**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**TÊN ĐỀ TÀI***: Bài toán dự báo doanh số bán hàng của doanh nghiệp Vinamilk sử dụng mạng Neural*

Giáo viên hướng dẫn**: PGS. TS. NGUYỄN TẤN KHÔI**

Sinh viên thực hiện**:**

Sinh viên 1: **NGUYỄN VĂN DŨNG** MSSV:**102210356**

Sinh viên 2: **HOÀNG NGUYỄN NHẬT MINH** MSSV:**102210361**

Lớp: **21TCLC\_NHAT2** Nhóm : **21N99**

**Đà Nẵng, 07/2022**

MỤC LỤC

[MỞ ĐẦU 1](#_Toc108030106)

[1. TỔNG QUAN ĐỀ TÀI 2](#_Toc108030107)

[1.1. Khái Niệm. 2](#_Toc108030108)

[1.2. Quá trình training của mạng nơron. 3](#_Toc108030109)

[2. CƠ SỞ LÝ THUYẾT 4](#_Toc108030110)

[2.1. Ý tưởng. 4](#_Toc108030111)

[2.2. Cơ sở lý thuyết. 5](#_Toc108030112)

[2.2.1. Các thành phần mạng thần kinh. 5](#_Toc108030113)

[2.2.2. Neural network operation propagation. 11](#_Toc108030114)

[2.2.3. Backprogation và Gradient descent và khái niệm Learning rate 12](#_Toc108030115)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 16](#_Toc108030116)

[3.1. Phát biểu bài toán 16](#_Toc108030117)

[3.2. Cấu trúc dữ liệu 19](#_Toc108030118)

[3.3. Thuật toán 20](#_Toc108030119)

[3.3.1. Một số hàm hỗ trợ 21](#_Toc108030120)

[3.3.2. Hàm chính 29](#_Toc108030121)

[3.3.3. Độ phức tạp của thuật toán 31](#_Toc108030122)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 31](#_Toc108030123)

[4.1. Tổ chức chương trình 31](#_Toc108030124)

[4.2. Ngôn ngữ cài đặt 32](#_Toc108030125)

[4.3. Kết quả thực hiện 32](#_Toc108030126)

[4.3.1. Giao diện chính của chương trình 32](#_Toc108030127)

[4.3.2. Các kết quả thực thi của chương trình 35](#_Toc108030128)

[4.3.3. Nhận xét đánh giá kết quả 36](#_Toc108030129)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 36](#_Toc108030130)

[5.1. Kết luận 36](#_Toc108030131)

[5.2. Hướng phát triển 38](#_Toc108030132)

[TÀI LIỆU THAM KHẢO 39](#_Toc108030133)

[CODE 40](#_Toc108030134)

**DANH MỤC HÌNH VẼ**

[Hình 1.1.1: Nơron nhân tạo. 2](#_Toc107957567)

[Hình 2.1.1: Mạng nơron sinh học. 4](#_Toc107957568)

[Hình 2.2.1: Cấu trúc mạng nơron nhân tạo. 6](#_Toc107957569)

[Hình 2.2.2: Tính đầu ra của nơron. 6](#_Toc107957570)

[Hình 2.2.3: Đồ thị hàm Sigmoid. 8](#_Toc107957571)

[Hình 2.2.4: Đồ thị hàm Tand. 9](#_Toc107957572)

[Hình 2.2.5: Đồ thị hàm Relu. 9](#_Toc107957573)

[Hình 2.2.6: Fully - connected neural networks. 10](#_Toc107957574)

[Hình 2.2.7: Pooling neural networks. 11](#_Toc107957575)

[Hình 2.2.8: Backpropagation. 12](#_Toc107957576)

[Hình 2.2.9: Biểu đồ giảm Loss xuống đến cực trị. 13](#_Toc107957577)

[Hình 3.1.1: Input đầu vào cho chương trình training. 16](#_Toc107957578)

[Hình 3.1.2: Output đầu ra cho chương trình training. 17](#_Toc107957579)

[Hình 3.1.3: Input đầu vào cho quá trình dự đoán. 18](#_Toc107957580)

[Hình 3.1.4: Input đầu vào cho quá trình dự đoán. 18](#_Toc107957581)

[Hình 3.1.5: Output đầu ra cho quá trình dự đoán. 19](#_Toc107957582)

[Hình 3.2.1: Khai báo thư viện. 19](#_Toc107957583)

[Hình 3.2.2: Khai báo kiểu dữ liệu. 19](#_Toc107957584)

[Hình 3.3.1: Hàm sigmoid. 22](#_Toc107957585)

[Hình 3.3.2: Hàm đạo hàm của sigmoid. 22](#_Toc107957586)

[Hình 3.3.3: Hàm input file. 23](#_Toc107957587)

[Hình 3.3.4: Hàm proccess data() 24](#_Toc107957588)

[Hình 3.3.5: Hàm random weight. 25](#_Toc107957589)

[Hình 3.3.6: Hàm take input. 26](#_Toc107957590)

[Hình 3.3.7: Hàm forward proga. 27](#_Toc107957591)

[Hình 3.3.8: Hàm update weight. 28](#_Toc107957592)

[Hình 3.3.9: Hàm after proga. 29](#_Toc107957593)

[Hình 3.3.10: Hàm train. 30](#_Toc107957594)

[Hình 3.3.11: Hàm proccess\_1. 30](#_Toc107957595)

[Hình 3.3.12: Hàm proccess\_2. 31](#_Toc107957596)

[Hình 4.3.1: Giao diện của chương trình. 33](#_Toc107957597)

[Hình 4.3.2: Chọn lựa chọn và nhập file chứa dữ liệu. 33](#_Toc107957598)

[Hình 4.3.3: Ví dụ về file dữ liệu đưa vào. 34](#_Toc107957599)

[Hình 4.3.4: Kiểm tra bảng dữ liệu đưa vào đã đúng chưa. 34](#_Toc107957600)

[Hình 4.3.5: Kết quả dự đoán. 35](#_Toc107957601)

[Hình 4.3.6: File dữ liệu thu được sau quá trình training. 35](#_Toc107957602)

[Hình 4.3.7: Kết quả của việc dự đoán doanh thu thu được. 36](#_Toc107957603)

[Hình 5.1.1: Kết quả thu được. 38](#_Toc107957604)

**LỜI CẢM ƠN**

Lời đầu tiên, tụi em xin gửi lời cảm ơn chân thành đến Trường Đại học Bách Khoa- Đại học Đà Nẵng đã cho tụi em một môi trường học tập thật tốt và bổ ích. Tụi em cũng xin gửi lời cảm ơn các thầy cô trong khoa Công Nghệ Thông Tin đã đem môn học PBL này vào chương trình dạy học để giúp chúng em có thể tiếp xúc với các đồ án nhằm học hỏi được các kinh nghiệm cần thiết cho công việc sau này.

Đặc biệt, tụi em cũng xin chân thành cảm ơn thầy - **PGS. TS. Nguyễn Tấn Khôi** đã tận tình hướng dẫn chỉ bảo tụi em cách làm trong suốt quá trình thực hiện đồ án để tụi em có thể hoàn thành nó một cách tốt nhất.

Trong quá trình thực hiện đề tài này, tụi em sẽ không tránh khỏi những sai sót không đáng có do chưa có nhiều kinh nghiệm cũng như vốn kiến thức còn chưa đủ nên rất mong thầy (cô) có thể góp ý giúp cho tụi em để tụi em có thể sửa đổi và thực hiện tốt hơn cho các đồ án sắp tới.

Tụi em xin chân thành cảm ơn!

*Đà nẵng, ngày 22 tháng 6 năm 2022.*

*Sinh viên 1:* *Sinh viên 2:*

Nguyễn Văn Dũng Hoàng Nguyễn Nhật Minh

MỞ ĐẦU

1. **Tính cấp thiết của bài toán**

* Trong thời đại ngày càng phát triển và tiên tiến như hiện nay thì, AI là một trong những công cụ giúp hỗ trỡ cho con người có một bức tiến cao hơn và đột phá hơn trong mọi lĩnh vực phát triển của đời sống. AI đang đóng một vai trò không thể thiếu trong các ngành nghề hiện nay bao gồm cả thương mại.
* Để đạt được lợi thế cạnh tranh thương mại trong môi trường kinh doanh thay đổi liên tục, việc dự báo nhu cầu là rất quan trọng cho một tổ chức để đưa ra quyết định đúng về sản xuất và quản lý hàng tồn kho. Trong quá trình cạnh tranh này thì, Mạng Neural được áp dụng vào để dự báo doanh số bán hàng qua từng tháng giúp cho doanh nghiệp có thể vững chắc phát triển trong thị trường.

1. **Mục tiêu nghiên cứu**

* Mục tiêu của nghiên cứu này là để đề xuất một kỹ thuật dự báo được mô hình hóa bằng các phương pháp thông minh nhân tạo sử dụng mạng neural nhân tạo. Hiệu quả của phương pháp đề xuất mang lại là một giải pháp tối ưu giúp cho doanh nghiệp có thể quản lý và so sánh doanh thu của công ty mình qua các tháng.
* Phạm vi nghiên cứu và đối tượng được nghiên cứu ở đây là doanh số bán hàng của thương hiệu sữa tươi Vinamilk qua các tháng.

1. **Phương pháp nghiên cứu**

* Phương pháp thu thập tài liệu hướng dẫn
* Phương pháp phân tích lý thuyết, ứng dụng vào ví dụ thực tế.
* Phương pháp Phân tích, thống kê từ nguồn dữ liệu thu được
* Phương pháp tư vấn, góp ý từ thầy giáo.
* Sử dụng cấu tạo của mạng neural để tìm ra quy luật và tính toán dự báo doanh thu của doanh nghiệp trong một tháng nhất định thông qua các dữ liệu được cung cấp trước đó.

# TỔNG QUAN ĐỀ TÀI

## Khái Niệm.

Mạng nơron nhân tạo, Artificial Neural Network (ANN) gọi tắt là mạng nơron, neural network, là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơron sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (gọi là phần tử xử lý hay nơron) kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó.

Một nơron là một đơn vị xử lý thông tin và là thành phần cơ bản của một mạng nơron.

Cấu trúc của một nơron được mô tả trên hình dưới:

Hình 1.1: Nơron nhân tạo


Hình 1.1.1: Nơron nhân tạo.

Các thành phần cơ bản của một nơron nhân tạo bao gồm:

* Tập các đầu vào: Là các tín hiệu vào (input signals) của nơron, các tín hiệu này thường được đưa vào dưới dạng một vec-tơ m chiều.
* Tập các liên kết: Mỗi liên kết được thể hiện bởi một trọng số (gọi là trọng số liên kết – Synaptic weight). Trọng số liên kết giữa tín hiệu vào thứ j với nơron k thường được kí hiệu là wjk. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.
* Bộ tổng (Summing function): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
* Ngưỡng (còn gọi là một độ lệch - bias): Ngưỡng này thường được đưa vào như một thành phần của hàm truyền.
* Hàm truyền (Transfer function) – còn gọi là Hàm kích hoạt (Activation function): Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng đã cho. Thông thường, phạm vi đầu ra của mỗi nơron được giới hạn trong đoạn [0,1] hoặc [-1, 1]. Việc lựa chọn hàm truyền nào là tuỳ thuộc vào từng bài toán và kinh nghiệm của người thiết kế mạng.
* Đầu ra: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.

Như vậy tương tự như nơron sinh học, nơron nhân tạo cũng nhận các tín hiệu đầu vào, xử lý (nhân các tín hiệu này với trọng số liên kết, tính tổng các tích thu được rồi gửi kết quả tới hàm truyền), và cho một tín hiệu đầu ra (là kết quả của hàm truyền).

## Quá trình training của mạng nơron.

Học là quá trình cập nhật trọng số sao cho giá trị hàm lỗi là nhỏ nhất. Một mạng nơron được huấn luyện sao cho với một tập các vec-tơ đầu vào X, mạng có khả năng tạo ra tập các vec-tơ đầu ra mong muốn Y của nó. Tập X được sử dụng cho huấn luyện mạng được gọi là tập huấn luyện (training set). Các phần tử x thuộc X được gọi là các mẫu huấn luyện (training example). Quá trình huấn luyện bản chất là sự thay đổi các trọng số liên kết của mạng. Trong quá trình này, các trọng số của mạng sẽ hội tụ dần tới các giá trị sao cho với mỗi vec-tơ đầu vào x từ tập huấn luyện, mạng sẽ cho ra vec-tơ đầu ra y như mong muốn.

Có ba phương pháp học phổ biến là học có giám sát (supervised learning), học không giám sát (unsupervised learning) và học tăng cường (Reinforcement learning).

**Học có giám sát trong các mạng nơron:**

Học có giám sát trong các mạng nơron thường được thực hiện theo các bước sau:

* Bước 1: Xây dựng cấu trúc thích hợp cho mạng nơron, chẳng hạn có n nơron vào, m nơron đầu ra, và khởi tạo các trọng số liên kết của mạng.
* Bước 2: Đưa một vec-tơ x trong tập mẫu huấn luyện X vào mạng.
* Bước 3: Tính vec-tơ đầu ra z của mạng.
* Bước 4: So sánh vec-tơ đầu ra mong muốn t (là kết quả được cho trong tập huấn luyện) với vec-tơ đầu ra z do mạng tạo ra; nếu có thể thì đánh giá lỗi.
* Bước 5: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vec-tơ x vào mạng, vec-tơ đầu ra z sẽ giống với t hơn.
* Bước 6: Nếu cần, lặp lại các bước từ 2 đến 5 cho tới khi mạng đạt tới trạng thái hội tụ. Việc đánh giá lỗi có thể thực hiện theo nhiều cách, cách dùng nhiều nhất là sử dụng lỗi tức thời: Err = (z - t), hoặc Err = |z - t| ; lỗi trung bình bình phương (MSE: meansquare error): Err = (z- t)2 /2.

Thuật toán tổng quát ở trên cho học có giám sát trong các mạng nơron có nhiều cài đặt khác nhau, sự khác nhau chủ yếu là cách các trọng số liên kết được thay đổi trong suốt thời gian học. Trong đó tiêu biểu nhất là thuật toán lan truyền ngược sai số.

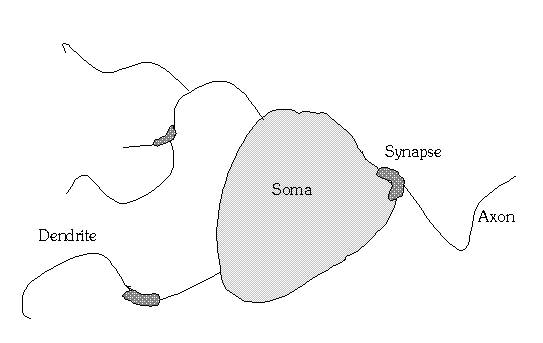
# CƠ SỞ LÝ THUYẾT

## Ý tưởng.

Mạng neural nhân tạo 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 những tình huống phù hợp.

Sau đây là những thành phần chính trong cấu trúc của một nơron trong bộ não con người:



Hình 2.1.1: Mạng nơron sinh học.

Trong đó:

* Các Soma là thân của nơron.
* Các dendrites là các dây mảnh, dài, gắn liền với soma, chúng truyền dữ liệu (dưới dạng xung điện thế) đến cho soma xử lý. Bên trong soma các dữ liệu đó được tổng hợp lại. Có thể xem gần đúng sự tổng hợp ấy như là một phép lấy tổng tất cả các dữ liệu mà nơron nhận được.
* Một loại dây dẫn tín hiệu khác cũng gắn với soma là các axon. Khác với dendrites, axons có khả năng phát các xung điện thế, chúng là các dây dẫn tín hiệu từ nơron đi các nơi khác. Chỉ khi nào điện thế trong soma vượt quá một giá trị ngưỡng nào đó (threshold) thì axon mới phát một xung điện thế, còn nếu không thì nó ở trạng thái nghỉ.
* Axon nối với các dendrites của các neural khác thông qua những mối nối đặc biệt gọi là synapse. Khi điện thế của synapse tăng lên do các xung phát ra từ axon thì synapse sẽ nhả ra một số chất hoá học (neurotransmitters); các chất này mở "cửa" trên dendrites để cho các ions truyền qua. Chính dòng ions này làm thay đổi điện thế trên dendrites, tạo ra các xung dữ liệu lan truyền tới các nơron khác.

Có thể tóm tắt hoạt động của một nơron như sau: nơron lấy tổng tất cả các điện thế vào mà nó nhận được, và phát ra một xung điện thế nếu tổng ấy lớn hơn một ngưỡng nào đó. Các nơron nối với nhau ở các synapses. Synapse được gọi là mạnh khi nó cho phép truyền dẫn dễ dàng tín hiệu qua các nơron khác. Ngược lại, một synapse yếu sẽ truyền dẫn tín hiệu rất khó khăn.

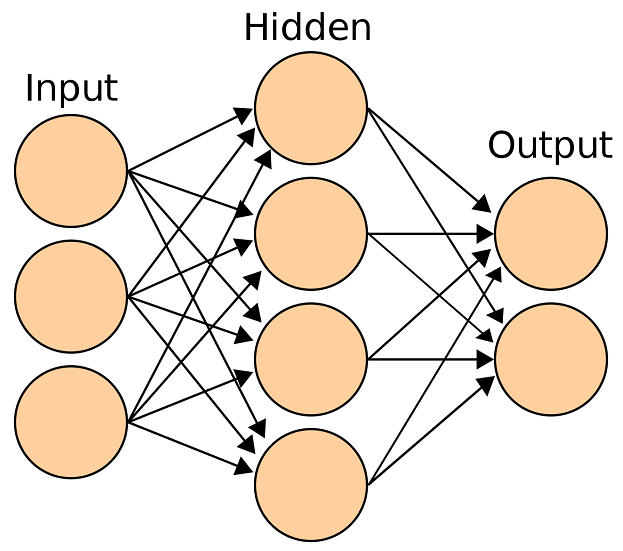
Như ta có thể thấy các thành phần như nơron, trọng số, bộ tổng, ngưỡng(hay còn gọi là độ lệch – bias), hàm truyền đều lấy ý tưởng từ các thành phần trong mạng nơron sinh học trong bộ não con người.

## Cơ sở lý thuyết.

### Các thành phần mạng thần kinh.

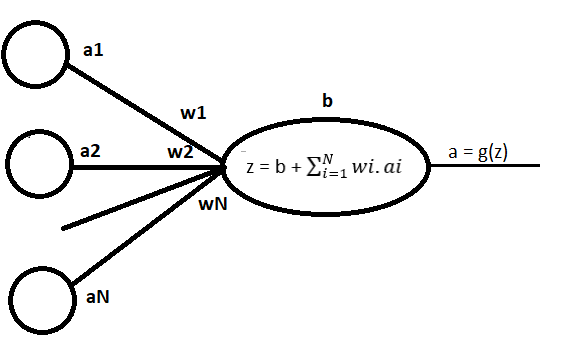
1. **Nơron.**

* Mỗi một nơron nhân tạo có nhiều đầu vào khác nhau và một đầu ra duy nhất có thể được gửi đến nhiều nơron tiếp theo. Đầu vào của một nơron có thể là dữ liệu bên ngoài( hình ảnh, tài liệu), hoặc đầu ra của các nơron khác. Đầu ra của các tế bào thần kinh cuối cùng đưa ra kết quả chính xác nhất sau quá trình training mạng, chẳng hạn như nhận dạng một đối tượng trong hình ảnh( ví dụ: nhận dạng mèo).

**

Hình 2.2.1: Cấu trúc mạng nơron nhân tạo.

Tính toán đầu ra của nơron:

****

Hình 2.2.2: Tính đầu ra của nơron.

Tính toán tổng trọng số của tất cả các đầu vào với a1,a2,...aN; w1,w2,...wN là các tham số đưa vào mạng.

* Thêm độ lệch – bias b vào tổng với b là tham số được đưa vào mạng.
* Chuyển tổng đến một hàm kích hoạt để xử lí đầu ra: a = g(z) với g(z) là activation function( hàm kích hoạt).

1. **Activation function(hàm kích hoạt).**

Trong một mạng nơron nhân tạo, activation function(hàm kích hoạt) đóng vai trò là thành phần phi tuyến tại output của các nơron. Nếu không có hàm kích hoạt phi tuyến, thì mạng nơron của chúng ta dù có nhiều lớp vẫn sẽ có hiệu quả như một lớp tuyến tính mà thôi. Để hiểu rõ hơn và trực quan hơn, ta sẽ xét một mạng nơron nhỏ gồm 2 lớp( có hàm kích hoạt tuyến tính tương ứng). Mạng này sẽ nhận vào input là X và trả ra output là Y:

Trong lớp thứ nhất, ta có trọng số , hệ số bias . Output được tính như sau:

Sau đó, output được đẩy qua hàm kích hoạt tuyến tính thu được kết quả là đầu ra của lớp thứ nhất.

Tương tự như vậy output lại trở thành đầu vào của lớp thứ 2:

Vì hàm kích hoạt là tuyến tính nên = c \* (c là một số thực).

Để cho đơn giản ta giả sử c = 1. Sử dụng công thức trên ta có:

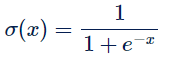
Công thức này có thể viết dưới dạng:

a[2] thực chất vẫn chỉ là một hàm tuyến tính của X ban đầu. Vì thế việc xếp chồng các lớp nơron lên nhau là vô nghĩa.

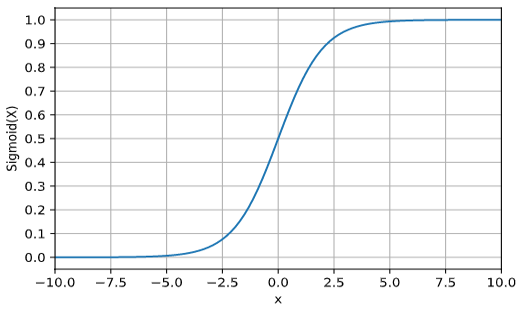
Dưới đây là một số hàm kích hoạt thường dùng cho mạng nơron nhân tạo.

* **Hàm Sigmoid**

Công thức:



Phân tích:



Hình 2.2.3: Đồ thị hàm Sigmoid.

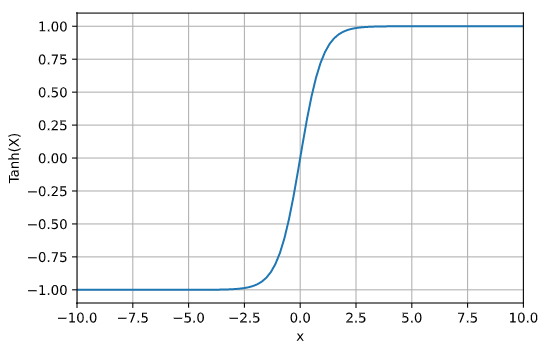
Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng(0,1)(xem đồ thị phía trên). Đầu vào là một số thực âm rất nhỏ sẽ cho đầu ra tiệm cận tới 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Ngoài ra hàm Sigmoid còn có đạo hàm rất đẹp nên được ứng dụng nhiều trong các bài toán.

* **Hàm Tand**

Công thức:



Phân tích:



Hình 2.2.4: Đồ thị hàm Tand.

Hàm Tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng(-1,1) (xem đồ thị phía trên). Đầu vào là một số thực âm rất nhỏ sẽ cho đầu ra tiệm cận tới -1, ngược lại , nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1.

Hàm Tand còn có thể biểu diễn bằng hàm Sigmoid như sau:

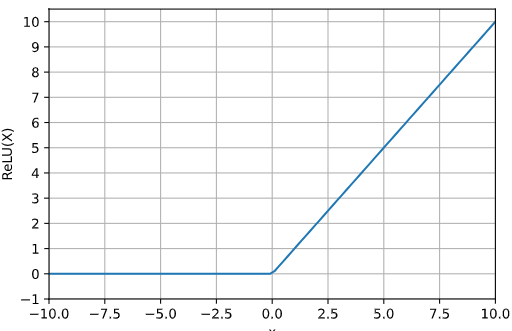


* **Hàm Relu**

Công thức:



Phân tích:



Hình 2.2.5: Đồ thị hàm Relu.

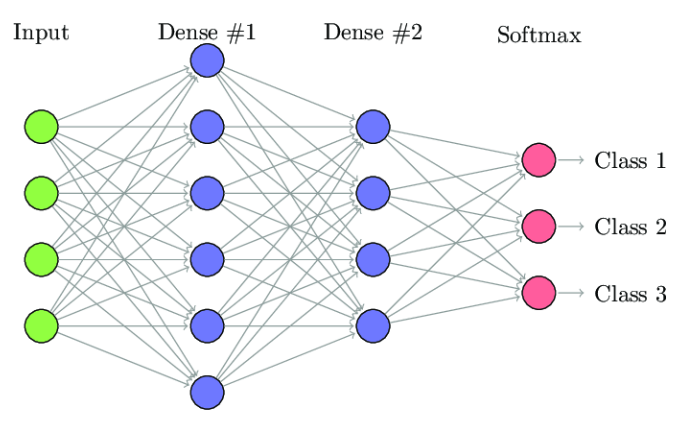
Hàm Relu lọc các giá trị < 0 và trả về giá trị đầu ra là các số thực lớn hơn hoặc bằng 0. Hàm Relu có tốc độ tính toán nhanh và tốc độ hội tụ nhanh hơn so với Sigmoid và Tanh.

1. **Tổ chức mạng nơron.**

Tổ chức mạng nơron có nhiều lớp. Các tế bào thần kinh của một lớp chỉ kết nối với các tế bào thần kinh của các lớp ngay trước và ngay sau lớp đó. Lớp nhận dữ liệu đầu vào là lớp input. Lớp tạo ra kết quả cuối cùng là lớp output. Giữa lớp input và lớp output có thể có không lớp ẩn hoặc nhiều lớp ẩn tùy vào từng điều kiện bài toán đưa ra. Mạng nơron cũng có thể có duy nhất một lớp vừa là lớp input vừa là lớp output và không có lớp ẩn.

Giữa các lớp có nhiều cách kết nối khác nhau:

* Fully connected: Mỗi nơron trong một lớp kết nối với mọi nơron trong lớp tiếp theo.



Hình 2.2.6: Fully - connected neural networks.

* Pooling: Một nhóm nơron thần kinh trong một lớp kết nối với một nơron duy nhất ở lớp tiếp theo( làm giảm số lượng nơron thần kinh ở trong lớp tiếp theo đó).

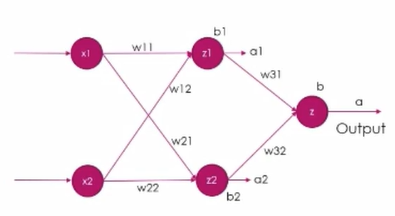


Hình 2.2.7: Pooling neural networks.

### Neural network operation propagation.

1. **Propagation.**

Là quá trình tính toán đầu ra trên mạng nơron từ đầu vào cho trước của mạng.



*Hình 2. 2.2 Propagation.*

* z1 = w11 \* x1 + w12 \* x2 + b1.
* a1 = = .
* Z2 = w21 \* x1 + w22 \* x2 + b2.
* A2 = = .
* Z = w31 \* a1 + w32 \* a2 + b.
* A = =
* A là output.

Để quá trình propagation được diễn ra chúng ta phải khởi tạo các tham số như trọng số weights và độ lệch – bias.

1. **Cost function.**

Cost function cho chúng ta biết kết quả đầu ra a và kết quả mục tiêu y của một ví dụ chúng chênh lệch như thế nào.

Chúng ta chọn hàm sau đây là Cost function với độ hữu ích cao và nó cũng sẽ trở nên đơn giản trong một số trường hợp đặc biệt.

*L(a,y) = (y + (1 y)*

1. **Loss function.**

Loss function là trung bình cộng của các cost trên toàn bộ ví dụ đưa vào. Nó phụ thuộc vào trọng số và độ lệch – bias.

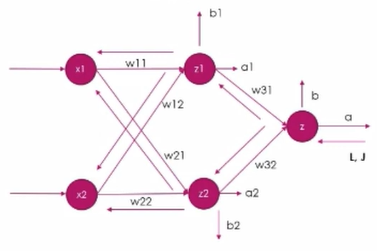
*J(weights, biasses) = .*

Mục tiêu của quá trình training mạng nơron là giảm giá trị của Loss sau từng training sao cho giá trị của Loss giảm xuống mức tối ưu nhất có thể bằng cách thay đổi trọng số và độ lệch – bias.

### Backprogation và Gradient descent và khái niệm Learning rate

Nhiệm vụ của Backpropagation là giảm giá trị của Loss sau mỗi lần training tiếp theo bằng cách điều chỉnh các hệ số weights và độ lệch – bias sao cho giá trị của Loss sẽ giảm xuống mức tối ưu nhất có thể.

Để giảm các giá trị của Loss và Cost thì chúng ta phải đi tìm đạo hàm từng phần của Loss hoặc của Cost trên mỗi tham số weight và trên bias, sau đó điều chỉnh các giá trị của weights và bias để hướng giá trị của Loss về cực trị của hàm Loss Function( tức là đưa giá trị của Loss về mức tối thiểu nhất có thể).

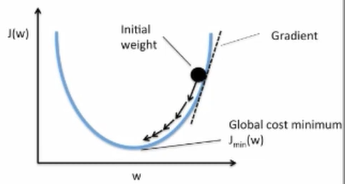


Hình 2.2.8: Backpropagation.

* w,b trong công thức trên đại diện cho tất cả weights và biases(w11, w12, b1, w21, w22, b2, w31, w32, b).

Quá trình thay đổi các trọng số weight và bias để giá trị của Loss giảm xuống đến cực trị gọi là quá trình Gradient descent.

Sau khi tính được đạo hàm từng phần của Loss trên mỗi tham số weight và bias thì chúng ta tiến hành thay đổi giá trị của weight và bias theo công thức sau:



Hình 2.2.9: Biểu đồ giảm Loss xuống đến cực trị.

là một số khá nhỏ so với 1 được gọi là Learning rate. Nếu như càng lớn thì độ giảm của Loss càng lớn, ngược lại nếu như càng nhỏ thì bước tiến giảm của Loss sẽ chậm hơn. Với việc chọn nhỏ sẽ đưa Loss về cực trị một cách chính xác hơn.

Bây giờ chúng ta sẽ đi thực hiện quá trình Gradient descent trên từng ví dụ.

Xét cấu trúc mạng nơron như Hình 2.2.2.

Ta có: Cost function: *L(a,y) = (y + (1 y)*

Theo quy tắc chuỗi của đạo hàm ta có:

Tính ta thấy:

Để tính được và ta có các công thức sau:

Tính đạo hàm riêng của z theo w31, w32, b:

And:

And:

Tại một thời điểm nào đó chúng ta sẽ tính toán được gradient cho toàn bộ trọng số weight và độ lệch bias như bảng sau:

|  |  |
| --- | --- |
| **Gradient** | **Value** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Chúng ta đã tính được gradient cho một ví dụ. Đối với tất cả các ví dụ chúng ta cần tính toán gradient của tất cả chúng và sau đó tính giá trị trung bình

Bây giờ chúng ta áp dụng gradient descent cho tất cả các weights và độ lệch bias:

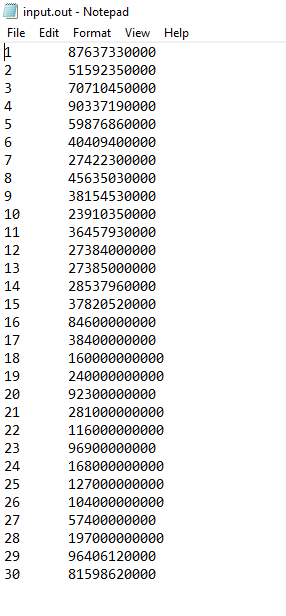
Kết thúc Backpropagation.

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

* Quá Trình Training:
  + Input: Một file chứa dữ liệu doanh thu của công ty theo dạng:

STT( số thứ tự ) Doanh thu



Hình .: Input đầu vào cho chương trình training.

* + Ouput: đưa ra là một file chứa trọng số (weight và bias) theo dạng:

W[1][1] W[1][2] ………. W[1][nn] B[1]

W[2][1] W[2][2] ………. W[2][nn] B[2].

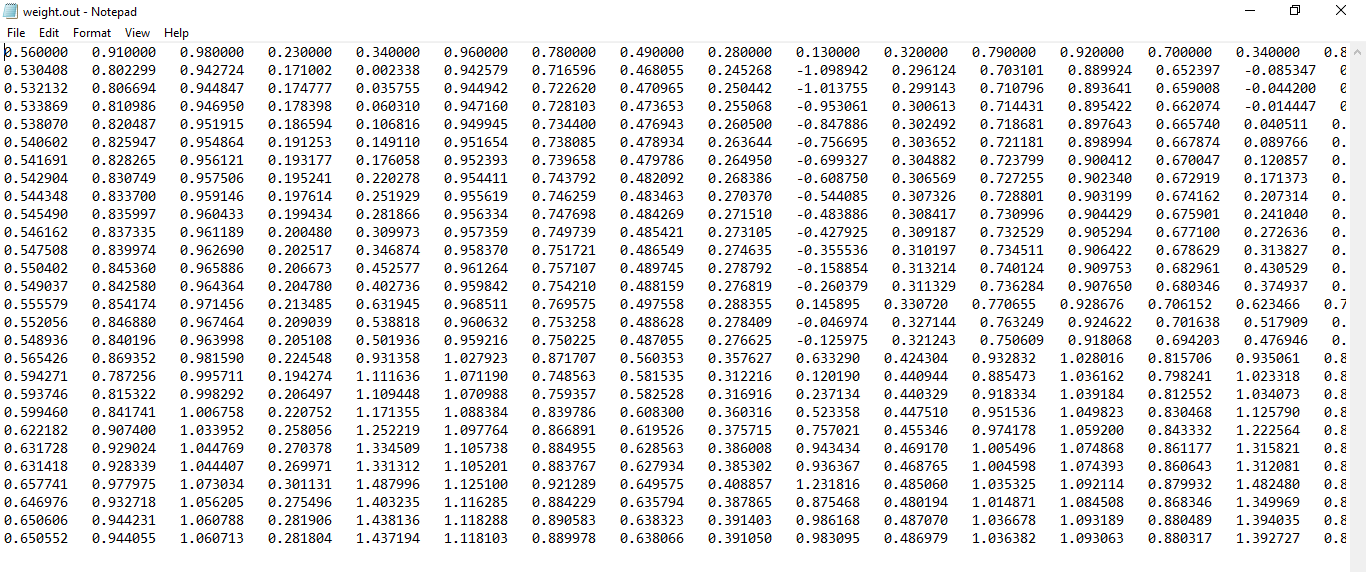
……… ……….. ………. ……….. …….

……… ……….. ………. ……….. …….

W[nn][1] W[nn][2] ………. W[nn][nn] B[nn]

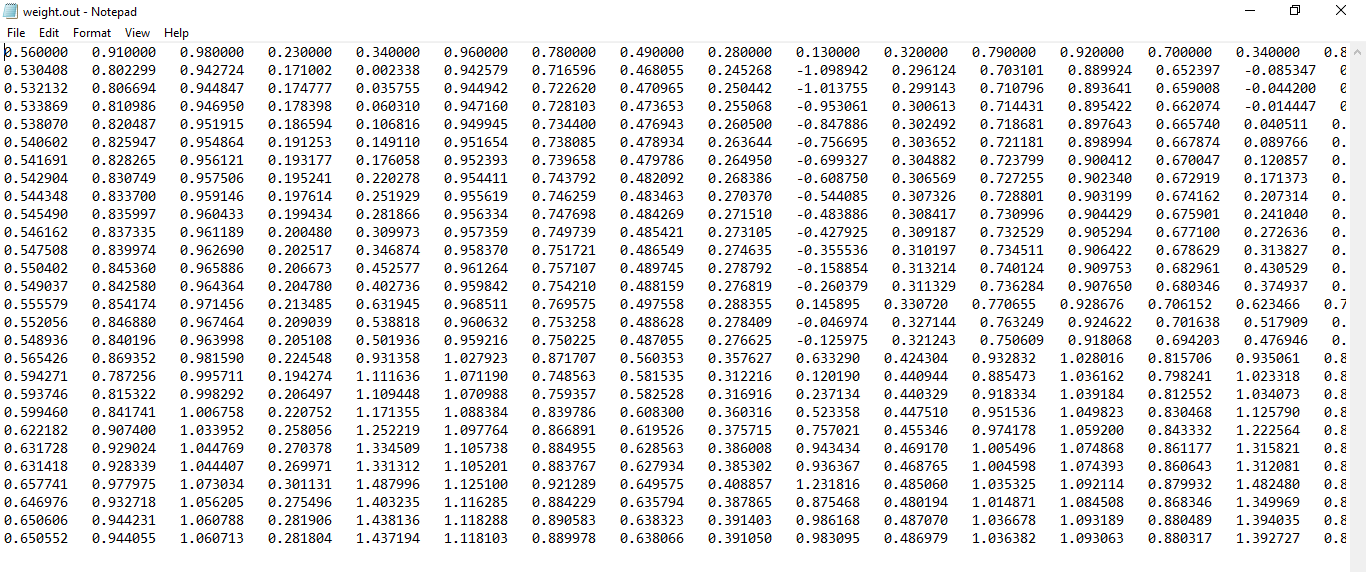
W[1][nn+1] W[1][nn+1] ………. W[nn][nn+1] Bo

*(chú thích : nn = NUM\_NEURAL : số lượng neural)*



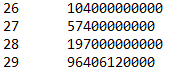
Hình 3.1.2: Output đầu ra cho chương trình training.

* Quá trình tính toán và dự đoán:
  + Input:
    - Một file chứa dữ liệu tính toán ( weight và bias ) của quá trình training trước đó:



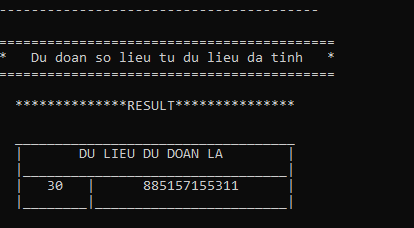
Hình 3.1.3: Input đầu vào cho quá trình dự đoán.

* + - Sau khi lấy được dữ liệu từ training chương trình sẽ thực hiện lấy dữ liệu 4 tháng cuối để dự đoán tháng tiếp theo :



Hình 3.1.4: Input đầu vào cho quá trình dự đoán.

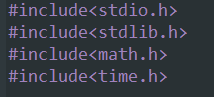
* + Output: doanh thu của tháng đó:



Hình 3.1.5: Output đầu ra cho quá trình dự đoán.

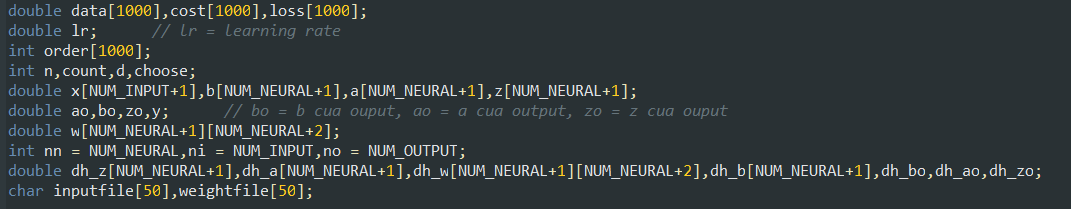
## Cấu trúc dữ liệu

1. Khai báo thư viện

**

Hình 3.2.1: Khai báo thư viện.

1. Khai báo kiểu dữ liệu



Hình 3.2.2: Khai báo kiểu dữ liệu.

+ Kiểu số nguyên: lr là learning rate, n là số phần từ có trong data, 2 biến count và d dùng để đếm, choose là dùng để lựa chọn, nn là số lượng neural, ni là số lượng input, no là số lượng output.

+ Kiểu số thực: ao và zo dùng để tính giá trị cho output, bo là giá trị bias của output, y là giá trị thực tế của output để so sánh,dh\_ao,dh\_bo,dh\_zo là các giá trị đạo hàm của ao,bo,zo.

+ Kiểu sâu chuỗi: inputfile[] dùng để chứa tên của của file nhập input, weightfile[] dùng tên của file để chứa dữ liệu weight và bias sau khi tính xong.

+ Kiểu mảng:

* Mảng 1 chiều: data[] dùng để chứa dữ liệu chọn vào, cost[] dùng để chứa các giá trị cost, loss[] dùng để chứa các giá trị của loss function, x[] dùng để chứa ni input đầu vào, b[] dùng để chứa bias cho từng neural, a[],z[] dùng để tính giá trị cho từng neural,dh\_z[],dh\_a[],dh\_b[] dùng để chứa đạo hàm của các mảng z[],a[] và b[]
* Mảng 2 chiều: w[][] là mảng dùng để chứa weight nối với từng neural với nhau, dh\_w[][] là mảng dùng để chứa đạo hàm của w[][]

## Thuật toán

* Thuật toán được trình áp dụng đối với trường hợp số input = 4 và số neural trong hidden layer = 4
* **B1: Lấy input và output từ data**

lặp ( i = 1-> n - 4):

k = 1

lặp ( j = i -> i +3 ):

x[k] = data[j]

k++

**+ Quá trình training :**

* + **B2: Propagation**

Lặp (i=1 -> 4):

// giá trị của neural

z =

a = // giá trị của của output

* + **B3: Quá trình lan truyền tính hiệu lỗi**

Lặp (i =1->4)

* + **B4: Backpropagation (Updata weight và bias)**

Lặp ( i = 1-> 4 ) :

Lặp ( i=1->4 ):

Lặp ( j=1->4 ):

+ Sau khi thực hiện sau n - ni lần training ta sẽ thu được trọng số weight và bias để thực hiện quá trình dự đoán: ( n ở đây là số dữ liệu đưa vào, ni là số input – number input )

* **B5: Dự đoán số liệu cho tháng tiếp theo:**

Lặp ( i=1->4 ):

x[i] = data[ n-4-1+i]

Lặp (i=1 -> 4): // propagation

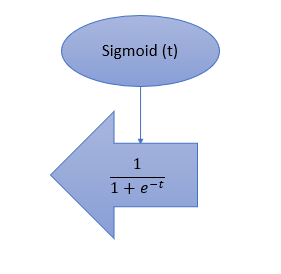
z =

kq =

* **Bảng vẽ sơ đồ khối:**

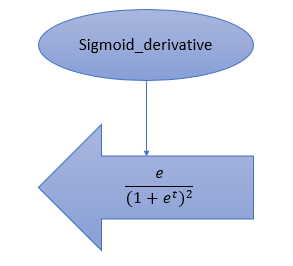
### Một số hàm hỗ trợ

* Hàm sigmoid ( t ) : Hàm này có chức năng dùng để tính sigmoid của một số t



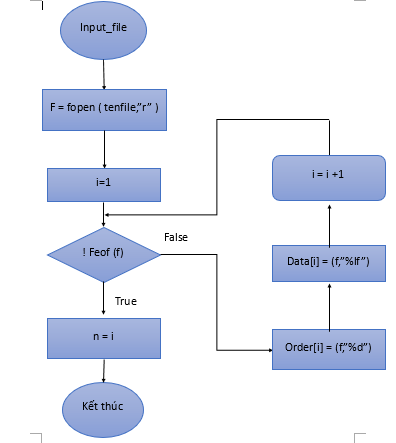
Hình 3.3.1: Hàm sigmoid.

* Hàm sigmoid – derivative ( t ): hàm này có chức năng tính đạo hàm của hàm sigmoid theo t



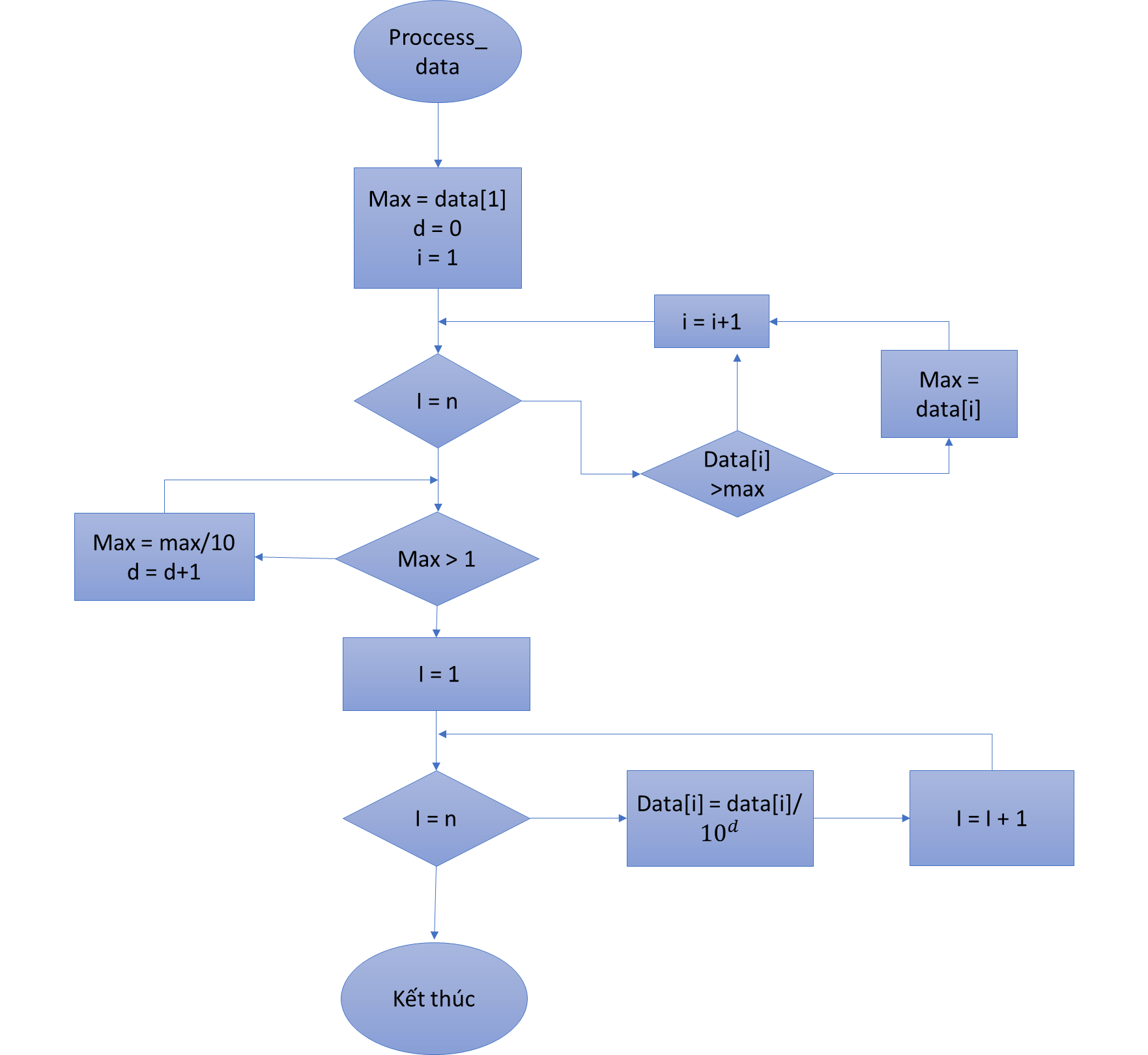
Hình 3.3.2: Hàm đạo hàm của sigmoid.

* Hàm Input\_file : Hàm này có chức năng dùng để lấy dữ liệu từ file và gán vào 2 mảng: Order ( Thứ tự ) và Data ( Dữ liệu )



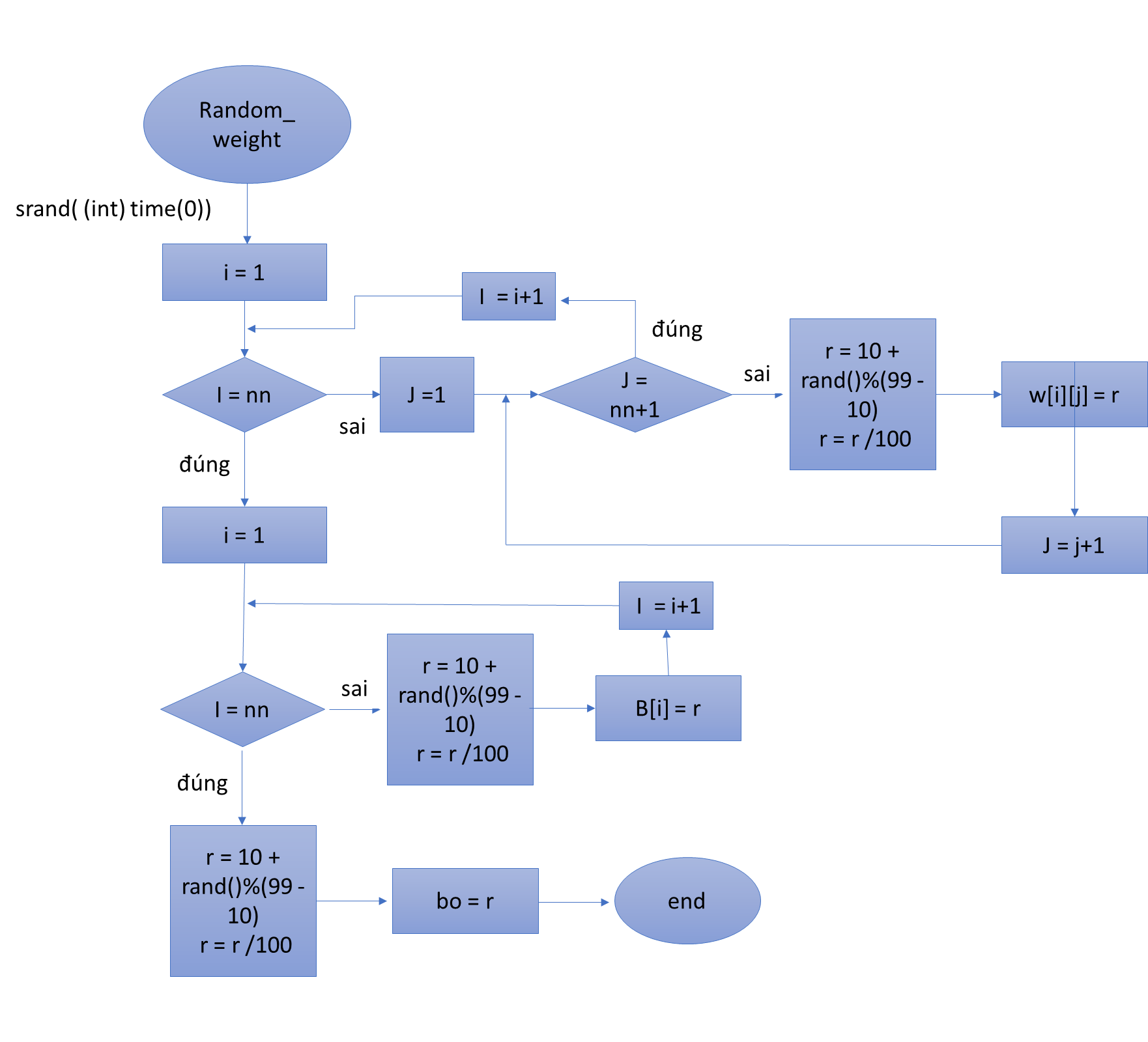
Hình 3.3.3: Hàm input file.

* Hàm process\_data : Hàm này dùng để đưa dự liệu từ một số to về 1 số chạy từ 0 tới 1:



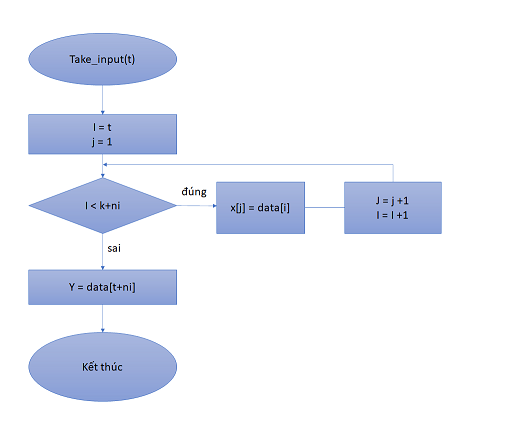
Hình 3.3.4: Hàm proccess data()

* Hàm random\_weight : Hàm có chức năng random weight và bias cho quá trình đầu tiên của việc training ( weight và bias sẽ được random là 1 số thực chạy từ 0 tới 1 và có 2 chữ số thập phân )



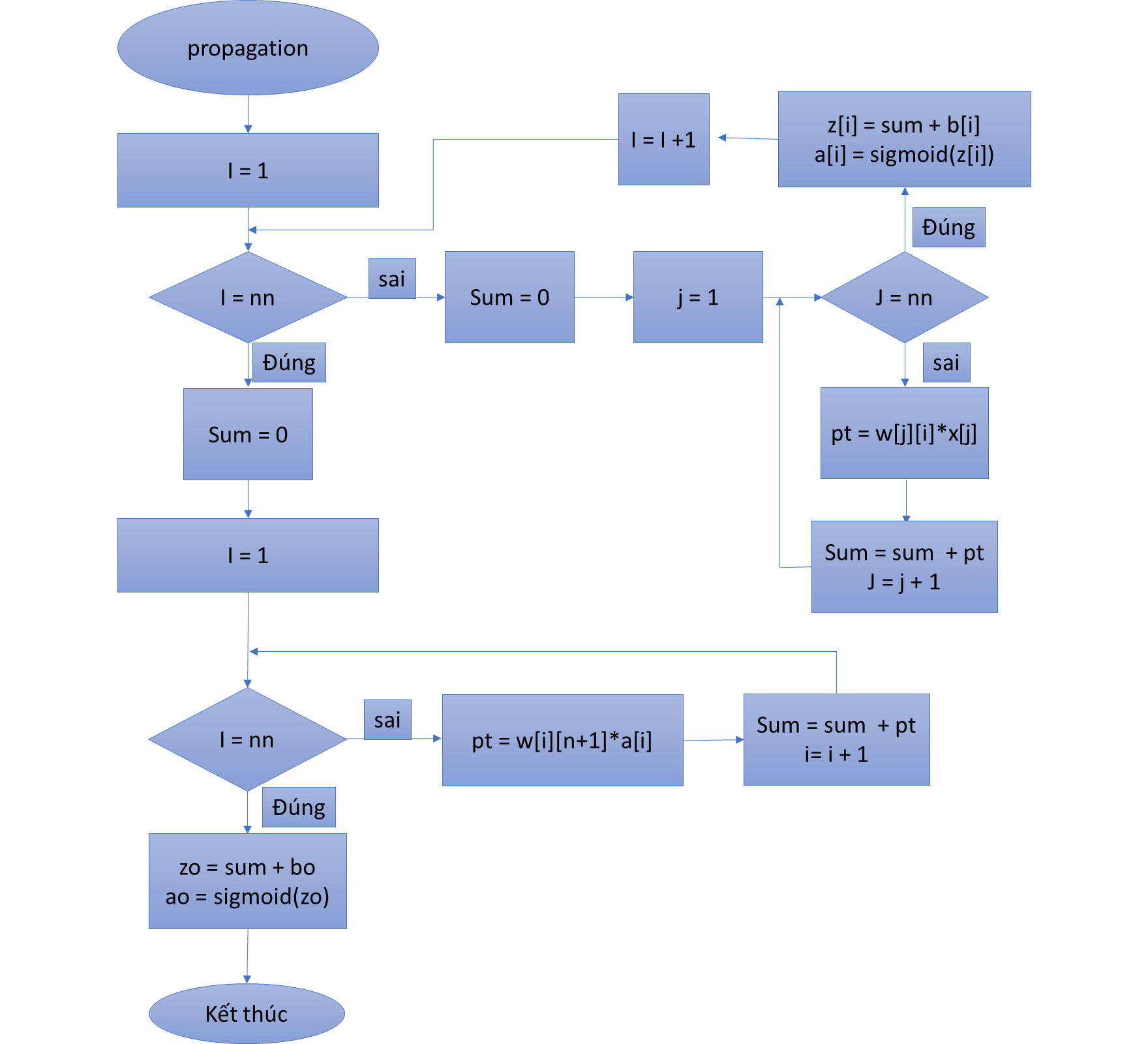
Hình 3.3.5: Hàm random weight.

* Hàm take\_input(t) : Hàm lấy input đầu vào để thực hiện việc tính toán ở lần thứ t



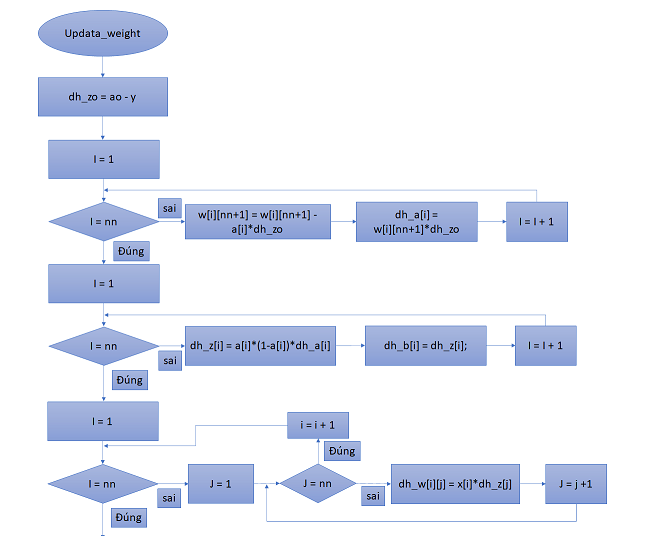
Hình 3.3.6: Hàm take input.

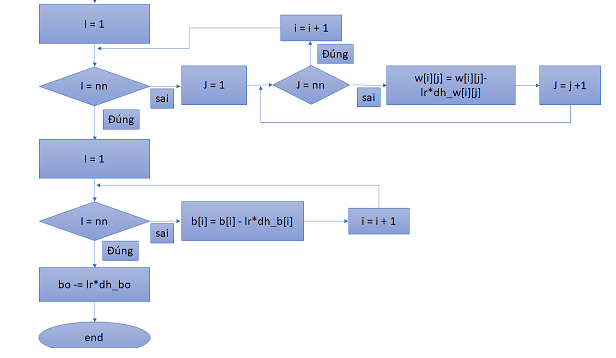
* Hàm propagation : hàm này có chức năng tính toán các quá trình progation trước ( tính z và a của các neural )



Hình 3.3.7: Hàm propagation.

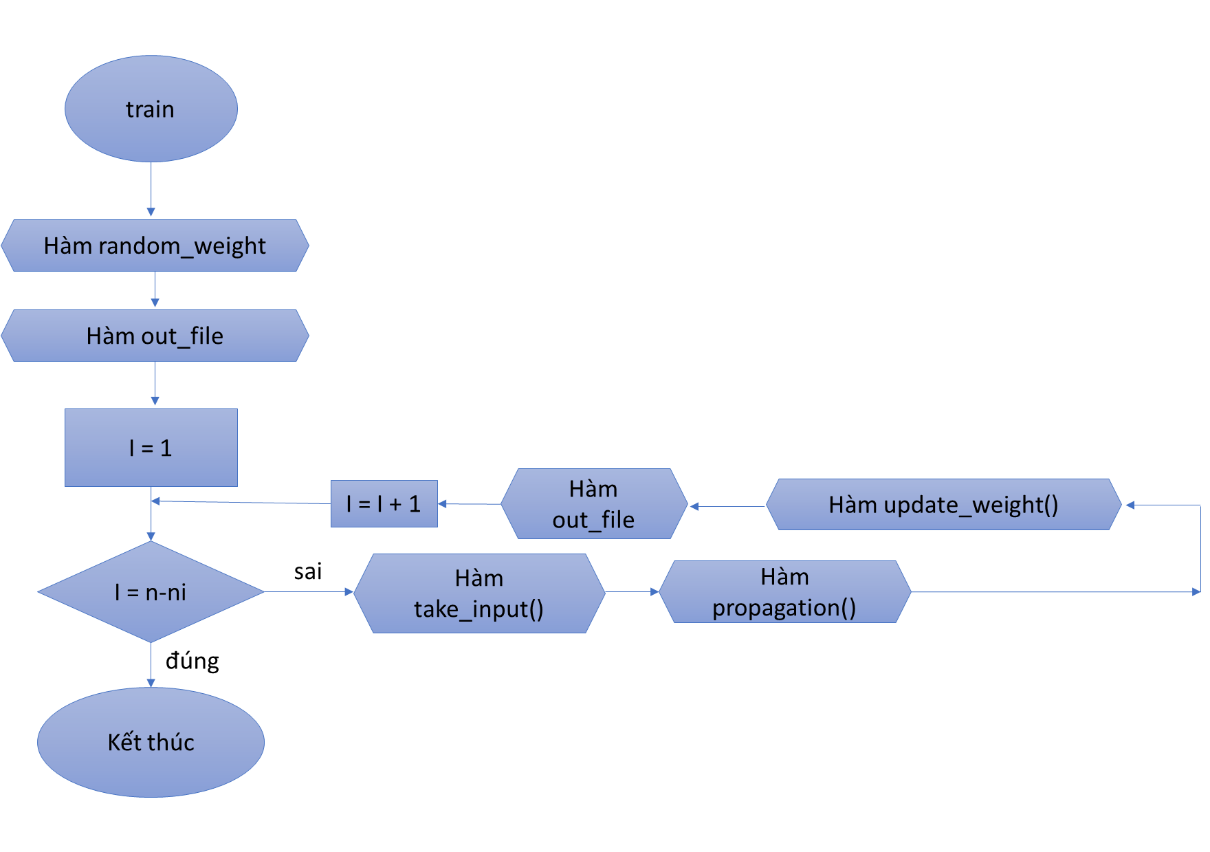
* Hàm update\_weight() : hàm này có chức năng thay đổi và chỉnh sửa các trọng số ( weight, bias ) trong quá trình lan truyền ngược ( backprogation ):





Hình 3.3.8: Hàm update weight.

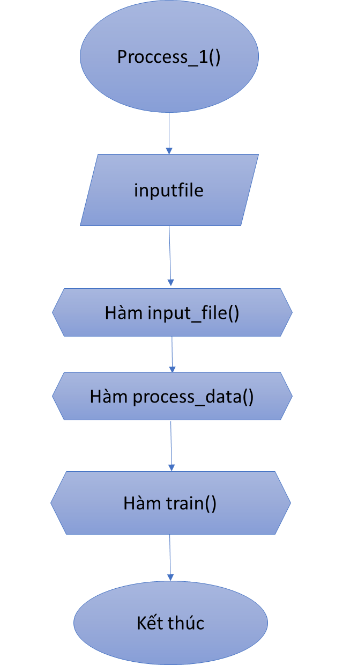
* Hàm train(): Hàm này có chức năng thực hiện chức năng training cho chương trình



Hình 3.3.10: Hàm train.

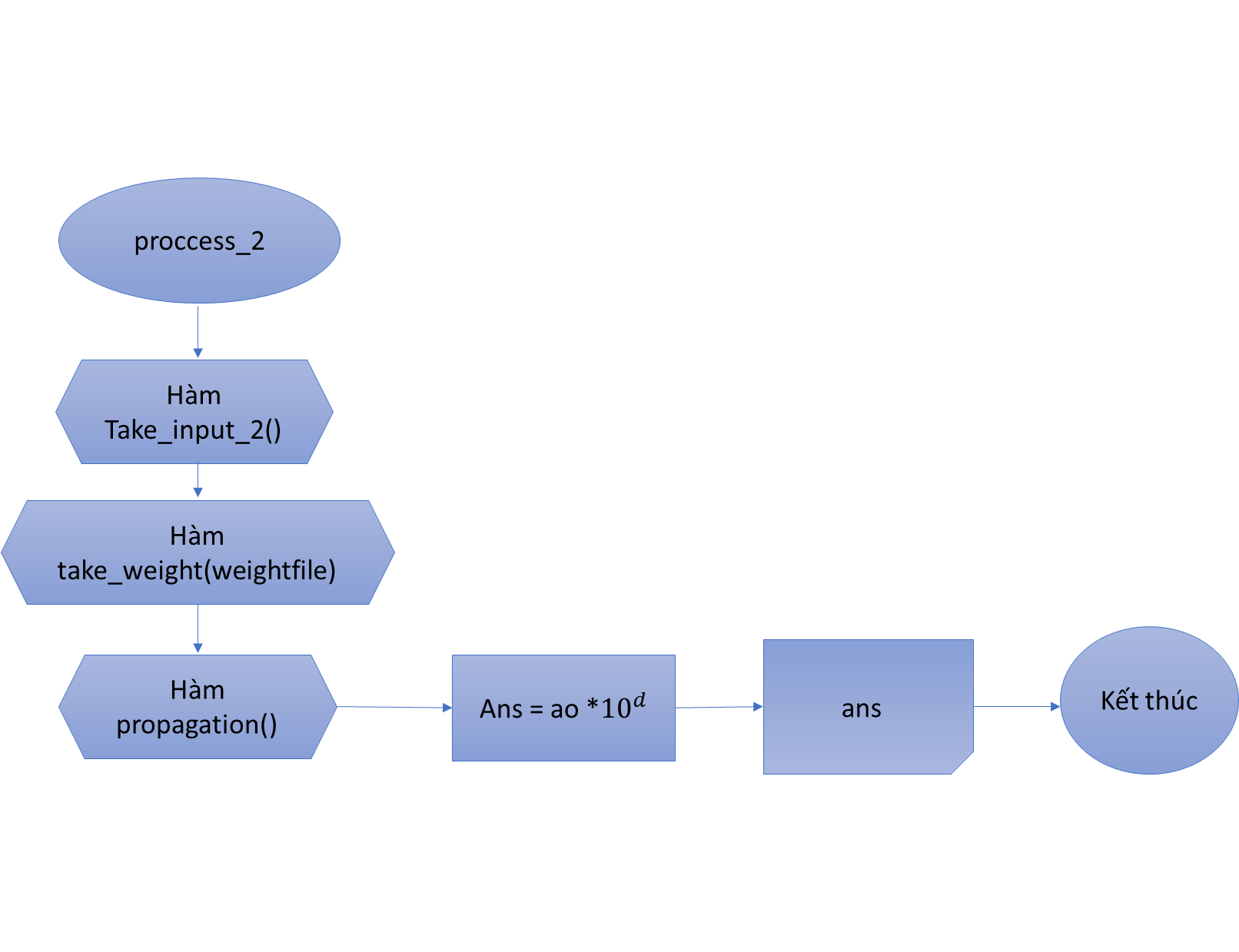
### Hàm chính

* Hàm proccess\_1() : thực hiện tất cả các khâu training từ việc lấy dữ liệu, xử lý số liệu và training



Hình 3.3.11: Hàm proccess\_1.

* Hàm proccess\_2 : dùng để dự đoán doanh thu của tháng tiếp theo từ dữ liệu training đã có



Hình 3.3.12: Hàm proccess\_2.

### Độ phức tạp của thuật toán

* Độ phức tạp của thuật toán là o (

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

* Đầu tiên bài toàn cần có 1 nguồn dữ liệu doanh thu của doanh nghiệp đưa vào. Ta tạo một file và ghi doanh thu của doanh nghiệp vào đó và tiến hành lấy dữ liệu từ file đó và lưu vào một mảng.
* Sau khi đã có dữ liệu đưa vào ta tiến hành viết hàm normalize dữ liệu, tiến hành xử lý dữ liệu đưa về các con số chạy từ 0 tới 1.

**Quá trình training**

* Trong quá trình training đầu tiên ta viết 1 hàm để random weight và bias trước quá trình tính toán. Sau đó viết 1 hàm in các weight và bias ra để dễ quan sát và thuận tiện cho quá trình tính toán và dự đoán sau này
* Trong quá trình training ta thực hiện 1 vòng lặp để thực hiện quá trình training đến cặp phần từ input và output cuối cùng. Trong quá trình đó, ta cần viết 1 hàm để lấy input và output từ mảng dữ liệu đã lấy được từ file ban đầu.
* Ta viết 1 hàm thực hiện để giá trị cho từng neural thông qua các giá trị weight và bias đã có trước đó. Sau khi thực hiện xong quá trình ta tiến hành viết hàm để chỉnh sửa và thay đổi lại các weight và bias. Sau quá trình backprogation, ta tiến hành tính cost và loss function thông qua một mảng để tiện theo dõi quá trình training của mạng neural.
* Tất cả tính toán weight và bias trong mỗi lần lặp đều được ghi vào một file để thuận tiện cho việc dự đoán doanh số sau này.

**Quá trình dự đoán**

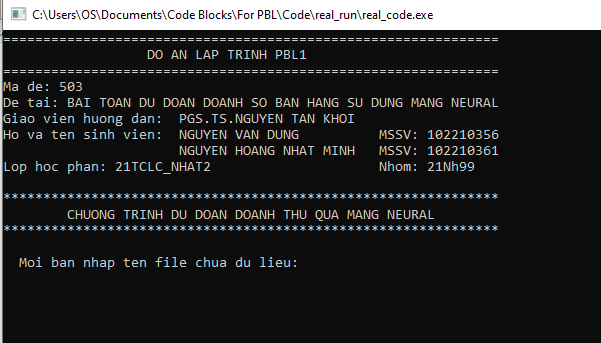
* Sau khi kết thức quá trình training ta sẽ nhận lại được một file chứa toàn bộ weight và bias trong quá trình training. Ta sẽ viết một hàm để lấy weight và bias từ file đó để tiến hành quá trình dự đoán.
* Sau khi đã lấy được trong số tiếp theo ta cần 4 input đầu vào, ta viết một hàm để lấy input đầu vào để tiến hành quá trình dự toán output đầu. Dự liệu khi lấy vào phải được tiến hành normalize chuyển từ dạng phức tạp về dạng chạy từ 0 tới 1.
* Sau đó thông qua mảng weight và bias chúng ta lấy được cùng một mảng input đầu vào ta tiến hành quá trình tính toán giá trị cho từng neural và đưa ra kết quả của việc dự đoán.

## Ngôn ngữ cài đặt

* Ngôn ngữ dùng để viết thuật toán ở đây là: C
* Ứng dụng dùng để viết thuật toán là: DevC++
* [Code](#code)

## Kết quả thực hiện

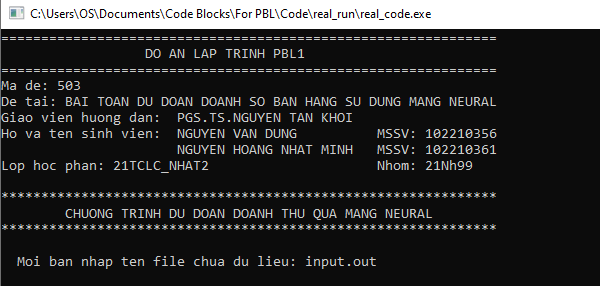
### Giao diện chính của chương trình



Hình 4.3.1: Giao diện của chương trình.

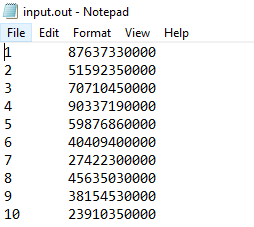
* Chương trình sẽ thực hiện 2 quá trình:

1. Thực hiện quá trình training cho mạng neural
   * + Bước 1: cần phải nhập đó chính là đường link dẫn tới file chưa dữ liệu (VD ở hình dưới là file “input.out”)



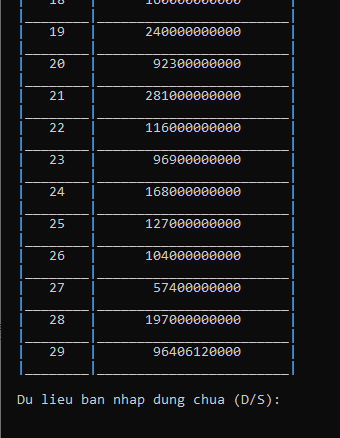
Hình 4.3.2: Chọn lựa chọn và nhập file chứa dữ liệu.

* + - * Trong file chứa dữ liệu sẽ được ghi dưới dạng:
      * STT doanh thu



Hình 4.3.3: Ví dụ về file dữ liệu đưa vào.

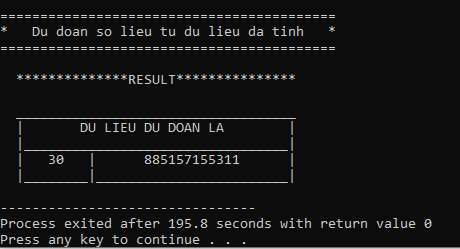
* + - Bước 2: Sau khi nhập xong file chứa dữ liệu, màn hình chương trình sẽ hiện ra dữ liệu mà chúng ta đã nhập, ở bước này chúng ta cần kiểm tra xem dữ liệu đưa vào đã đúng chưa: Nếu như đã đúng rùi thì nhấn ‘D’ còn nếu chưa đúng thì nhấn ‘S’ và thực hiện lại bước 1.



Hình 4.3.4: Kiểm tra bảng dữ liệu đưa vào đã đúng chưa.

1. Thực hiện quá trình dữ đoán

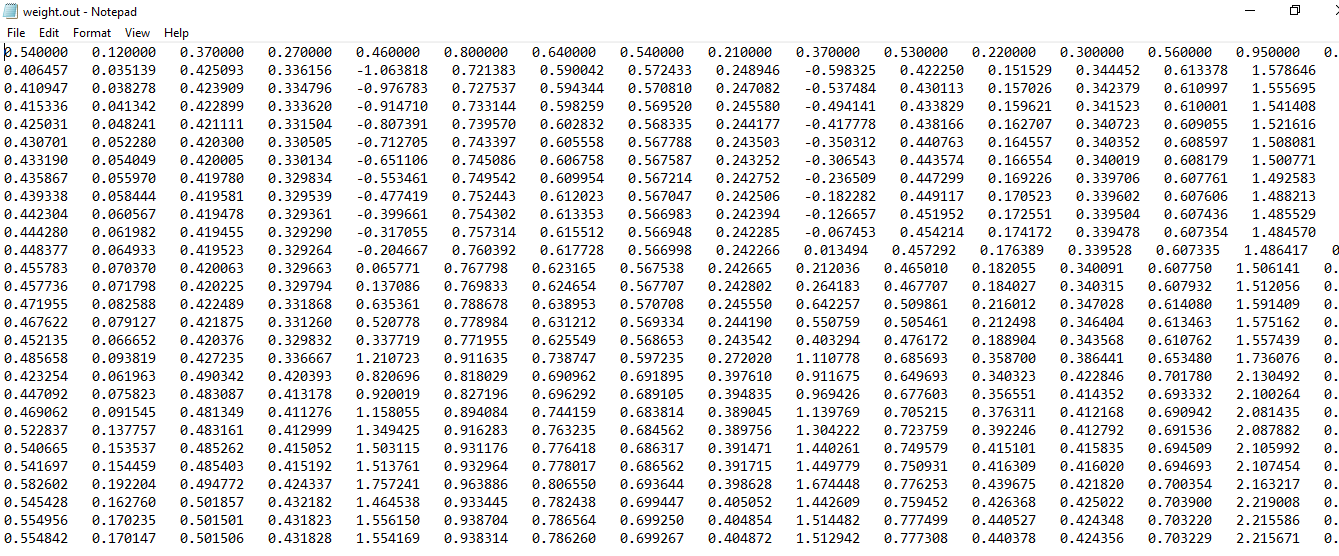
* Sau khi thực hiện xong quá trình training chương trình sẽ đưa ra cho chúng ta kết quả dự đoán trong tháng tiếp theo:



Hình 4.3.5: Kết quả dự đoán.

### Các kết quả thực thi của chương trình

* Chương trình có 2 quá trình:
  + Training Neaural ( quá tình 1 ) : Sau khi thực hiện các đoạn trong chương trình chương trình sẽ đưa ra một file (VD ở đây là file “weight.out”) chứa toàn bộ quá trình training của thuật toán:



Hình 4.3.6: File dữ liệu thu được sau quá trình training.

Thứ tự lần lượt sẽ là:  
W[1][1] W[1][2] ………. W[1][nn] B[1]

W[2][1] W[2][2] ………. W[2][nn] B[2].

**- - - - -**

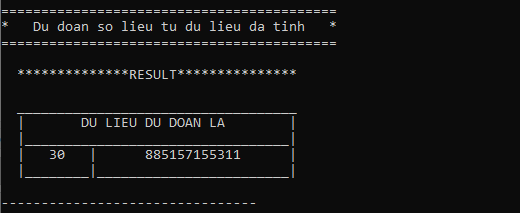
**- - - - -**

W[nn][1] W[nn][2] ………. W[nn][nn] B[nn]

W[1][nn+1] W[1][nn+1] ………. W[nn][nn+1] Bo

(chú thích : nn = NUM\_NEURAL : số lượng neural)

* + Dự đoán doanh thu của một tháng ( quá tình 2 ):Sau khi thực hiện xong chương trình thuật toán sẽ đưa ra dự đoán của mình



Hình 4.3.7: Kết quả của việc dự đoán doanh thu thu được.

### Nhận xét đánh giá kết quả

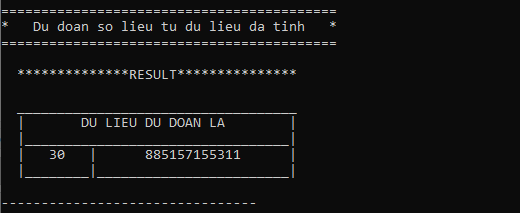
* Để có thể đưa ra một dữ đoán chính xác và gần đúng thì trong quá trình training phải đưa vào một số lượng lớn dữ liệu doanh thu của doanh nghiệp trong những năm trước đó.
* Nếu dữ liệu của đưa vào quá ít thì trong quá trình training cần phải đưa vào một lượng learning rate đủ lớn để kích thích cũng như đẩy nhanh quá trình training ( Nhưng cách này sẽ khiến cho sai số trong quá trình dự đoán sẽ rất lớn )
* Nhưng để việc dự đoán có thể chính xác thì việc đó phụ thuộc vào quá trình training cũng như nguồn dữ liệu đưa vào cần có một mức dao động nhất định , bên đó cạnh đó việc điều chỉnh số hidden layer hoặc chỉ số learning rate cũng ảnh hưởng khá lớn đến việc đưa ra con số dự đoán chính xác

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

* Hệ thống hóa các nội dung cơ bản về mạng nơron nhân tạo và thuật toán di truyền
* Nghiên cứu các phương pháp kết hợp giải thuật Di truyền với giải thuật Lan truyền ngược sai số nhằm đạt tới một kết quả trọn vẹn của bài toán tối ưu trọng số mạng nơron nhân tạo
* Xây dụng chương trình có khả năng dự đoán được doanh số bán hàng của một doanh nghiệp trong một tháng nhất định
* Kết quả đã có thể dự đoán được số liệu doanh thu của Hiệu sữa Vinamilk

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Tháng năm** | **Ngày** | **Doanh thu** |
| 1 | 201606 | 1 | 87637330000 |
| 2 | 201606 | 2 | 51592350000 |
| 3 | 201606 | 3 | 70710450000 |
| 4 | 201606 | 4 | 90337190000 |
| 5 | 201606 | 5 | 59876860000 |
| 6 | 201606 | 6 | 40409400000 |
| 7 | 201606 | 7 | 27422300000 |
| 8 | 201606 | 8 | 45635030000 |
| 9 | 201606 | 9 | 38154530000 |
| 10 | 201606 | 10 | 23910350000 |
| 11 | 201606 | 11 | 36457930000 |
| 12 | 201606 | 12 | 27384000000 |
| 13 | 201606 | 13 | 27385000000 |
| 14 | 201606 | 14 | 28537960000 |
| 15 | 201606 | 15 | 37820520000 |
| 16 | 201606 | 16 | 84600000000 |
| 17 | 201606 | 17 | 38400000000 |
| 18 | 201606 | 18 | 160000000000 |
| 19 | 201606 | 19 | 240000000000 |
| 20 | 201606 | 20 | 92300000000 |
| 21 | 201606 | 21 | 281000000000 |
| 22 | 201606 | 22 | 116000000000 |
| 23 | 201606 | 23 | 96900000000 |
| 24 | 201606 | 24 | 168000000000 |
| 25 | 201606 | 25 | 127000000000 |
| 26 | 201606 | 26 | 104000000000 |
| 27 | 201606 | 27 | 57400000000 |
| 28 | 201606 | 28 | 197000000000 |
| 29 | 201606 | 29 | 96406120000 |
|  |  |  |  |



Hình 5.1.1: Kết quả thu được.

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | 201606 | 30 | 81598620000 |

## Hướng phát triển

* Dự báo doanh số bán hàng của doanh nghiệp qua từng tháng qua đó giúp doanh nghiệp kiểm soát được giá trị, số lượng sản phẩm bán ra, nhu cầu thị trường qua đó có những biện pháp cần thiết, kịp thời nhằm nâng cao năng suất cho mặt hàng.
* Mạng nơ-ron nhân tạo có thể đánh giá dữ liệu giá và khai quật các cơ hội giao dịch dựa trên phân tích dữ liệu lịch sử. Từ đó giúp cho doanh nghiệp đưa ra một kế hoạch phát triển hợp lý
* Thử nghiệm ứng dụng dự báo doanh thu bán hàng qua hàng tháng, thử nghiệm dự báo lợi nhuận thu được, biến động giá trị thị trường tận dụng việc tính kết quả nhanh chóng của mạng nơron.
* Ứng dụng vào các bài toán khác trong thực tiễn như: nhận dạng chữ viết tay, nhận dạng chuyển động, nhận dạng đồ vật, nhận dạng biển số xe, nhận dạng và giả lập giọng nói qua việc học các mẫu âm thanh,...
* Xây dựng một hệ thống siêu máy tính mini phục vụ cho công việc huấn luyện mạng nơron, cải tiến mô hình mạng nơron kết hợp các phương pháp học khác nhau.

TÀI LIỆU THAM KHẢO

1. <https://www.youtube.com/watch?v=tAy8a6c598w&list=PLmtLOdLKF-og3HsyUGGNR5lhn96TOUllX>.
2. [http://trituevietvn.com/chi-tiet/-NGHIEN-CUU-VE-MANG- NEURON-NHAN-TAO-VA-UNG-DUNG-DU-BAO-DOANH-SO-BAN-HANG-60](http://trituevietvn.com/chi-tiet/-NGHIEN-CUU-VE-MANG-%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20NEURON-NHAN-TAO-VA-UNG-DUNG-DU-BAO-DOANH-SO-BAN-HANG-60).
3. <https://imech.ac.vn/upload/NewsImage/2021/1/12/ung-dung-mang-noron-nhan-tao-vao-bai-toan-du-bao-thuy-van.pdf>.
4. <https://itnavi.com.vn/blog/neural-network-la-gi/>.
5. <https://nttuan8.com/bai-4-backpropagation/>.
6. <http://kdientu.duytan.edu.vn/media/50176/ly-thuyet-mang-neural.pdf>.
7. <https://codereview.stackexchange.com/questions/191498/simple-neural-network-in-c>.

CODE

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<math.h>**

**#include<time.h>**

**#define NUM\_INPUT 4**

**#define NUM\_NEURAL 4**

**#define NUM\_OUTPUT 1**

// Moi mang deu bat dau tu so 1 nen kich co cong them 1

**double** data[1000],cost[1000],loss[1000];

**double** lr = 20;

**int** order[1000];

**int** n,d;

**double** x[NUM\_INPUT+1],b[NUM\_NEURAL+1],a[NUM\_NEURAL+1],z[NUM\_NEURAL+1];

**double** ao,bo,zo,y,min,max; // bo = b cua ouput, ao = a cua output, zo = z cua ouput

**double** w[NUM\_NEURAL+1][NUM\_NEURAL+2];

**int** nn = NUM\_NEURAL,ni = NUM\_INPUT,no = NUM\_OUTPUT;

**double** dh\_z[NUM\_NEURAL+1],dh\_a[NUM\_NEURAL+1],dh\_w[NUM\_NEURAL+1][NUM\_NEURAL+2],dh\_b[NUM\_NEURAL+1]

**double** dh\_bo,dh\_ao,dh\_zo;

**char** inputfile[50],weightfile[50] = "weight.out";

**void** out\_result(**double** ans){

**printf**("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*RESULT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

**printf**("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

**printf**(" | DU LIEU DU DOAN LA |\n");

**printf**(" |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

**printf**(" | %2d | %12.lf |\n",n+1,ans);

**printf**(" |\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

}

**void** out\_data(){

**printf**("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

**printf**(" | BANG DU LIEU DA NHAP |\n");

**printf**(" |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

**printf**(" | STT | DOANH THU |\n");

**printf**(" |\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

**for** (int i=1;i<=n;i++){

**printf**(" | %2d | %12.lf |\n",order[i],data[i]);

**printf**(" |\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

}

}

**double** sigmoid(**double** t){

**return** 1/(1+exp(-t));

}

// dung de lay input tu file

**void** input\_file(**double** data[], **int** order[], **int** &n, **char** tenFile[50]){

FILE \*f;

f = fopen(tenFile,"r");

**if**(f==NULL){

**printf**("\n Loi Mo File \n");

**return**;

}**int** i=1;

**while**(!feof(f)){

// lay input tu file

f**scanf**(f,"%d",&order[i]);

f**scanf**(f,"%lf",&data[i]);

i++;

}fclose(f);

n = i-2;

}

// dung de in ra w b bo cost ra file

**void** out\_file(**char** tenFile[100]){

FILE \*f;

f = fopen(tenFile, "a");

**if**(f==NULL){

**printf**("\n Loi mo file \n");

**return**;

}

**int** i,j;

**for**(i=1;i<=nn;i++){ // in ra file tu w[1][1] toi w[4][4]

**for**(j=1;j<=nn;j++){

f**printf**(f,"%lf ",w[i][j]);

}f**printf**(f,"%lf ",b[i]);

}**for**(i=1;i<=nn;i++){

f**printf**(f, "%lf ",w[i][nn+1]);

}f**printf**(f,"%lf \n",bo);

fclose(f);

}

// dung de random weight va cac bias

**void** rand\_weight(**double** w[][NUM\_NEURAL+2],**double** b[],**double** &bo, **int** nn){

**in**t i,j;

srand((int)time(0));

**double** r ;

**for**(i=1;i<=nn;i++){

**for**(j=1;j<=nn+1;j++){

r = 10 + rand()%(99 - 10);

r /= 100;

w[i][j] = r;

}

}**for**(i=1;i<=nn;i++){

r = 10 + rand()%(99 - 10);

r /= 100;

b[i] = r;

}r = 10 + rand()%(99-10);

r /= 100;

bo = r;

}

// xu ly so lieu dua ve 0 va 1

**void** proccess\_data(**double** data[],**int** n){

**double** max = data[1];

**int** i;

d = 0;

**for**(i=1;i<=n;i++){ // tim so lon nhat trong data

**if**(data[i]>=max){

max = data[i];

}

}

**while**(max>=1){ // tinh so chu so cua max

max /= 10;

d++;

}

**for**(i=1;i<=n;i++){

data[i] /= pow(10,d);

}

}

// lay ra 4 input tu data

**void** take\_input(**int** k,**double** x[],**double** data[],**int** ni){

**int** i,j=1;

**for**(i=k;i<k+ni;i++){

x[j]=data[i];

j++;

}y = data[k+4];

}

// propagation

**void** propagation(){

// tinh z va a

**double** sum,pt ;

**in**t i,j;

**for**(i=1;i<=nn;i++){

sum = 0;

**for**(j=1;j<=nn;j++){

pt = w[j][i]\*x[j];

sum += pt;

}z[i] = sum + b[i];

a[i] = sigmoid(z[i]);

}

// tinh zo va ao

sum = 0;

**for**(i=1;i<=nn;i++){

pt = w[i][nn+1]\*a[i];

sum += pt;

}zo = sum + bo;

ao = sigmoid(zo);

}

// update weight process (backprogation)

**void** update\_weight(){

**int** i,j;

// ouput layer -> hidden layer

dh\_zo = ao - y;

**for**(i=1;i<=nn;i++){

w[i][nn+1] = w[i][nn+1] - a[i]\*dh\_zo;

dh\_a[i] = w[i][nn+1]\*dh\_zo;

}dh\_bo = dh\_zo;

//hidden layer -> input layer

**for**(i=1;i<=nn;i++){

dh\_z[i] = a[i]\*(1-a[i])\*dh\_a[i];

dh\_b[i] = dh\_z[i];

}**for**(i=1;i<=nn;i++){

**for**(j=1;j<=nn;j++){

dh\_w[i][j] = x[i]\*dh\_z[j];

}

}

// update weight

**for**(i=1;i<=nn;i++){

**for**(j=1;j<=nn+1;j++){

w[i][j] -= lr\*dh\_w[i][j];

}

}

// update bias

**for**(i=1;i<=nn;i++){

b[i] -= lr\*dh\_b[i];

}bo -= lr\*dh\_bo;

}

// training process

**void** train(**char** tenFile[50]){

**int** i,j;

rand\_weight(w,b,bo,nn);

out\_file(tenFile);

**for**(i=1;i<=n-ni;i++){

take\_input(i,x,data,ni);

propagation();

update\_weight();

out\_file(tenFile);

}

}

**void** proccess\_1(){

//lay du lieu

**printf**("\n Moi ban nhap ten file chua du lieu: ");

gets(inputfile);

input\_file(data,order,n,inputfile);

out\_data();

// xu ly so lieu

proccess\_data(data,n);

//training

train(weightfile);

}

**void** take\_input\_2(){

**int** i;

**for**(i=1;i<=ni;i++){

x[i] = data[n-ni+i]\*pow(10,d);

}

}

**void** take\_weight(**char** tenFile[50]){

FILE \*f;

f = fopen(tenFile, "r");

**if**(f==NULL){

**printf**("\n Loi mo file \n");

**return**;

}

**int** i,j;

**for**(i=1;i<=nn;i++){

**for**(j=1;j<=nn;j++){

f**scanf**(f,"%lf",&w[i][j]);

}f**scanf**(f,"%lf",&b[i]);

}**for**(i=1;i<=nn;i++){

f**scanf**(f,"%lf",&w[i][nn+1]);

}f**scanf**(f,"%lf",&bo);

fclose(f);

}

**void** proccess\_2(){

**printf**("\n==========================================");

**printf**("\n\* Du doan so lieu tu du lieu da tinh \*");

**printf**("\n==========================================\n");

// lay du lieu

take\_input\_2();

// xu ly du lieu

proccess\_data(x,ni);

// lay weight

take\_weight(weightfile);

// tinh toan va dua ra du doan

propagation();

double ans = ao\*pow(10,d);

out\_result(ans);

}

**int** main(){

**printf**("==============================================================\n");

**printf**(" DO AN LAP TRINH PBL1 \n");

**printf**("==============================================================\n");

**printf**("Ma de: 503\n");

**printf**("De tai: BAI TOAN DU DOAN DOANH SO BAN HANG SU DUNG MANG NEURAL\n");

**printf**("Giao vien huong dan: PGS.TS.NGUYEN TAN KHOI\n");

**printf**("Ho va ten sinh vien: NGUYEN VAN DUNG MSSV: 102210356\n");

**printf**(" NGUYEN HOANG NHAT MINH MSSV: 102210361\n");

**printf**("Lop hoc phan: 21TCLC\_NHAT2 Nhom: 21Nh99\n\n");

**printf**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

**printf**(" CHUONG TRINH DU DOAN DOANH THU QUA MANG NEURAL \n");

**printf**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

**char** check='S';

**while**(check != 'D'){

proccess\_1();

**printf**("\n Du lieu ban nhap dung chua (D/S): ");

**scanf**("%c",&check);

**printf**("\n----------------------------------------\n");

fflush(stdin);

}

proccess\_2();

**return** 0;

}

**🕮🙡🙠END🙣🙣🕮**