TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN**

**THỰC TẬP TỐT NGHIỆP**

**NGÀNH: KĨ THUẬT PHẦN MỀM**

**NGHIÊN CỨU DEEP LEARNING, ỨNG DỤNG NHẬN DẠNG CHỮ SỐ VIẾT TAY**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | Bùi Ngọc Minh  Lớp ĐH KTPM1 K11 |
| Giảng viên hướng dẫn: | TS. Nguyễn Bá Nghiễn |

***Hà Nội, 03/2020***

# LỜI NÓI ĐẦU

Ngày nay với thời đại công nghệ phát triển, AI – Artificial Intelligence và cụ thể hơn Machine Learning đang là xu thế và nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư. Trí tuệ nhân tạo đang len lỏi và từng ngõ ngách trong cuộc sống của chúng ta và trở nên quá quen thuộc như Siri của Apple, hệ thống tự nhận diện khuôn mặt của Facebook, xe tự lái, giảm thiểu tai nạn của Google..

Machine Learning là một tập con của AI. Là một lĩnh vực nhỏ của ngành Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đầu vào mà không cần phải lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của máy tính đã được nâng lên một tầm cao mới, dữ liệu ngày một lớn đã được các công ty công nghệ lớn thu thập được. Lĩnh vực mới được ra đời để phục vụ nhu cầu đó là Deep Learning. Deep Learning đã giúp máy tính thực thi những công việc tưởng chừng như không thể như phân loại các vật thể trong một bức ảnh, tự tạo chú thích cho ảnh, giả giọng người, ...

Bắt kịp theo xu hướng của thế giới hiện tại, và tại Việt Nam cũng đang rất thiếu nguồn nhân lực về ngành này nên em đã quyết định chọn đề tài “Nghiên cứu Deep Learning, ứng dụng nhân dạng chữ số viết tay” do thầy Nguyễn Bá Nghiễn hướng dẫn.

Với bài toán nhận dạng chữ số viết tay thì vấn đề trở nên phức tạp vì các lí do như không có khái niệm về kích cỡ, font chữ, cùng 1 số mỗi người viết một khác.

Nội dung bài báo cáo gồm 3 chương

* Chương 1: Tổng quan về đề tài
* Chương 2: Deep Learning
* Chương 3: Cài đặt thực nghiệm và đánh giá

# LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy, các cô khoa Công nghệ Thông tin – Trường Đại học Công Nghiệp Hà Nội đã tận tình dạy dỗ và truyền đạt cho em nhiều kiến thức quý báu trên ghế nhà trường.

Đặc biệt em xin tỏ lòng biết ơn sâu sắc đến thầy Ts. Nguyễn Bá Nghiễn đã tận tình giúp đỡ và truyền đạt nhiều kinh nghiệm về mảng Deep Learning để em có thể thực hiện và hoàn thành tốt đề tài đã chọn.

Ngoài ra em cũng xin chân thành cảm ơn các bạn lớp Kĩ Thuật Phần Mềm 1 – K11, trường Đại học Công Nghiệp Hà Nội, những đồng nghiệp ở công ty hiện đang đang đi làm đã luôn sát cánh, động viên và giúp đỡ mình rất nhiều trong quá trình thực hiện đề tài này.

Em xin chân thành cảm ơn!

**Mục lục**

[LỜI NÓI ĐẦU 2](#_Toc33383805)

[LỜI CẢM ƠN 3](#_Toc33383806)

[CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI 5](#_Toc33383807)

[1. Giới thiệu về bài toán nhận dạng chữ số viết tay 5](#_Toc33383808)

[2. Đặt vấn đề và hướng giải quyết 6](#_Toc33383809)

[CHƯƠNG II: DEEP LEARNING 7](#_Toc33383810)

[1. Features Engineering 7](#_Toc33383811)

[1.1. Binning 8](#_Toc33383812)

[1.2. One-hot encoding 8](#_Toc33383813)

[1.3. Scaling 9](#_Toc33383814)

[2. Network Architecture 9](#_Toc33383815)

[2.1. Từ Perceptron đến Neutral Network 9](#_Toc33383816)

[2.2. Convolunation Neutral Network - CNN 9](#_Toc33383817)

[3. Optimization 9](#_Toc33383818)

[CHƯƠNG III: CÀI ĐẶT THỰC NGHIỆM VÀ ĐÁNH GIÁ 10](#_Toc33383819)

# CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI

## Giới thiệu về bài toán nhận dạng chữ số viết tay

Nhận dạng chữ số viết tay là 1 dạng trong bài toán Phân loại hình ảnh – Image Classification. Đây là một trong những bài toán phân loại phổ biến nhất trong lĩnh vực Computer Vision và Deep Learning.

Bộ cơ sở dữ liệu **MNIST – Modified National Institure of Standards and Technology** là bộ cơ sở dữ liệu lớn nhất về chữ số viết tay và được sử dụng trong hầu hết các thuật toán nhận dạng hình ảnh (Image Classification). MNIST bao gồm 2 tập dữ liệu, tập dữ liệu dùng để training có 60000 các ví dụ khác nhau về chữ số viết tay từ 0 đến 9, tập dữ liệu để test thì gồm 10000 ví dụ khác nhau tương tự. Tất cả đều đã được gán nhãn (label). Bộ cơ sở dữ liệu này có thể được tải xuống từ trang chủ: <http://yann.lecun.com/exdb/mnist/>. Dưới đây là hình ảnh ví dụ về 1 số hình ảnh trong cơ sở dữ liệu MNIST.

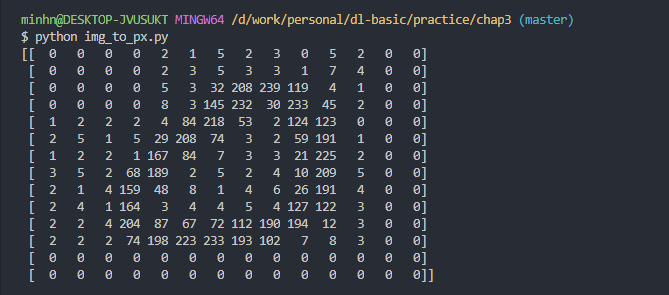


Bộ cơ sở dữ liệu chữ số viết tay MNIST

Mỗi bức ảnh là một ảnh đen trắng có kích thước 28x28 pixel (tổng 784 pixel). Mỗi pixel mang một giá trị là số từ 0 đến 255. Các pixel càng gần 0 thì càng đen, càng gần 255 thì càng trắng. Mỗi ảnh đều được chuẩn hóa về kích thước 28x28 và dạng ảnh xác (gray scale) cũng như đã được căn giữa cố định vị trí. Dưới đây là ví dụ về ảnh chữ số 0 và giá trị của các pixel trong ảnh.

D:\work\personal\dl-basic\practice\chap3\training\0\img_153.jpg

Một ví dụ về số 0 trong CSDL MNIST



Ma trận pixel của ảnh trên

Đây là tập dữ liệu rất phổ biến được sử dụng để thử nghiệm các thuật toán do tính chất **khá dễ** của nó. Có câu rằng **Thuật toán chạy tốt trên MNIST chưa chắc chạy tốt trên dataset khác nhưng không tốt trên MNIST thì đa phần không tốt trên các dataset khác**. Chính vì lý do đó nên MNIST được coi là dataset phổ biến nhất trong bài toán phân loại ảnh,

## Đặt vấn đề và hướng giải quyết

Mỗi ảnh đều là ảnh xám và có kích thước 28x28 và đều được đánh label cho từng ảnh từ 0 đến 9. Và chúng ta cần dự đoán ảnh là số mấy.

Trong các thuật toán cở bản Machine Learning thì để phân loại chúng ta có thuật toán Logistic Regression. Tuy nhiên output của thuật toán chỉ là giá trị nhị phân và là một mô hình Neutral Network đơn giản. Do đó chúng ta sẽ cần áp dụng mô hình Neutral Network cho bài toán phân loại ảnh.

Với mô hình Neutral Network thì chúng ta có các loại như Neutral Network cổ điển, CNN – Convolutional Neutral Network và RNN – Recurrent Neutral Network.

Do input của bài toán là ảnh nên chúng ta sẽ sử dụng mô hình Neutral Network cổ điển hoặc CNN.

# CHƯƠNG II: DEEP LEARNING

## Features Engineering

Feature Engineering là quá trình chuyển đổi tập dữ liệu thô ban đầu thành các thuộc tính (features) có thể giúp biểu diễn tập dữ liệu ban đầu tốt hơn, để giải quyết các bài toán dễ dàng hơn, giúp tương thích với từng mô hình dự đoán cụ thể cũng như cải thiện độ chính xác của mô hình dự đoán hiện tại.

Feature Engineering là một giai đoạn không thể thiếu trong quá trình phát triển bất kì một hệ thống thông minh nào.

Phần lớn các bài toán Machine Learning có thể được thể hiện trong hình vẽ sau:



Nguồn: machinelearningcoban.com

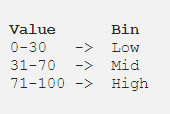
Với khối Feature Extraction (Feature Engineering) chúng ta cần phải thiết kế sao cho từ tạo ra một Feature Extractor biến dữ liệu thô ban đầu thành dữ liệu phù hợp với mục đích sử dụng.

Một số kỹ thuật thường được sử dụng:

### Binning

Là một kỹ thuật để chuyển các dữ liệu liên tục thành các nhóm dữ liệu. Thực hiện bằng cách nhóm các giá trị vào các “bin” đã được xác định trước. Các giá trị liên tục sau đó được thay thế bằng tên của “bin” đã được định nghĩa trước mà chứa giá trị đó.

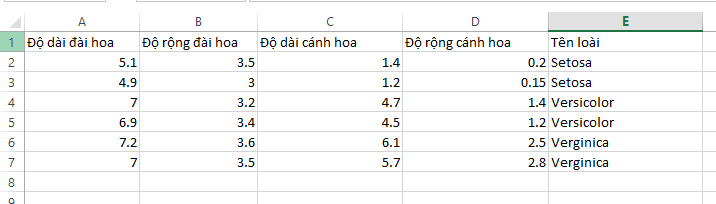
Ví dụ: Trong tập dữ liệu từ 0 -> 100. Ta chia thành 3 bin như sau



### One-hot encoding

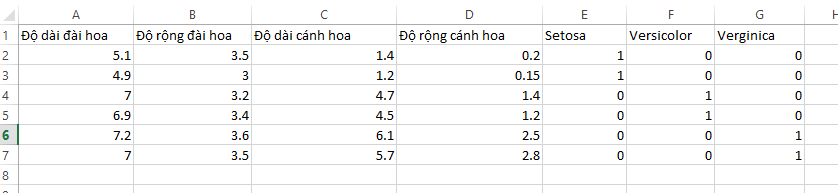
Là một trong những phương thức phổ biến nhất trong Machine Learning. Kỹ thuật này sẽ tách value của 1 cột thành nhiều cột và đánh số 0 hoặc 1 vào mỗi cột. Các giá trị nhị phân này biểu thị mối quan hệ giữa các cột đã tách ra với cột gốc.

Ví dụ: Bài toán phân loại hoa IRIS ta có 3 loài hoa là Setosa, Versicolor, Verginica



IRIS dataset

Sau khi áp dụng One-hot encoding thì chúng ta sẽ có được bảng mới như sau:



One-hot encoding IRIS

### Scaling

Khi các điểm dữ liệu chệnh lệch nhau quá lớn, một thành phần có giá trị trong khoảng 0-1, thành phần khác lại có giá trị từ 0-1000. Lúc này chúng ta cần chuẩn hóa dữ liệu trước khi thực hiện các bước tiếp theo.

Một vài phương pháp thường dùng:

* + 1. Rescaling

Phương pháp này thông thường sẽ đưa tất cả các thành phần dữ liệu về cùng 1 khoảng [0;1] hoặc [-1;1]. Ví dụ nếu muốn đưa một thành phần về khoảng [0;1] công thức sẽ là:

Trong đó:

|  |  |
| --- | --- |
| x: | Giá trị ban đầu |
| x’: | Giá trị sau khi chuẩn hóa |
| min(x), max(x): | Giá trị max và min được tính trên toàn bộ dữ liệu ở cùng một thành phần |

* + 1. Standardization

Phương pháp này giả sử mỗi thành phần đều có phân phối chuẩn với kỳ vọng là 0 và phương sai là 1. Khi đó, công thức chuẩn hóa sẽ là:

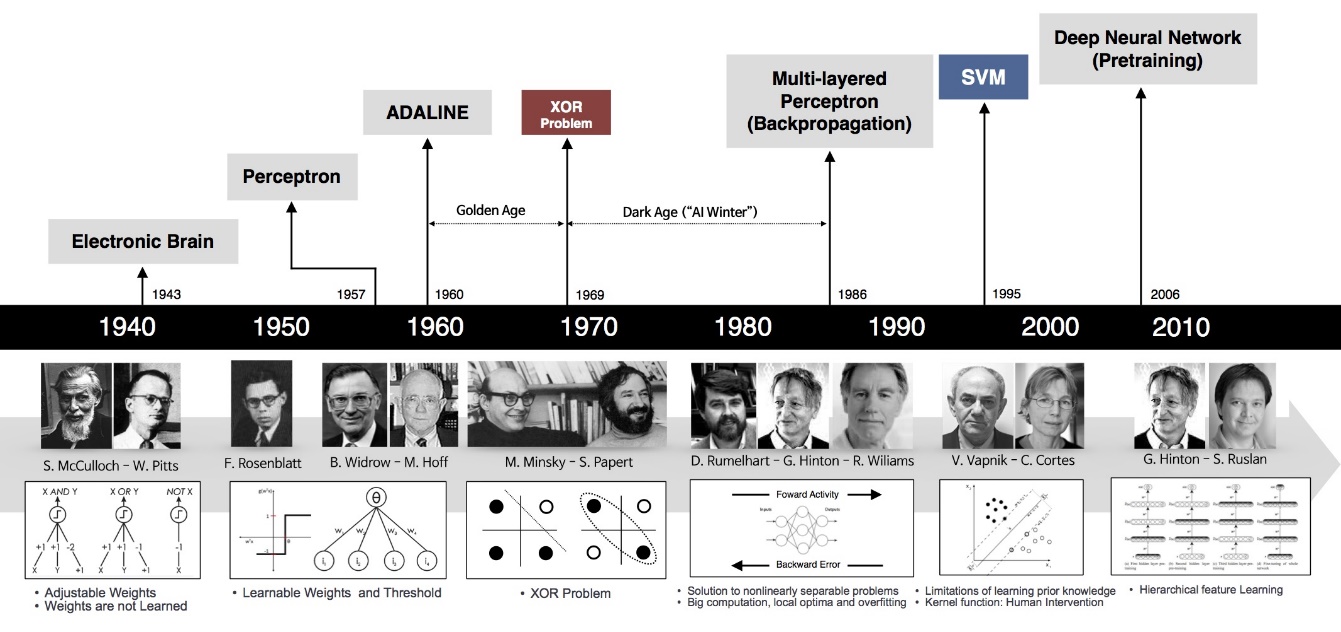
Trong đó:

|  |  |
| --- | --- |
| x: | Giá trị ban đầu |
| x': | Giá trị sau khi chuẩn hóa |
| µ: | Kỳ vọng |
| σ: | Độ lệch chuẩn |

## Network Architecture

### Từ Perceptron đến Neutral Network

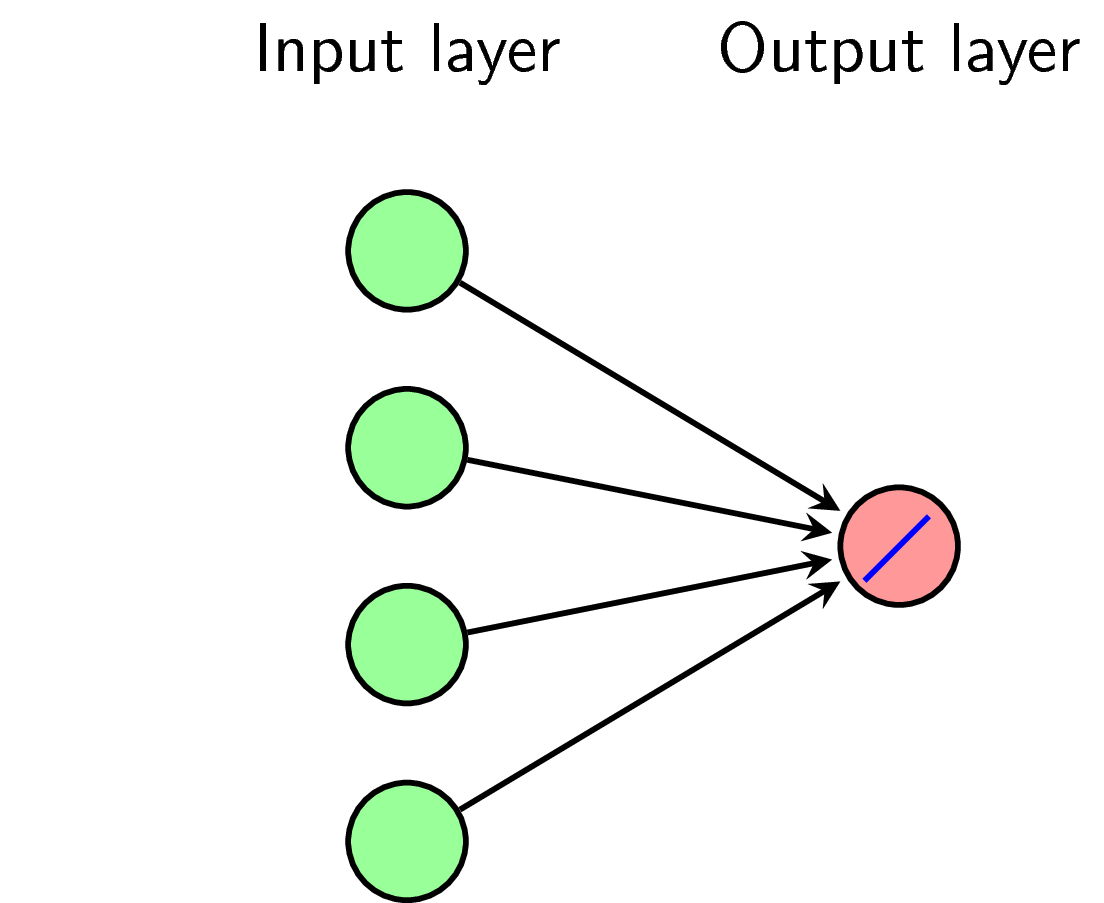
### Trong những năm gần đây Deep Learning mới được nhắc nhiều đến, tuy nhiên những nền tảng cơ bản đầu tiên đã được xuất hiện từ rất lâu. Dưới đây là lịch sử hình thành của Deep Learning



Nguồn: machinelearningcoban.com

Phiên bản sơ khai của Neutral Network là Perceptron Learning Algorithm (Perceptron) là một thuật toán supervised learning giúp giải quyết các bài toán phân lớp nhị phân.

Vào thời điểm đấy mọi người đều tin rằng thuật toán này sẽ làm được những việc tưởng chừng như không thể như có thể đi, nói chuyện, nhìn, nhận thức, tự sinh sản. Tuy nhiên vào năm 1969 Marvin Minsky và Seymour Papert đã chứng minh rằng perceptron **không thể** học được bài toán **XOR**.



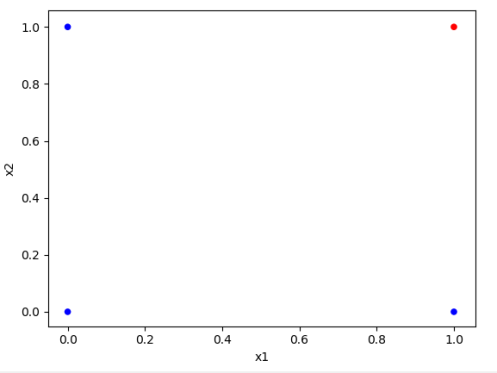
Mô hình Perceptron

Với thuật toán Perceptron chúng ta sẽ tìm ra đường thẳng chia dữ liệu thành 1 miền để cho ra kết quả.

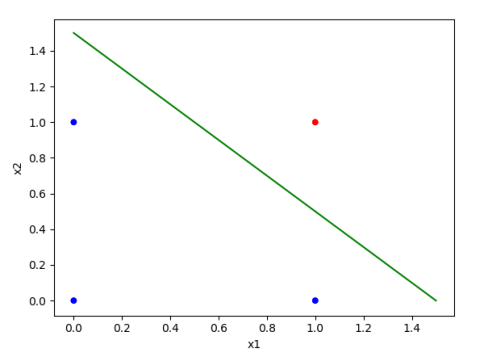
Ví dụ với bài toán AND, ta có bảng chân lý:

|  |  |  |
| --- | --- | --- |
| A | B | A AND B |
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

Biểu diễn dữ liệu dưới dạng đồ thị ta có:

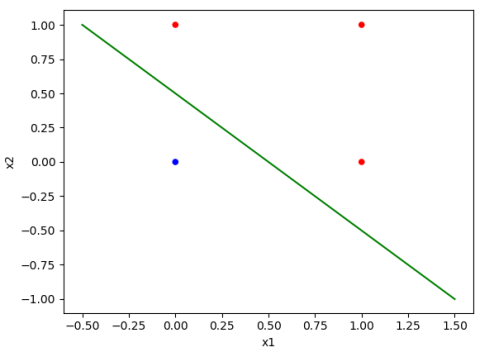


Áp dụng thuật toán Perceptron ta tìm ra được đường thẳng chia thành 2 miền dữ liệu như sau:



Tương tự với bài toán OR ta có bảng chân lý vào đường thẳng chia cắt 2 miền dữ liệu như sau:

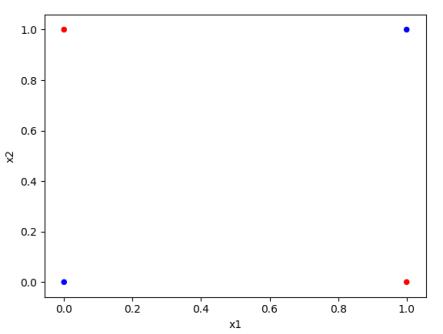
|  |  |  |
| --- | --- | --- |
| A | B | A OR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



Đối với bài toán XOR, ta có bảng chân lý:

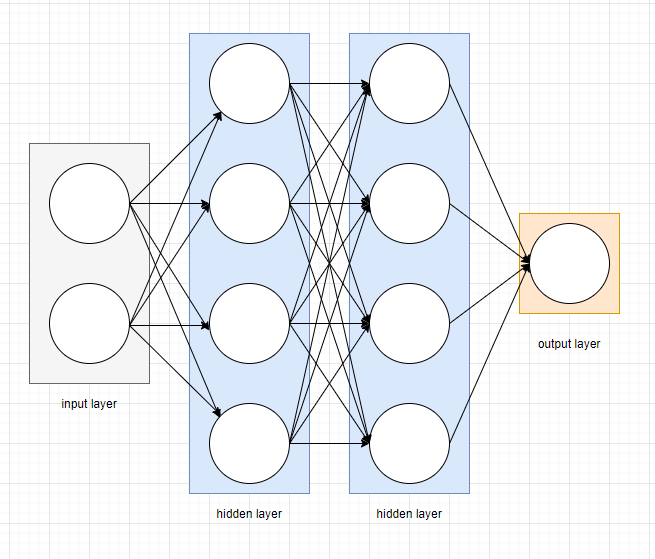
|  |  |  |
| --- | --- | --- |
| A | B | A XOR B |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

Ta có đồ thị đối với bài toán XOR:



Như vậy chúng ta có thể thấy không thể dùng 1 đường thẳng để chia dữ liệu bài toán XOR thành 2 miền được. Lúc này người ta mới cần 1 giải pháp mới đó là **Neutral Network**.

Mô hình **Neutral Network** tổng quát:



Mô hình Neutral Network

Mô hình Neutral Network với nhiều hidden layer (được gọi là multiple layer perceptron – MLP) là 1 mô hình nâng cấp của Perceptron có thể huấn luyện 1 cách hiệu quả dựa trên một quy trình đơn giản gọi là **Backpropagation (lan truyền ngược)**. Việc tính đạo hàm của các hàm số phức tạp mô tả quan hệ giữa input vào output của bài toán là rất quan trọng vì hầu hết các thuật toán tối ưu đều được thực hiện thông qua việc tính đạo hàm. Việc này giúp Neutral Network thoát khỏi những hạn chế tồn đọng của Perceptron.

Trong mô hình Neutral Network, layer đầu tiên là input layer, các layer nằm giữa là các hidden layers và layer cuối cùng là output layer. Mỗi hình tròn được gọi là node của mỗi layer.

Mỗi mô hình Neutral Network của từng bài toán luôn có 1 input layer, 1 output layer và có hoặc không các hidden layer. Tổng số layer của mô hình được quy ước là tổng số layer – 1 (không tính input layer).

Ví dụ trong hình trên thì chúng ta có 1 input layer với 2 nodes, 2 hidden layers mỗi hidden layer có 4 nodes và output layer có 1 node. Tổng số lượng layer của mô hình là 3 layer.

Chúng ta sẽ quy ước số node trong hidden layer thứ i là .

Ma trận kích thước là ma trận hệ số giữa layer (k-1) và layer k, trong đó là hệ số kết nối từ node thứ i của layer k-1 đến node thứ j của layer k.

Với node thứ i trong layer ta thực hiện 2 bước:

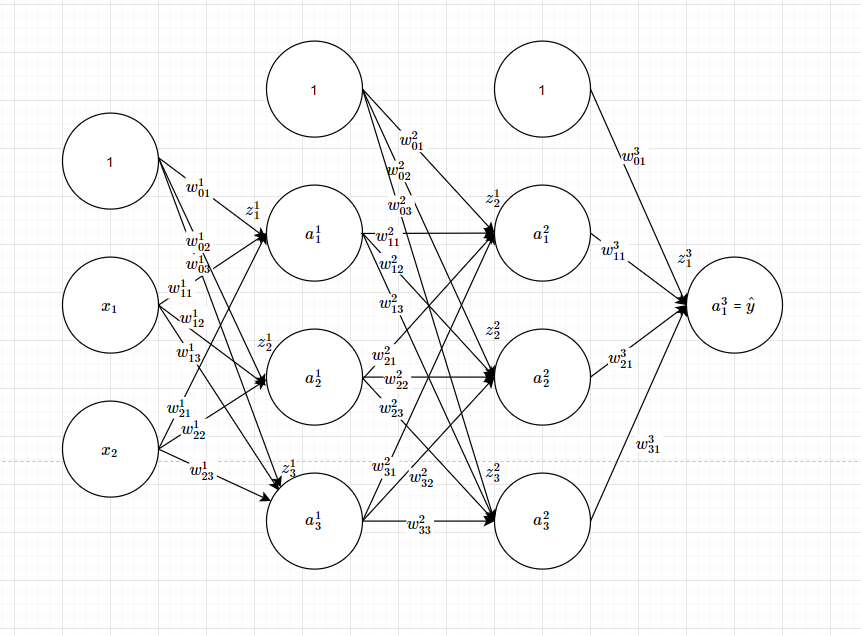
* Tính tổng linear: , là tổng tất cả các node trong layer trước đấy nhân với hệ số w tương ứng.
* Áp dụng hàm activation:

Vector có kích thước là giá trị các node của layer k sau bước tính tổng các linear

Vector có kích thước là giá trị của các node trong layer k sau bước áp dụng hàm activation cho vector

Lưu ý: tại mọi layer l

Quay trở lại bài toán XOR, giả sử chúng ta áp dụng mô hình Neutral Network cổ điển vào bài toán, gồm 2 hidden layers, mỗi layer 3 node như sau.



Mô hình Neutral Network cho bài toán XOR

Giá trị ở các node được tính như sau:

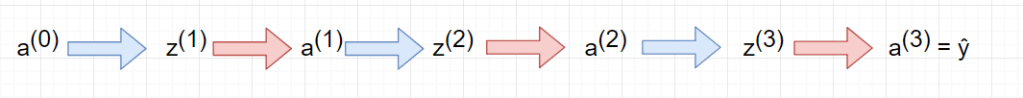
Tại node thứ nhất layer 1 ta có:

Hay tại node thứ 3 layer 2 ta có:

Tương tự tính ra được giá trị của các node còn lại.

Gọi là ma trận hệ số của input layer. Viết 1 cách ngắn gọn công thức tính ma trận hệ số a của các layer ta có:

Tóm lại chúng ta có quá trình Feedforward như sau:



Feedforward

Từ input đầu vào chúng ta có thể tính ra được giá trị dự đoán , việc cần làm ở đây chính là đi tìm hệ số W. Chúng ta có thể áp dụng thuật toán Gradient Descent, đi tìm đạo hàm của các hệ số đối với loss function. Và việc tính đạo hàm của các hệ số trong bài toán này sẽ áp dụng thuật toán Backpropagation.

**Backpropagation**

Gọi X là ma trận hệ số input của bài toán XOR và Y là ma trận hệ số output. Ta có 2 ma trận X, Y như sau:

**Độ chính xác của phương trình (Loss function)**

Với mỗi điểm ta có hàm mất mát:

Hàm mất mát của thuật toán Neutral Network tổng quát trên toàn bộ dữ liệu sẽ như sau:

**Gradient Descent**

Để tìm điểm cực tiểu của hàm mất mát, ta có thể dùng thuật toán Gradient Descent để tìm đạo hàm của các hệ số W với hàm mất mát. Nhưng hàm mất mát lại quá phức tạp khiến việc tìm đạo hàm riêng với từng là rất khó. Thuật toán Backpropagation giúp chúng ta dễ dàng tìm được các đạo hàm riêng này.

Với mỗi điểm ta có hàm mất mát:

trong đó là giá trị mà model dự đoán còn là giá trị thực tế của dữ liệu.

Đạo hàm riêng của giá trị dự đoán trên hàm mất mát:

Tính đạo hàm riêng của , áp dụng chain rule chúng ta có:

Do đó :

Tương tự chúng ta có đạo hàm riêng của các :

Đạo hàm riêng của và theo L, áp dụng chain rule ta có công thức tổng quát như sau:

Một lưu ý khi sử dụng Gradient Descent với Neutral Network đó là ta không nên khởi tạo tất cả các giá trị vì nếu làm vậy sẽ làm giá trị các nút trong cùng 1 lớp khác lớp input sẽ bị giống nhau, dẫn đến việc thuật toán không thể kết thúc được. Thay vào đó chúng ta nên khởi tạo là các giá trị ngẫu nhiên, thông thường sẽ khởi tạo giá trị ngẫu nhiên nằm trong vùng lân cận của 0.

Mô hình chung thuật toán Neutral Network:

* Chọn các giá trị ngẫu nhiên và giá trị learning rate α
* Liên tiếp lặp các phép biến đổi

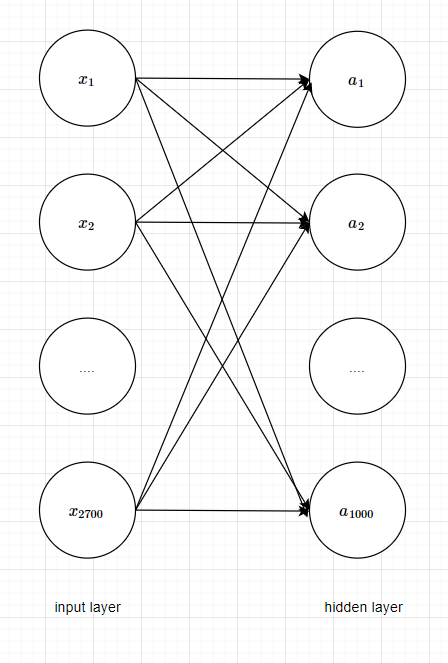
với các đạo hàm riêng được tính bằng thuật toán Backpropagation.

* Thuật toán dừng lại khi giá trị hàm mất mát thay đổi rất nhỏ hoặc trị tuyệt đối các đạo hàm riêng rất nhỏ. Nếu thuật toán không thể kết thúc thì chọn lại giá trị α sao cho hợp lý và chạy lại bước trên.

### Convolunation Neutral Network - CNN

Trong mô hình Neutral Network cổ điển, mỗi hidden layer là một fully connected layer, nghĩa là mỗi node được kết nối với tất cả các node trong layer trước đấy. Cả mô hình thì được gọi là fully connected neutral network (FCN).

Lấy ví dụ với bài toán phân loại ảnh mỗi ảnh có kích thước là 30x30 pixel. Nếu để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel của ảnh với tổng số node là 30x30x3 = 2700 nodes.



Giả sử ta khởi tạo số lượng nodes trong hidden layer là 1000 nodes. Khi đấy số lượng W được sinh ra là 2701\*1000=2701000 weights. Số lượng tham số rất lớn, nếu đối với bài toán xử lý ảnh 256x256 khi đấy số lượng tham số được sinh ra là rất lớn. Khi đó chúng ta cần phải pháp tốt hơn đó là mô hình Convolutional Neutral Network (CNN).

Trong 1 bức ảnh thì các pixel ở gần nhau sẽ có mối liên hệ với nhau hơn là các pixel ở cách xa nhau. Nếu chúng ta sử dụng mô hình Neutral Network cổ điển thì khi đấy sẽ bị mất đi tính chất này của ảnh.

## Optimization

# CHƯƠNG III: CÀI ĐẶT THỰC NGHIỆM VÀ ĐÁNH GIÁ