các object có sẵn ở tất cả các module bao gồm: Buffer, \_\_dirname, \_\_filename, console, exports, global, module, process; các function có sẵn bao gồm: require, setTimeout, … và một class có sẵn là Buffer.

**Asynchronous** (bất đồng bộ): Không giống như các ngôn ngữ lập trình truyền thống C#, Java, C++,… chương trình sẽ chạy tuần tự từng lệnh và chỉ thực hiện lệnh tiếp theo khi lệnh trước đó đã thực hiện xong, điều này sẽ sinh ra một tráng thái hay gọi là trạng thái chờ, Javascript là ngôn ngữ lập trình bất đồng bộ, khi thực thi chương trình có thể bỏ qua một số bước chờ không cần thiết, thực hiện nhiều công việc song song cùng lúc.

Một ví dụ về lập trình bất đồng bộ:

const fs = require('fs'); // tham chiếu module file system

let filePath = './text.txt'; // khai báo đường dẫn tập tin

// Thực hiện đọc tập tin

fs.readFile(filePath, function done(error, data){

console.log(data.toString()); // Xuất kết quả đọc được

});

console.log('Finished'); // Thông báo kết thúc

Trong ví dụ trên theo trình tự nội dung tập tin sẽ được ghi lên màn hình trước dòng thông báo kết thúc “Finished”, tuy nhiên kết quả thực tế khi thực thi dòng thông báo kết thúc sẽ thực hiện trước bởi bì hàm fs.readFile là hàm bất đồng bộ. Khi thực gặp lệnh này chương trình sẽ thực gọi sự kiện thực hiện lệnh này và không chờ lệnh này thực hiện xong để thực thi lệnh tiếp theo mà sẽ thực hiện các lệnh tiếp theo tiếp, khi nào hàm này thực hiện xong sẽ trả về một sự kiện được thực hiện trong hàm *done.* Hàm done nếu có lỗi, biến lỗi sẽ được nhận trong tham số thứ nhất là *error*, ngược lại nếu thành công, biến *error* sẽ nhận giá trị **null** và giá trị đọc được sẽ nhận vào tham số thứ hai là *data*. Đây là quy ước chung cho các hàm bất đồng bộ nó được gọi là cơ chế callback.

Như vậy để thực hiện các lệnh một cách tuần tự, nhà phát triển sẽ lồng các câu lệnh vào trong hàm callback. Trong trường hợp cần rất nhiều các câu hàm, câu lệnh thực hiện tuần tự sẽ tạo nên một cơ chế callback lồng nhau rất phức tạp, rất khó sử dụng và phát hiện lỗi.

Để giải quyết tình trạng trên có một cơ chế được xây dựng đó là Promise, tuy nhiên Promise không phải là cách giải quyết tối ưu nhất, ở phiên bản mới nhất của NodeJS đã hỗ trợ cơ chế async/await theo chuẩn ES7. Đây là cách giải quyết tối ưu nhất trong vấn đề bất đồng bộ của Javascript. Phần này sẽ được trình bày cụ thể ở chương tiếp theo.

**Kiểu dữ liệu JSON:** là một dạng dữ liệu dùng để trao đổi dữ liệu giữa các ngôn ngữ, nền tảng với nhau. Kiểu dữ liệu JSON có đặc điểm dễ viết, dễ đọc, dễ phân tích và có thể tạo ra một cách dễ dàng. JSON được xây dựng bởi hai cấu trúc:

* Một tập hợp các cặp tên - giá trị. Trong các ngôn ngữ khác nhau, nó được xây dựng như một đối tượng, bản ghi, struct, vector, danh sách, hoặc mảng kết hợp.
* Là 1 tập hợp các giá trị đã được sắp xếp.

Một ví dụ về kiểu dữ liệu JSON:

**{**

"***MSSV***": 55133917,

"***ho\_ten***": "Nguyễn Tất Chủ",

"***gioi\_tinh\_nam***": true,

"***tong\_ket***": ***null***,

"***cac\_mon\_da\_hoc***": **[**

"Kỹ thuật lập trình",

"Tin học cơ sở",

"Trí tuệ nhân tạo"

**]**,

"***thong\_tin\_lop***": **{**

"lop": "55-CNTT1",

"khoa-hoc": "2013-2017"

**}**

**}**

Trong ví dụ trên giá trị bên trái dấu “:” là khoá, bên phải là giá trị. Các giá trị được lưu trữ ở nhiều dạng khác nhau, có thể là một số, một giá trị Boolean, giá trị một mảng hoặc một object khác, cặp ngoặc “{}” biểu thị cho 1 object, “[]” được hiểu như một mảng.

Có một kiểu ngôn ngữ tương tự là XML, tuy nhiên trong phạm vi đề này không đề cập đến.

**REST:** viết tắt của Representaion State Transfer, là một kiến trúc trong việc thiết kế hệ thống phân tán (distributed system) kiểu trao đổi dữ liệu dạng JSON hoặc XML mà sử dụng các ràng buộc chặt chẽ. Một số tiêu chuẩn của REST bao gồm:

* Mô hình client - server
* Phi trạng thái (stateless interation), có thể lưu nhưng ko đủ điều kiện xác thực.
* Thống nhất giao thức (uniform interface).

Các cơ bản quy tắc được quy định chung của RESP thông qua giao thức HTTP bao gồm:

* Để tạo một tài nguyên trên máy chủ sử dụng phương thức POST.
* Để truy xuất một tài nguyên sử dụng GET.
* Để cập nhật một tài nguyên sử dụng PUT.
* Để xoá một tài nguyên sử dụng DELETE.

## Angular Framework

AngularJS là một framework viết bằng Javascript được sử dụng phía client có cấu trúc cho các ứng dụng web động. Nó cho phép bạn sử sụng HTML như là ngôn ngữ mẫu và cho phép mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng một cách rõ ràng súc tích.

AngularJS là framework mã nguồn mở hoạt động dưới giấy phép Apache Lincense được đưa ra lần đầu năm 2009 bởi Misko Hevery và Adam Abrons. Hiện tại thư viện này được duy trì và phát triển bởi hãng Google.

Các tính năng cốt lõi của Framework AngularJS bao gồm:

* Data-binding: Nó tự động đồng bộ hóa dữ liệu giữa thành phần model và view.
* Scope: Là những đối tượng hướng đến model, nó hoạt động như là cầu nối giữa controller và view
* Controller: Đây là những tính năng của AngularJS mà được giới hạn tới một scope cụ thể
* Service: AngularJS hoạt động với một vài dịch vụ (service) có sẵn , ví dụ $http để tạo XMLHttpRequests. Nó là các singleton object mà được khởi tạo duy nhất một lần trong ứng dụng
* Filter: Nó lựa chọn (hay là lọc) các tập con từ tập item trong các mảng và trả về các mảng mới
* Directive: Directive là các marker trong các phần tử DOM (như các phần tử, thuộc tính, css và nhiều hơn thế). Nó có thể dùng để tạo các thẻ HTML riêng phục vụ những mục đích riêng. AngularJS có những directive có sẵn như ngBind, ngModel, ngController, ngApp…
* Template:Là các rendered view với các thông tin từ controller và model. Nó có thể được sử dụng trong các file riêng rẽ (ví dụ như index.html) hoặc nhiều view với một trang sử dụng “partials”
* Routing: Là khái niệm của sự chuyển dịch qua lại các view
* Deep Linking: Cho phép bạn mã hóa trạng thái các ứng dụng trên địa chỉ URL để nó có thể được bookmark. Các ứng dụng có thể được phục hồi lại từ các địa chỉ URL với cùng một trạng thái
* Dependency Injection: AngularJS có sẵn một hệ thống con dependency injection để giúp các lập trình viên tạo ra các ứng dụng dễ phát triển, dễ hiểu và kiểm tra

AngularJS được thiết kế kết hợp giữa mô hình MVC và MVVM hay còn được là MVW. Những ưu điểm nổi bật của AngularJS đó là:

* AngularJS cung cấp khả năng tạo ra các Single Page Application với API service một cách rất rõ ràng và dễ dàng để duy trì, nâng cấp.
* AngularJS cung cấp khả năng Data binding tới HTML do đó giúp người dùng cảm giác linh hoạt, thân thiện, trực quan.
* AngularJS code dễ dàng trong giai đoạn unit test, functional testing.
* AngularJS sử dụng dependency injection.
* AngularJS cung cấp khả năng tái sử dụng các component (thành phần).
* Với AngularJS, lập trình viên sẽ viết ít code hơn, với nhiều chức năng hơn.
* Với AngularJS, view là thành phần trong trang HTML thuần, trong khi controller được viết bởi JavaScript với quá trình xử lý nghiệp vụ.
* AngularJS có thể sử dụng kết hợp với các framework, thư viện khác mà không gây xung đột

Bên cạnh những ưu điểm, AngularJS cũng có những nhược điểm riêng:

* Không an toàn là một JavaScript Framework, ứng dụng được viết bởi AngularJS nên không an toàn. Vấn đề này được giải quyết nếu phía máy chủ web có cơ chế chứng thực và phân quyền phù hợp
* Được xây dựng bằng JavaScript nên khi người sử dụng vô hiệu hoá tính năng Javascript thì ứng dụng bị vô hiệu hoá.

Các thành phần của AngularJS quan trọng khi xây dựng ứng dụng bao gồm các directive: ngApp, ngController, ngModel,… trong chương tiếp theo sẽ cụ thể hoá các thành phần này trên ứng dụng mà đồ án xây dựng.

## NoSQL và hai hệ quản trị CSDL MongoDB, Redis

### Cơ sở dữ liệu NoSQL

Với các công nghệ phát triển website, hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ Quản trị Cơ sở dữ liệu. Tuy nhiên thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống

Thuật ngữ NoSQL có nghĩa là Non-Relation hoặc Not Only SQL – không rằng buộc hoặc phi quan hệ, ám chỉ những đến CSDL không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm. Bảng so sánh dưới đây sẽ phân biệt NoSQL và SQL:

|  |  |  |
| --- | --- | --- |
|  | ***NoSQL*** | ***SQL*** |
| Mô hình cấu trúc | * Không có quan hệ * Lưu trữ dữ liệu dứoi dạng JSON, key-value, graph, … | * Có quan hệ * Lưu trữ dữ liệu dưới dạng các bảng |
| Dữ liệu | * Linh hoạt, dữ liệu không cần lưu những thuộc tính không cần thiết * Có thể bổ xung thuộc tính bất cứ khi nào một cách dễ dàng * Các quan hệ thường được tóm gọn và trình bày trong một đối tượng trên mỗi dòng dữ liệu * Tốt cho các trường hợp dữ liệu không có cấu trúc, phức tạp hoặc lồng nhau | * Thêm thuộc tính có thể yêu cầu thay đổi cấu trúc các bảng hoặc dữ liệu bị ghi đè * Quan hệ được tóm gọn và tổng quát đế sử dụng kết nối và tham chiếu tới các bảng * Tốt cho dữ liệu có cấu trúc và các thuộc tính thường được cố định, không thay đổi |

Bảng 1: Bảng so sánh CSDL NoSQL và cư cở dữ liệu quan hệ truyền thống

Những đặc điểm của CSDL NoSQL :

* Looser consistency: NoSQL tổ chức lưu trữ và truy xuất dữ liệu theo cơ chế “thoáng hơn trong đảm bảo tính nhất quán của dữ liệu” so với mô hình dữ liệu quan hệ tryền thống nhằm cải thiện hiệu suất, đảm bảo dữ liệu luôn được đáp ứng tốt hơn, CSDL này chấp nhận sự trùng lặp dữ liệu.
* Eventual consistency: NoSQL không yêu cầu phải đảm bảo tính nhất quán của dữ dữ liệu ngay tức thì mà sẽ hiện thực tính nhất quán của dữ liệu theo cơ chế lan truyền
* Distributed storage: hay còn gọi là lưu trữ phân tán là thay vì dữ liệu được lưu trữ trên một máy chủ duy nhất thì hệ thống sẽ lưu dữ liệu trên nhiều máy khác nhau dưới sự kiểm soát của phần mềm
* Horizontal scalable: hay còn gọi là khả năng mở rộng chiều ngang. Bình thường, với các hệ quản trị cơ sở dữ liệu quan hệ , khi mà dữ liệu quá lớn phương pháp tăng khả năng lưu trữ là sẽ phải mở rộng (nâng cấp máy chủ), còn đối với NoSQL thì chỉ cần bổ sung thêm máy chủ khác vì hệ thống hỗ trợ lưu trữ phân tán trên nhiều máy

Ưu điểm của NoSQL:

* NoSQL là mã nguồn mở: điều này có nghĩa là bạn sẽ dễ dàng phát triển một ứng dụng có sử dụng NoSQL mà không phải tốn chi phí license.
* Dễ mở rộng quy mô: NoSQL đã thay thế cách mở rộng quy mô truyền thống của các hệ quản trị cơ sở dữ liệu quan hệ bằng hình thức “mở rộng ra ngoài”. Với hình thức mở rộng đặc biệt này, thay vì phải bổ sung thêm các máy chủ lớn hơn vào hệ thống khi dữ liệu lớn lên, thì NoSQL lại hỗ trợ doanh nghiệp phân tán dữ liệu qua nhiều máy chủ khi dữ liệu gia tăng.
* Hỗ trợ các mô hình dữ liệu khác nhau tuỳ mục đính và cách thức lưu trữ dữ liệu như lưu kiểu key-value, BigTable, lưu document hay lưu thông tin graph

Nhược điểm của CSDL này đó là:

* Thiếu tính tương thích: các CSDL NoSQL khác nhau không tương thích với nhau, mỗi CSDL NoSQL có một giao diện và cung cấp API khác nhau và chưa có một tiêu chuẩn chung nào. Điều này có nghĩa là bạn sẽ gặp khó khăn trong việc chuyển từ nhà cung cấp này sang nhà cung cấp khác
* Hạn chế về nghiệp vụ: NoSQL hiện chưa hỗ trợ các dạng phân tích dữ liệu lớn và mạnh mẽ mà các doanh nghiệp đã quen thuộc trong các RDBMS

Phân loại các CSDL NoSQL: Có nhiều cách phân loại các cơ sở dữ liệu NoSQL khác nhau, mỗi loại với các loại và loại con khác nhau, một số trong số đó có thể chồng chéo lên nhau. Dưới đây là một phân loại cơ bản dựa trên mô hình dữ liệu, chia các hệ quản trị CSDL thành các nhóm khác nhau:

* Column: Accumulo, Cassandra, Druid, HBase, Vertica
* Document: Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
* Key-value: Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB
* Graph: AllegroGraph, InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog
* Multi-model: Alchemy Database, ArangoDB, CortexDB, FoundationDB, MarkLogic, OrientDB

Một phân loại chi tiết hơn như sau, dựa trên cách phân loại của Stephen Yen[[1]](#endnote-1):

|  |  |
| --- | --- |
| **Loại** | **Hệ quản trị CSDL** |
| Key-Value Cache | Coherence, eXtreme Scale, GigaSpaces, GemFire, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached, Terracotta, Velocity |
| Key-Value Store | Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike |
| Key-Value Store (Eventually-Consistent) | DovetailDB, Oracle NoSQL Database, Dynamo, Riak, Dynomite, MotionDb, Voldemort, SubRecord |
| Key-Value Store (Ordered) | Actord, FoundationDB, Lightcloud, LMDB, Luxio, MemcacheDB, NMDB, Scalaris, TokyoTyrant |
| Máy chủ Cấu trúc dữ liệu | Redis |
| Tuple Store | Apache River, Coord, GigaSpaces |
| Cơ sở dữ liệu đối tượng | DB4O, Objectivity/DB, Perst, Shoal, ZopeDB |
| Document Store | Clusterpoint, Couchbase, CouchDB, DocumentDB, Lotus Notes, MarkLogic, MongoDB, Qizx, RethinkDB, XML-databases |
| Wide Column Store | BigTable, Cassandra, Druid, HBase, Hypertable, KAI, KDI, OpenNeptune, Qbase |

Bảng 2: Bảng phân loại hệ quản trị CSQL NoSQL

### Hệ quản trị CSDL MongoDB

Nổi lên trong những CSDL NoSQL, MongoDB (bắt nguồn từ humongous) là một mã nguồn mở miễn phí được viết bằng C++, đa nền tảng thuộc dạng CSDL NoSQL lưu trữ dữ liệu theo hướng Document dạng JSON với. MongoDB được phát triển bởi MongoDB Inc và được xuất

* Các đặc điểm?
* Cài đặt và sử dụng?
* Hiệu xuất?

### Hệ thống lưu trữ cache máy chủ Redis

Redis là một lựa chọn tốt nhất trong những việc xây dựng các ứng dụng cần lưu trữ dữ liệu dạng Memcached . Memcached là một dạng lưu trữ có các đặc điểm: lưu trữ dữ liệu theo đạng key-value, tất cả dữ liệu trên RAM, dữ liệu có thể hết hạn hoặc không và đặc điểm quan trọng nhất của Memcached đó là có thể truy xuất rất nhanh.

Redis giống MongoDB thuộc dạng CSDL NoSQL lưu trữ dữ liệu dạng key – value. Bên cạnh lưu trữ key-value trên RAM với hiệu năng cao, redis còn hỗ trợ lưu trữ dữ liệu trên đĩa cứng (persistent redis) cho phép phục hồi dữ liệu khi gặp sự cố. Ngoài tính năng replicatation (sao chép giữa master-client), tính năng cluster (sao lưu master-master) cũng đang được phát triển.

Ngoài những đặc điểm phù hợp cho Memcahed, Redis còn có các ưu điểm:

* Hỗ trợ nhiều Databases
* Truy vấn theo Key
* Hỗ trợ counters dữ liệu kiểu integer
* Cấu trúc dữ liệu cấp cao: key là một string nhưng value thì không giới hạn ở một string mà có thể là List, Sets, Sorted sets, …
* Thao tác dữ liệu chuyên biệt
* Tự động phân trang danh sách
* Redis hỗ trợ mở rộng master-slave nếu chúng ta muốn sự an toàn hoặc mở rộng, co giãn trong việc lưu trữ data

Các cài đặt và sử dụng Redis: Redis là một phần mềm mã nguồn mở nên nhà phát triển có thể cài tự biên dịch chương trình để chạy theo cơ chế services của hệ điều hành hoặc sử dụng các trình cài đặt gói. Ví dụ:

* Trên macOS: *brew install redis*
* Trên Windows: *choco install redis-64*

Việc sử dụng Redis được tích hợp tuỳ theo framework khác nhau. Bộ thư viện làm việc có thể tham khảo tại: <https://github.com/nrk/predis>

## Thuật toán phân lớp dữ liệu bằng Neural Network

Nghiên cứu về nơron thần kinh từ lâu đã trở thành đề tài được nhiều nhà khoa học quan tâm. Nhưng kỷ nguyên của mạng nơron chính thức được bắt đầu với báo cáo khoa học của Mc Culloch và Pitts năm 1943[[2]](#endnote-2) mô tả một phép tính logic của mạng nơron. Báo cáo này được công chúng đón nhận cho đến năm 1949 học thuyết về mạng nơron chính thức của Mc. Culloch và Pitts được mô tả chủ yếu trong bài giảng thứ hai trong bốn bài giảng mà Von Neumann đã phát biểu tại trường đại học Illinois. Sự phát triển tiếp theo của quá trình nghiên cứu mạng nơron được đánh dấu vào năm 1949 với việc xuất bản cuốn sách “The Organization of Behavior: A neuropsychological Theory” của Donald Olding Hebb. Đến năm 1952, cuốn sách của Ashby “Design for a Brain” đã mô tả những điều kiện cần và đủ đối với một hệ thống hoạt động giống như bộ não “đó là phải học để còn tồn tại trong môi trường luôn thay đổi và nhận được những cái nó cần”. Tiếp đó, năm 1954, Minsky đã viết luận án tiến sĩ mang tên “Theory of Neural-analog Reinforcement Systems and Application to Brain-Model Problem” tại trường Đại học Princeton, sau đó là bài báo của ông “Steps Toward Artificial Interligence” năm 1961 về việc học củng cố trong mạng nơron hiện nay. Một chủ đề khác được đánh giá cao là phát kiến về bộ nhớ liên kết của Taylor vào năm 1956, mở đầu một loạt các kết quả phát triển to lớn về sau. Các kết quả có thể kể đến là sự ra đời của mạng Perceptron được Frank Rosenblatt công bố vào năm 1957 và được coi là “mạng nơron truyền thẳng đơn giản nhất”. Tiếp đó năm 1960, mạng nơron khác được Bernard Widrow và Marcian Hoff giới thiệu là ADALINE (ADAptive LINear Element). Với mạng ADALINE lần đầu tiên kiểu hội tụ các mạch con chứa trọng số trước node tổng được sử dụng để phân lớp các mẫu. Năm 1969, Minsky và Papert xuất bản cuốn “Perceptron, An Introduction to Computational Geometry” chỉ ra những giới hạn trong mạng Perceptron một lớp đơn và đề nghị khắc phục trong mạng Perceptron nhiều lớp. Những năm 1970 nổi bật với các sự kiện: các mạng liên kết của Kohonen và Anderson (1972), Cognitron – mạng tự tổ chức nhiều lớp đầu tiên do Kunihiko Fukushima người Nhật giới thiệu năm 1975. Thời kì phát triển vượt bậc của quá trình nghiên cứu mạng nơron là những năm 80 với sự ra đời của một loạt các mạng nơron có giá trị: mạng Hopfied của John Hopfield (1982), SOM (Self-Organization Map) của Kohonen, máy Boltzmann của Ackley, Hinton và Sejnowski. Tiếp theo là sự ra đời của mạng Back- Propagation năm 1986 do D. Rumelhart, G. Hilton và R. Williams giới thiệu, sau đó là mạng ART (Adaptive Resonance Networks) vào năm 1987 của Gail Carpenter và Stephen Grossberg tại đại học Boston. Trong những năm 1990 việc ứng dụng các mô hình mạng nơron vào phục vụ các lĩnh vực trong cuộc sống được phát triển mạnh mẽ.

### Ý tưởng xây dựng mạng Neural nhân tạo

Mạng nơron nhân tạo được thiết kế dựa trên mô hình mạng nơron thần kinh với các phần tử nơron nhân tạo của nó là sự mô phỏng nơron sinh học. Các mức tổ chức bộ não và cấu trúc mạng nơron sinh vật có thể được tham khảo trong tài liệu. Trong phần này chỉ tập trung tìm hiểu cấu tạo của nơron sinh học và nơron nhân tạo để thấy được sự tương quan giữa chúng.

Nơron sinh vật có nhiều dạng khác nhau như dạng hình tháp, dạng tổ ong, dạng rễ cây. Tuy khác nhau về hình dạng, chúng có cấu trúc và nguyên lý hoạt động chung. Một tế bào nơron gồm bốn phần cơ bản sau:

Các nhánh và rễ: Các nhánh và rễ là các bộ phận nhận thông tin, các đầu nhạy hoặc các đầu ra của các nơron khác bám vào rễ hoặc nhánh của một nơron. Khi các đầu vào từ ngoài này có sự chênh lệch về nồng độ K+ , Na+ hay Cl- so với nồng độ bên trong của nó thì xẩy ra hiện tượng thấm từ ngoài vào trong thông qua một cơ chế màng thấm đặc biệt. Hiện tượng thẩm thấu như vậy tạo nên một cơ chế truyền đạt thông tin với hàng ngàn hàng vạn lối vào trên một nơron sinh vật, ứng với hàng nghìn hàng vạn liên kết khác nhau. Mức độ thẩm thấu được đặc trưng bởi cơ chế màng tượng trưng bằng một tỷ lệ. Tỷ lệ đó được gọi là tỷ trọng hay đơn giản gọi là trọng (Weight).

Thân thần kinh (Soma): Thân thần kinh chứa các nhân và cơ quan tổng hợp protein. Các ion vào được tổng hợp và biến đổi. Khi nồng độ các ion đạt đến một giá trị nhất định, xảy ra quá trình phát xung (hay kích thích). Xung đó được phát ở các đầu ra của nơron. Dây dẫn đầu ra xung được gọi là dây thần kinh.

Dây thần kinh (Axon): Dây thần kinh là đầu ra. Đó là phương tiện truyền dẫn tín hiệu. Dây thần kinh được cấu tạo gồm các đốt và có thể dài từ micro mét đến vài mét tuỳ từng kết cấu cụ thể. Đầu ra này có thể truyền tín hiệu đến các nơron khác.

Khớp thần kinh là bộ phận tiếp xúc của đầu ra nơron với rễ, nhánh của các nơron khác. Chúng có cấu trúc màng đặc biệt để tiếp nhận các tín hiệu khi có sự chênh lệch về nồng độ ion giữa bên trong và bên ngoài. Nếu độ lệch về nồng độ càng lớn thì việc truyền các ion càng nhiều và ngược lại. Mức độ thẩm thấu của các ion có thể coi là một đại lượng thay đổi tuỳ thuộc vào nồng độ như một giá trị đo thay đổi được gọi là trọng.

Trong não người có khoảng 15 tỷ nơron, mỗi nơron được nối với nhiều nơron khác bằng những khớp thần kinh (synapses), một nơron có thể có đến hơn nghìn synapses, và số synapses tổng cộng lại được ước lượng khoảng 1 triệu tỷ. Tín hiệu được truyền qua thân nơron tới các synapses, và tuỳ theo trạng thái của chúng mà một hay nhiều nơron khác sẽ được kích thích để tiếp tục truyền tín hiệu. Đó là mức thấp nhất, nhưng từ đó tổ chức lên các tầng trên như thế nào, có thể có được một bản đồ chi tiết của một bộ não tới từng nơron không?

Mạng nơron sinh vật tổ chức thành từng lớp (layer). Bao gồm:

* Mạng một lớp: là tập hợp các phần tử nơron có đầu vào và đầu ra trên mỗi một phần tử. Nếu mạng nối đầu ra của các phần tử này với đầu vào của phần tử kia gọi là mạng tự liên kết (autoassociative)
* Mạng hai lớp: gồm một lớp đầu vào và một lớp đầu ra riêng biệt.
* Mạng nhiều lớp: gồm một lớp đầu vào và một lớp đầu ra riêng biệt. Các lớp nằm giữa lớp đầu vào và lớp đầu ra gọi là lớp ẩn (hidden layers)
* Mạng truyền thẳng: là mạng hai hay nhiều lớp mà quá trình truyền tín hiệu từ đầu ra lớp này đến đầu vào lớp kia theo một hướng
* Mạng truyền ngược: là mạng mà trong đó một hoặc nhiều đầu ra của các phần tử lớp sau truyền ngược tới đầu vào của lớp trước
* Mạng truyền ngược: là mạng mà trong đó một hoặc nhiều đầu ra của các phần tử lớp sau truyền ngược tới đầu vào của lớp trước

Một tế bào nơron gồm các phần cơ bản sau:

Nhánh và rễ: Đây là bộ phận tiếp nhận thông tin của tế bào nơron. Các đầu nhạy hoặc các đầu ra của các nơron khác bám vào rễ hoặc nhánh của một nơron. Khi các đầu vào từ ngoài này có sự chênh lệch về nồng độ K+, Na+ hay Cl- so với nồng độ bên trong của nó thì xảy ra hiện tưởng thấm từ ngoài vào trong thông qua cơ chế màng thấm đặc biệt. Hiện tượng thẩm thấu như vậy tạo nên một cơ chế truyền đạt thông tin với hàng ngàn hàng vạn lối vào trên một nơron sinh học ứng với hàng nghìn hàng vạn liên kết khác nhau. Mức độ thẩm thấu của các ion có thể coi là một đại lượng thay đổi tùy thuộc vào nồng độ gọi là tỷ trọng hay đơn giản là trọng số (weight)

Dây thần kinh: Đây là phương tiện truyền dẫn tín hiệu của giữa các nơron. Dây thần kinh được cấu tạo gồm các đốt và có thể dài từ vài micro mét đến vài mét tùy từng kết cấu cụ thể

Thân thần kinh: Thân thần kinh chứa các nhân và cơ quan tổng hợp protein. Trong quá trình các ion vào được tổng hợp và biến đổi khi nồng độ các ion đạt đến một giá trị nhất định, xảy ra quá trình phát xung (hay kích thích). Xung đó được phát ở đầu ra của nơron thông qua dây thần kinh

Khớp thần kinh: Là bộ phận tiếp xúc của đầu ra nơron với nhánh, rễ của các nơron khác. Chúng có cấu trúc màng đặc biệt để tiếp nhận các tín hiệu khi có sự chênh lệch về nồng độ ion giữa bên trong và bên ngoài. Nếu độ lệch càng lớn thì việc truyền các ion càng nhiều và ngược lại.

Hình 1: Cấu trúc của một Neural sinh học

Những nghiên cứu trên đây là cơ sở mở đường cho con người nghiên cứu phương pháp mô phỏng lại mạng nơron sinh vật, đó chính là mạng nơron nhân tạo.

### Cấu trúc Neural nhận tạo

Trên cơ sở cấu trúc của nơron sinh học người ta đề xuất mô hình nơron nhân tạo gồm 3 phần chính: Bộ tổng liên kết đầu vào, bộ động học tuyến tính và bộ phi tuyến. Nơron nhân tạo là sự mô phỏng nơron sinh học. Sơ đồ dưới đây minh hoạ cụ thể cấu trúc của một Neural:

w*1*

w*2*

w*n*

θ

*f(x)*

...

∑

*v(t)*

*y(t)*

Sơ đồ 1: Cấu trúc neural nhân tạo

Trong đó:

* n: Số lượng đầu vào mô tả tín hiệu vào từ các đầu nhạy thần kinh hoặc các nơron khác
* xi(t): Các đầu vào ngoài; với i là chỉ số chạy, i=1,2...n
* wi: Trọng số liên kết ngoài giữa các đầu vào *i* tới nơron t
* θ: Độ lệch (bias), là hằng số xác định ngưỡng kích thích hay ức chế
* v(t) = : hàm tổng các đầu vào tác động ở thân nơron t
* y(t): Tín hiệu đầu ra nơron t
* *f*: hàm kích hoạt

Hàm kích hoạt là hàm thể hiện đặc điểm kích hoạt hay ức chế một neural. Đặc điểm chung của hàm này là bị chặn, đơn điệu tăng và tăng liên tục. Các hàm này được chia thành hai nhóm là hàm bước nhảy và hàm liên tục. Bảng sau đây liệt kê một số hàm thường được sử dụng:

|  |  |  |
| --- | --- | --- |
| ***Tên hàm*** | ***Công thức*** | ***Đồ thị*** |
| Bước nhảy đơn vị |  |  |
| Hàm tang-hyperbol |  |  |
| Hàm sigmoid |  |  |

Bảng 3: Các hàm kích hoạt thường dùng trong mô hình neural nhân tạo

### Mạng Neural nhân tạo và luật học

Các nơron sinh học liên kết với nhau thành lớp tạo thành mạng nơron sinh học. Lấy ý tưởng từ mô hình trên, mạng neural nhân tạo ra đời với cấu trúc tương tự trong đó một neural là một khối tính toán gồm nhiều đầu vào, mỗi đầu vào có một trọng số đặc trưng cho tính ức chế hay kích hoạt giữa các neural, mỗi neural còn được gọi là các nút (node). Có nhiều cách kết hợp các nút để thành thành một mạng neural, mỗi cách kết hợp tạo thành một lớp mạng nơron nhân tạo khác nhau, chức năng của mạng được xác định bởi các đặc điểm như cấu trúc liên kết, quá trình xử lý bên trong từng nút và mức độ liên kết các nút với nhau.

Kiến trúc mạng neural nhân tạo được xác định bởi các yếu tố:

* Số lượng các tín hiệu đầu vào và đầu ra
* Số lượng các lớp
* Số lượng nút của mỗi lớp
* Trọng số liên kết
* Cách thức liên kết giữa các neural trên một lớp hoặc giữa các lớp với nhau

Dựa trên kiến trúc của mạng neural người ta chia mạng neural thành các nhóm khách nhau, có nhiều cách phân loại mạng dựa trên các tiêu chí khác nhau.

* Dựa trên số lượng lớp trong mạng chia thành 2 loại là mạng neural một lớp và mạng nhiều lớp
* Dựa trên cách thức liên kết, phân thành 3 loại: mạng truyền thẳng, mạng hồi quy và mạng tự tổ chức.

Mức độ liên kết giữa các neural với nhau được xác định thông qua quá trình học của mạng hay còn gọi là quá trình huấn luyện mạng. Tương tự như bộ não con người có thể lưu trữ và tiếp nhận thông tin có thể xem trọng số là phương pháp để mạng neural lưu trữ thông tin tri thức giúp máy tính có thể phân biệt và dự đoán được những thông tin đã được học. Như vậy nhiệm vụ của quá trình học là cập nhật các trọng số để tìm ra một liên kế hoàn chỉnh cùng với một bộ trọng số tối ưu nhất.

* Đưa ra khái niệm học, các loại học

Quá trình học tham khảo // luận văn nghiên cứu, tìm hiểu ….. @thesls/report/luận văn nghiên cứu …. .pdf

* Các kỹ thuật học?
* Cách rút kinh nghiệm?

### Một số cấu trúc mạng Neural điển hình

* Mạng Perceptron
* Mạng Hopfield
* **Mạng Back Propagation:** cần nói rõ về mạng này, sau này sử dụng mạng này-> thuật toán lan truyền ngược
* Mạng tự

### Các yếu tố ảnh hưởng đến độ chính xác của mạng Neural

Lợi thế lớn nhất của ANN là khả năng dược sử dụng như một cơ chế xấp xỉ hàm tuỳ ý mà chúng học được từ các dữ liệu huấn luyện. Tuy nhiên việc sử dụng ANN thực tế không đơn giản như vậy. Độ chính xác của phụ thuộc vào rất nhiều yếu tố để tìm ra mạng phù hợp không phải là một việc hết sức khó khăn.

Vấn đề đầu tiên khi giải quyết bài toán với ANN đó là chọn mô hình phù hợp, việc chọn mô hình sẽ phụ thuộc vào cách tổ chức dữ liệu của ứng dụng cụ thể, mô hình đơn giản làm cho hiệu quả bài toán thấp nhưng nếu mô hình quá phức tạp thì sẽ dấn đến thách thức lớn trong quá trình huấn luyện.

Việc chọn số lớp ẩn, về mặt lý thuyết người ta đã chứng minh được mạng chỉ cần một lớp ẩn có thể giải quyết được hầu hết các bài toán. …

Vấn đề huấn luyện mạng:

* Ảnh hướng của các tham số huấn luyện learning rate, iterator,

## Bài toán nhận dạng chữ viết tay

* Thực tiễn
* Kỹ thuật nhận dạng
* Các thách thức trong bài toán nhân dạng chữ viết tay
* NODEJS
* AngularJS, Bootstrap
* MongoDB
* Redis
* THUẬT TOÁN PHÂN LỚP (MẠNG NEURAL, SVM, RANDOM FOREST, DNN)
  + Neural sinh học và ý tưởng xây mạng neural nhân tạo
    - Tham khảo file Thelf/Report/ 2278500-xu-ly-anh-140707223606-phpapp02.pdf **p.68-p.75**
  + Neural nhận tạo và luật học
  + Mô hình và cấu trúc mạng neural nhân tạo
  + Các yếu tố ảnh hưởng đến độ chính xác của mạng neural
  + Quy trình xây dựng mạng neural
* XỬ LÝ ẢNH, TRÍCH CHỌN ĐẶC TRƯNG

# CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

## Xây dựng bộ dữ liệu sử dụng

* + CÁCH LÀM, SỐ SL, KÍCH THƯỚC HÌNH
  + XÂY DỰNG TẬP TRAIN, TẬP TEST
  + PHƯƠNG PHÁP TRÍCH CHỌN ĐẶC TRƯNG
  + KÊT QUẢ THU THẬP

## Xây dựng mạng

* + ĐẦU VÀO
  + SỐ LỚP ẨN
  + SỐ NEURAL LỚP ẨN
  + ĐẦU RA
  + CÁCH HUẤN LUYỆN học giám sat/học k giám sát/học tăng cường/
  + KIỂM TRA KẾT QUẢ NHẬN DẠNG

## Xây dựng ứng dụng

* Các yêu cầu, chức năng, các mô hình UML

Thực hiện:

* + SERVER SIDE: NODEJS, các module sử dụng, mô hình xây dựng ứng dụng MVC,
  + CLIENT SIDE: ANGULARJS, BOOTSTRAP, JQUERY, các module, mô hình MVW
  + XỬ LÝ ẢNH: OPENCV

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 

DANH MỤC TÀI LIỆU THAM KHẢO

1. Yen, Stephen. “NoSQL is a Horseless Carriage” (PDF). NorthScale. [↑](#endnote-ref-1)
2. McCulloch, Warren; Walter Pitts (1943). “A Logical Calculus of Ideas Immanent in Nervous Activity”. Bulletin of Mathematical Biophysics. [↑](#endnote-ref-2)