

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CƠ SỞ

TÌM HIỂU MẠNG NEURON NHÂN TẠO ĐỂ DỰ
BÁO LẠM PHÁT Ở VIỆT NAM

Giảng viên hướng dẫn:	PHẠM ĐÌNH TÀI
Sinh viên thực hiện:	NGUYỄN THÀNH PHÁT
MSSV:	2000006273
Chuyên ngành:	Khoa học dữ liệu
Môn học:	Đồ án cơ sở Khoa học dữ liệu
Khóa:	2020

Tp.HCM, tháng 9 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CƠ SỞ

TÌM HIỂU MẠNG NEURON NHÂN TẠO ĐỂ DỰ
BÁO LẠM PHÁT Ở VIỆT NAM

Giảng viên hướng dẫn:	PHẠM ĐÌNH TÀI
Sinh viên thực hiện:	NGUYỄN THÀNH PHÁT
MSSV:	2000006273
Chuyên ngành:	Khoa học dữ liệu
Môn học:	Đồ án cơ sở Khoa học dữ liệu
Khóa:	2020

Tp.HCM, tháng 9 năm 2022

LỜI CẢM ƠN

Trước khi bắt đầu nội dung đồ án, chúng em muốn gửi lời cảm ơn đến những người đã chia sẻ kinh nghiệm, giúp đỡ chúng em trong quá trình thực hiện đề tài. Đầu tiên là thầy Phạm Đình Tài, giảng viên học phần Mạng Neural và giải thuật di truyền. Thầy là người đã cung cấp các kiến thức cơ bản về Khoa học dữ liệu cũng như các kiến thức chuyên ngành giúp chúng em có nền tảng để xây dựng ý tưởng và hiện thực hóa đề tài này.

Tiếp theo, em cũng xin bày tỏ lòng biết ơn đối với những thầy cô khoa CNTT đã truyền đạt kiến thức về các môn học cơ bản cũng như hướng dẫn chi tiết về cách thức trình bày đồ án.

Ngoài ra, các kiến thức chuyên sâu về Deep Learning trên mạng Internet cũng vô cùng hữu ích. Cảm ơn những người đã bỏ công sức ra chia sẻ, lan tỏa sự hiểu biết cho nhiều người khác.

Do kiến thức còn hạn chế nên đồ án không thể không tránh khỏi những thiếu sót. Rất mong quý thầy cô cũng như các anh chị và các bạn đóng góp ý kiến để chúng em có thể rút kinh nghiệm hoàn thành tốt hơn.

Em xin chân thành cảm ơn!

LỜI MỞ ĐẦU

Trong những năm gần đây, Việt Nam ta đã và đang trên đà phát triển vô cùng nhanh chóng. Nhiều thành tựu về khoa học kỹ thuật góp phần không nhỏ thúc đẩy sự tăng trưởng về kinh tế, dần thu hẹp khoảng cách với các quốc gia phát triển trên toàn thế giới. Tuy nhiên bên cạnh đó, ta vẫn phải đối mặt với nhiều biến động tiêu cực, đặc biệt là tình trạng lạm phát hiện nay. Mức độ lạm phát nếu không được kiểm soát một cách ổn định có thể gây ảnh hưởng lớn đối với công cuộc xây dựng và đổi mới đất nước.

Nắm được tình hình trên, nhóm chúng em đã vận dụng các kiến thức về mạng Neural nhân tạo để hình thành ý tưởng xây dựng một hệ thống có khả năng dự báo lạm phát tại Việt Nam. Hệ thống sử dụng hồi quy tuyến tính, lan truyền ngược, mạng RNN... giúp đưa ra dự đoán với độ chính xác cao. Thông qua các kết quả dự đoán này, nhà nước có thể điều chỉnh lại các chính sách kinh tế, tài chính của mình để bình ổn lạm phát ở một mức độ phù hợp với nền kinh tế thị trường.

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
TRUNG TÂM KHẢO THÍ

KỶ THI KẾT THÚC HỌC PHẦN
HỌC KỲ 3 NĂM HỌC 2021 - 2022

PHIẾU CHẤM THI TIỂU LUẬN/BÁO CÁO

Môn thi: Đồ án cơ sở Khoa học dữ liệuLớp học phần: 20DTH2A

Nhóm sinh viên thực hiện :

1. Nguyễn Minh Trí.....Tham gia đóng góp: 50%

2. Nguyễn Thành Phát.....Tham gia đóng góp: 50%

3.Tham gia đóng góp:

4.Tham gia đóng góp:

5.....Tham gia đóng góp:

6.....Tham gia đóng góp:

7.....Tham gia đóng góp:

8.....Tham gia đóng góp:

Ngày thi: 22/09/2022.....Phòng thi:

Đề tài tiểu luận/báo cáo của sinh viên : Tìm hiểu mạng Neuron nhân tạo để dự báo lạm phát ở Việt Nam

Phân đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Tiêu chí (theo CDR HP)	Đánh giá của GV	Điểm tối đa	Điểm đạt được
Cấu trúc của báo cáo		
Nội dung			
- Các nội dung thành phần		
- Lập luận		
- Kết luận		
Trình bày		
TỔNG ĐIỂM			

Giảng viên chấm thi
(ký, ghi rõ họ tên)

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU.....	1
1. Tổng quan về Khoa học dữ liệu.....	1
1.1 Khoa học dữ liệu là gì?	1
1.2 Mối quan hệ giữa Khoa học dữ liệu và Trí tuệ nhân tạo?	2
2. Ứng dụng của Khoa học dữ liệu	2
3. Deep learning là gì?	3
4. Ứng dụng của KHDL trong giải quyết vấn đề lạm phát.....	3
4.1 Lạm phát là gì?	3
4.2 Ảnh hưởng của lạm phát đối với nền kinh tế.....	4
4.3 Thực trạng vấn đề lạm phát ở Việt Nam trong những năm gần đây	5
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	6
1. Mạng thần kinh (Neuron Network)	6
1.1 Mạng nơ-ron sinh học.....	6
1.2 Mạng nơ-ron nhân tạo.....	7
2. Cấu trúc mạng nơ-ron	8
2.1 Perceptron là gì?	8
2.2 Hàm kích hoạt.....	9
2.3 Mạng nơ-ron nhiều lớp.	12
3. Cách thức hoạt động của mạng nơ-ron	13
3.1 Lan truyền thẳng	13
3.2 Hàm mất mát (Loss Function)	15

3.3 Một số giải thuật hàm mất mát thường dùng:.....	15
3.4 Lan nguyên ngược	16
3.5 Gradient Descent.....	18
4. Mạng thần kinh hồi quy (RNN).....	19
4.1 Cấu trúc mạng RNN truyền thống	20
4.2 Cấu trúc chi tiết.....	23
4.3 Các dạng bài toán RNN	23
4.4 Ưu và nhược điểm của một kiến trúc RNN thông thường	25
4.5 Vấn đề Vanishing gradient trong RNN (Đạo hàm bị triệt tiêu).....	25
5. Bộ nhớ dài-ngắn hạn (LSTM).....	27
5.1 Ý tưởng cốt lõi của LSTM.....	28
5.2 Cấu trúc chi tiết của GRU/LSTM	30
CHƯƠNG 3: THỰC NGHIỆM.....	31
1. Thu thập dữ liệu	31
2. Xử lý dữ liệu	33
3. Trực quan hoá dữ liệu	35
4. Xây dựng mô hình dự đoán	36
5. Tiến hành dự đoán, đánh giá.....	39
6. Mã nguồn hoàn chỉnh:	42
CHƯƠNG 4: KẾT LUẬN	43
1. Ưu điểm	43
2. Nhược điểm.....	43

3. Hướng phát triển	43
---------------------------	----

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC BẢNG HÌNH

Hình 1: Data Science là gì ?.....	1
Hình 2: 6 ứng dụng của Data Science.....	2
Hình 3: Mô hình Deep Learning.....	3
Hình 4: Lạm phát	4
Hình 5: Cấu trúc của một nơ ron sinh học	6
Hình 6: Cấu trúc đơn giản của một Perceptron	7
Hình 7: Cấu trúc của một mạng ANN	7
Hình 8: Cấu trúc đầy đủ của một Perceptron.....	8
Hình 9: Một số hàm kích hoạt thường dùng	9
Hình 10: Đồ thị hàm Sigmoid.....	10
Hình 11: Đồ thị hàm ReLU.....	11
Hình 12: Đồ thị hàm tanh	12
Hình 13: Cấu trúc của mạng nơ ron nhiều lớp.....	12
Hình 14: Cách chọn số lớp ẩn cho phù hợp.....	13
Hình 15: Hàm mất mát (Loss Function)	15
Hình 16: Lan truyền ngược	18
Hình 17: Quá trình suy giảm độ dốc đến khi tìm được cực tiểu.....	19
Hình 18: Cấu trúc mạng ANN truyền thống.....	20
Hình 19: Cấu trúc đơn giản của RNN.....	21
Hình 20: Cấu trúc cơ bản của RNN	21
Hình 21: Cấu trúc chi tiết mạng RNN	22
Hình 22: Công thức và đồ thị mô tả ba hàm kích hoạt thường dùng trong RNN.....	23
Hình 23: Cấu trúc chi tiết 1 nút của mạng RNN.....	23
Hình 24: Ví dụ về one to many.....	24
Hình 25: Các dạng bài toán RNN	24
Hình 26: Vanishing Gradient.....	26
Hình 27: Cấu trúc mạng RNN truyền thống.....	27

Hình 28: Cấu trúc mạng LSTM được cải tiến từ RNN	28
Hình 29: Mô tả dạng bảng chuyển thể hiện trạng thái tế bào trong LSTM.....	29
Hình 30: Cổng sàng lọc thông tin LSTM	29
Hình 31: Cấu trúc chi tiết của GRU và LSTM	30
Hình 32: Số liệu từ data.wordbank.org.....	31
Hình 33: Dữ liệu CPI Việt Nam	32
Hình 34: Mô tả dữ liệu trong file world_CPI.csv	33
Hình 35: Công thức hàm Minmaxscaler.....	36
Hình 36: Mô tả cách phân tập X và Y từ dữ liệu CPI.....	37

DANH MỤC CÁC TỪ VIẾT TẮT

ANN(Artificial Neural Network): mạng nơ ron nhân tạo

CPI(Consumer Price Index): chỉ số giá tiêu dùng

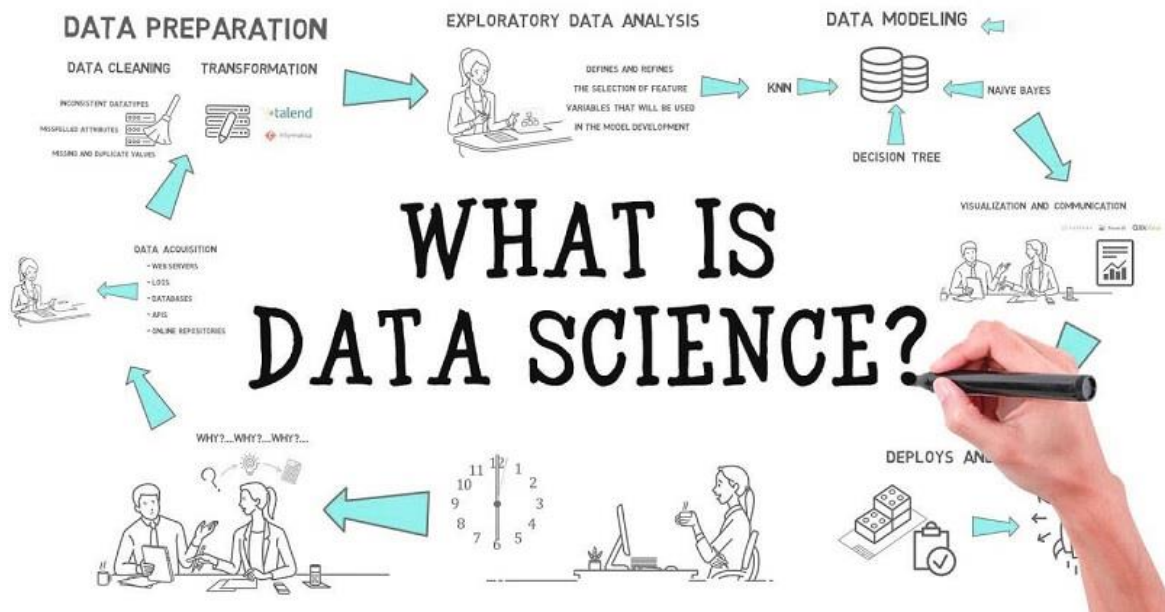
RNN(Recurrent Neural Network): mạng thần kinh nhân tạo

LSTM(Long short-term memory): mạng bộ nhớ dài ngắn

Chương 1: GIỚI THIỆU

1. Tổng quan về Khoa học dữ liệu

1.1 Khoa học dữ liệu là gì?



Hình 1: Data Science là gì ?

“**Khoa học dữ liệu (Data Science)** là một lĩnh vực liên ngành về các quá trình và các hệ thống rút trích tri thức hoặc hiểu biết từ dữ liệu ở các dạng khác nhau, kể ở dạng cấu trúc hay phi cấu trúc, là sự tiếp nối của một số lĩnh vực phân tích dữ liệu như khoa học thống kê, khai phá dữ liệu, tương tự như khám phá tri thức ở các cơ sở dữ liệu.” - Theo Wikipedia.

Hiểu một cách đơn giản, Khoa học dữ liệu là khoa học về việc quản trị và phân tích dữ liệu, trích xuất các giá trị từ dữ liệu để tìm ra các hiểu biết, các tri thức hành động, các quyết định dẫn dắt hành động.

1.2 Mối quan hệ giữa Khoa học dữ liệu và Trí tuệ nhân tạo?

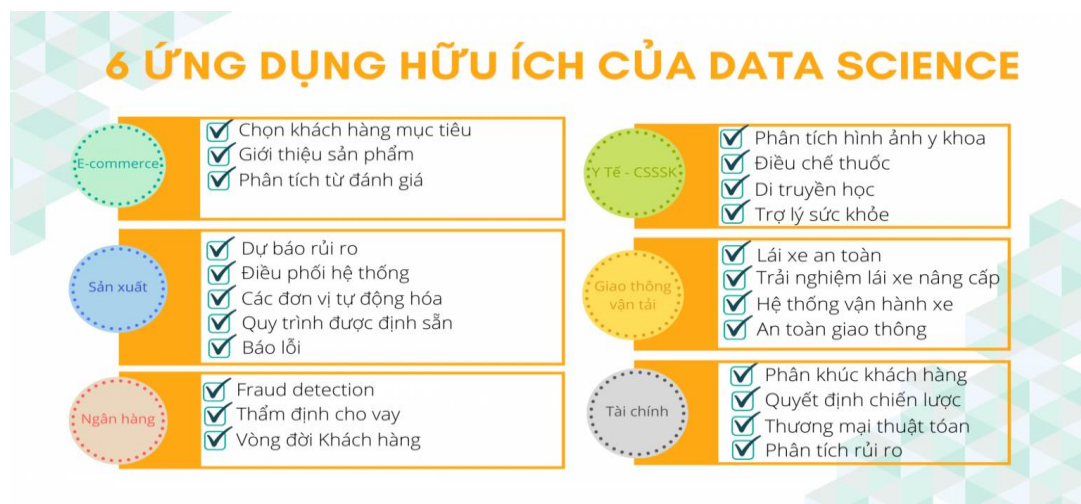
Giữa **Khoa học dữ liệu (DS)** và **Trí tuệ nhân tạo (AI)** luôn có một mối quan hệ đan xen lẫn nhau, tuy nhiên mỗi ngành sẽ có một đặc thù riêng.

Đối với **DS**, trọng tâm nghiên cứu là dữ liệu, dữ liệu có thể là thô chưa qua xử lý. Mục tiêu của người nghiên cứu là rút trích những thông tin có ý nghĩa, các đặc trưng quan trọng nhất từ tập dữ liệu từ đó đưa ra các biểu đồ trực quan hoá, phát triển các mô hình dự đoán trong tương lai sao cho người điều hành có những thông tin cần thiết để có thể có những quyết định có lợi cho công ty.

Ngược lại, **AI** cũng xoay quanh dữ liệu, tuy nhiên mục tiêu của nó là tạo ra một mô hình thuật toán có khả năng tư duy như con người và có thể biến các thuật toán đó thành chuỗi hành động trong môi trường thực tế do máy móc thao tác. Các vấn đề nghiên cứu của **AI** thường xoay quanh: Nhận diện vật thể, xử lý ngôn ngữ tự nhiên, nhận diện khuôn mặt,...

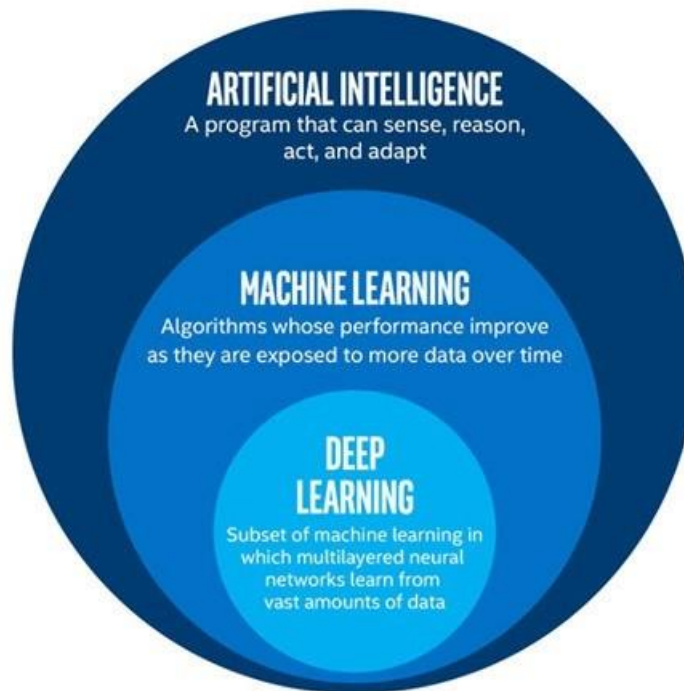
2. Ứng dụng của Khoa học dữ liệu

Data Science có tầm ảnh hưởng đến nền kinh tế, được ứng dụng đa dạng theo từng lĩnh vực. Từ y tế, tài chính, sản xuất, thương mại điện tử,.. Và còn nhiều lĩnh vực khác nữa. Sau đây là 6 lĩnh vực có thể ứng dụng những khả năng của **DS**.



Hình 2: 6 ứng dụng của Data Science

3. Deep learning là gì?



Hình 3: Mô hình Deep Learning

Deep Learning là một nhánh Machine Learning sử dụng nhiều lớp Neural Network để đưa ra một mô hình toán học trên dữ liệu có sẵn. Học sâu hay Deep Learning thường được nhắc đến cùng với Dữ liệu lớn (Big Data) và Trí tuệ nhân tạo (AI) đã có nhiều ứng dụng trong thực tế, đang phát triển mạnh theo sự phát triển của tốc độ máy tính đặc biệt là khả năng tính toán trên GPU và sự tăng nhanh của dữ liệu cùng với các framework giúp khả năng làm việc xây dựng model trở nên dễ dàng hơn.

4. Ứng dụng của KHDL trong giải quyết vấn đề lạm phát

4.1 Lạm phát là gì?

Lạm phát là sự tăng mức giá chung một cách liên tục của hàng hóa và dịch vụ theo thời gian và sự mất giá trị của một loại tiền tệ nào đó. Khi mức giá chung tăng cao, một đơn vị

tiền tệ sẽ mua được ít hàng hóa và dịch vụ hơn so với trước đây, do đó lạm phát phản ánh sự suy giảm sức mua trên một đơn vị tiền tệ.



Hình 4: Lạm phát

4.2 Ảnh hưởng của lạm phát đối với nền kinh tế

Tích cực:

Kích thích tiêu dùng, vay nợ, đầu tư, giảm bớt thất nghiệp trong xã hội.

Cho phép chính phủ có thêm khả năng lựa chọn các công cụ kích thích đầu tư vào một số lĩnh vực kém ưu tiên thông qua việc mở rộng tín dụng, giúp phân phối lại thu nhập và các nguồn lực trong xã hội theo các định hướng mục tiêu trong khoảng thời gian nhất định có chọn lọc.

Tiêu cực:

Ảnh hưởng đến lãi suất dẫn đến hậu quả là nền kinh tế có thể bị suy thoái và thất nghiệp gia tăng.

Làm cho thu nhập thực tế của người lao động giảm xuống. lạm phát làm giảm thu nhập thực thông qua các khoản lãi và các khoản lợi tức.

Lạm phát tăng cao còn khiến cho những người thừa tiền vợ vét và thu gom hàng hoá, tài sản, vẩn nạn đầu cơ xuất hiện. Tình trạng ngày càng làm mất cân đối nghiêm trọng quan hệ cung – cầu hàng hoá trên thị trường, giá cả hàng hoá lại càng lên cao hơn. Và đương nhiên, những người dân nghèo sẽ trở nên khốn khó hơn khi không mua được những hàng hóa thiết yếu. Tạo ra một khoảng cách ngày càng lớn giữa người nghèo và người giàu.

4.3 Thực trạng vấn đề lạm phát ở Việt Nam trong những năm gần đây

Quỹ Tiền tệ Quốc tế đã nâng dự báo lạm phát toàn cầu năm 2022 lên mức 5,7% ở các nền kinh tế phát triển và 8,7% ở các nền kinh tế mới nổi và đang phát triển. Áp lực lạm phát tăng cao tại các nền kinh tế lớn và là đối tác thương mại hàng đầu của Việt Nam như CPI Mỹ tháng 5/2022 tăng 8,6% so với cùng kỳ năm 2021 tiếp tục ở mức cao nhất trong vòng 40 năm qua, CPI của khu vực đồng euro tháng 5/2022 tăng 8,1% gấp 4 lần so với lạm phát mục tiêu 2% của Ngân hàng trung ương châu Âu. Tại châu Á, CPI tháng 5/2022 so với cùng kỳ năm trước của một số nước như Hàn Quốc tăng 5,4%, Thái Lan tăng 7,1%, Indonesia tăng 3,55%...

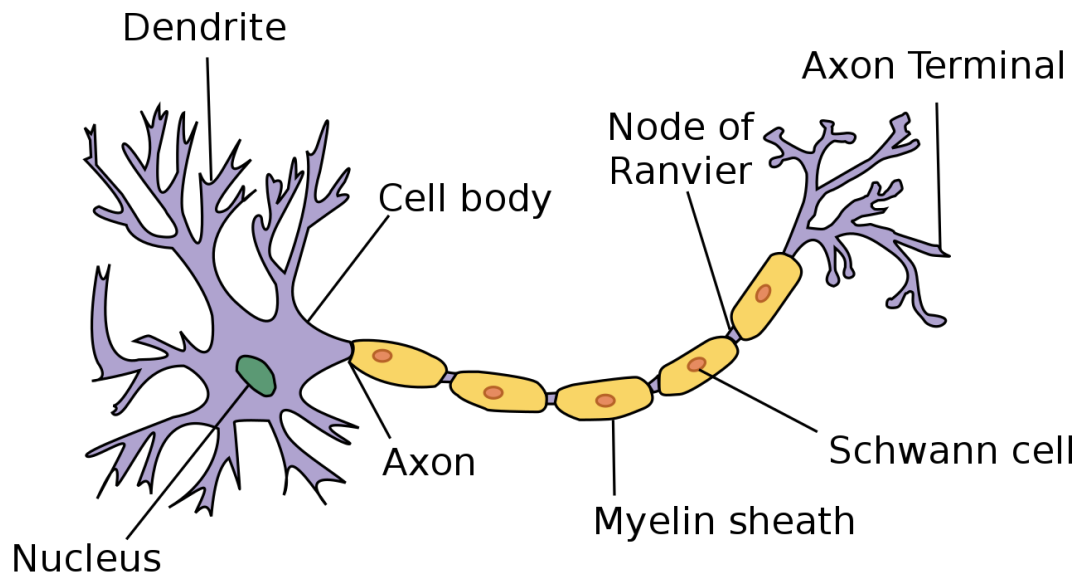
Nhìn chung ở Việt Nam những năm gần đây, tỷ lệ lạm phát vẫn ở mức ổn định và kiểm soát được. Trong 6 tháng đầu năm 2022, CPI Việt Nam chỉ tăng 2,44% và lạm phát cơ bản tăng 1,25% là tương đối thấp, trong khi lạm phát ở nhiều quốc gia trên thế giới cao kỷ lục hàng chục năm và giá cả nhiều mặt hàng như xăng dầu, nguyên nhiên vật liệu đầu vào của nền sản xuất tăng cao.

Chương 2: CƠ SỞ LÝ THUYẾT

1. Mạng thần kinh (Neuron Network)

1.1 Mạng nơ-ron sinh học

Nguồn gốc ý tưởng của việc nghiên cứu trí thông minh nhân tạo được bắt nguồn từ việc khám phá ra cấu trúc mạng nơ-ron sinh học.



Hình 5: Cấu trúc của một nơ-ron sinh học

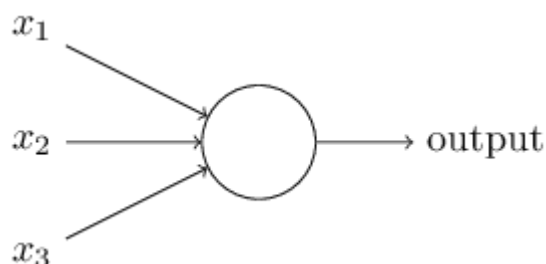
Trong hệ thống thần kinh sinh học, nơ-ron là tế bào sống và còn là đơn vị lưu trữ cơ bản trong bộ não của con người. Có khoảng 200 tỉ nơ-ron trong bộ não con người. Mỗi nơ-ron được liên kết với khoảng từ 1.000 đến 10.000 nơ-ron khác thông qua các khớp thần kinh (synapse). Các tín hiệu xung điện được truyền từ tế bào nơ-ron này sang tế bào nơ-ron khác thông qua các khớp thần kinh. Có tất cả khoảng 125 nghìn tỉ khớp thần kinh trong bộ não của con người.

Từ đây, các nhà nghiên cứu đã tìm ra cách mô hình hoá mạng nơ-ron này trên máy tính và cho ra đời mô hình **mạng nơ-ron nhân tạo (Artificial Neural Network)**.

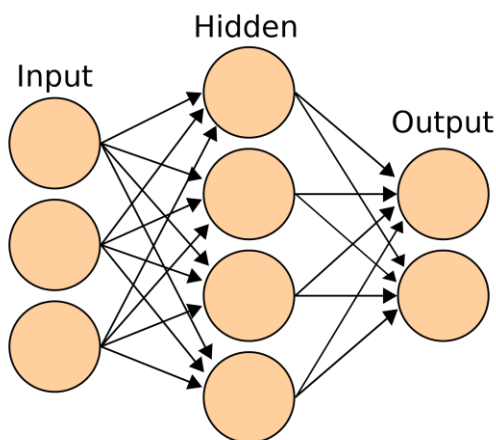
1.2 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (ANN) là một mô hình toán học hay mô hình tính toán được xây dựng dựa trên các mạng nơ-ron sinh học. Nó gồm có một nhóm các nơ-ron nhân tạo (nút) nối với nhau, và xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút.

Các nút trên mạng nơ-ron nhân tạo được gọi là **Perceptron**.



Hình 6: Cấu trúc đơn giản của một Perceptron



Hình 7: Cấu trúc của một mạng ANN

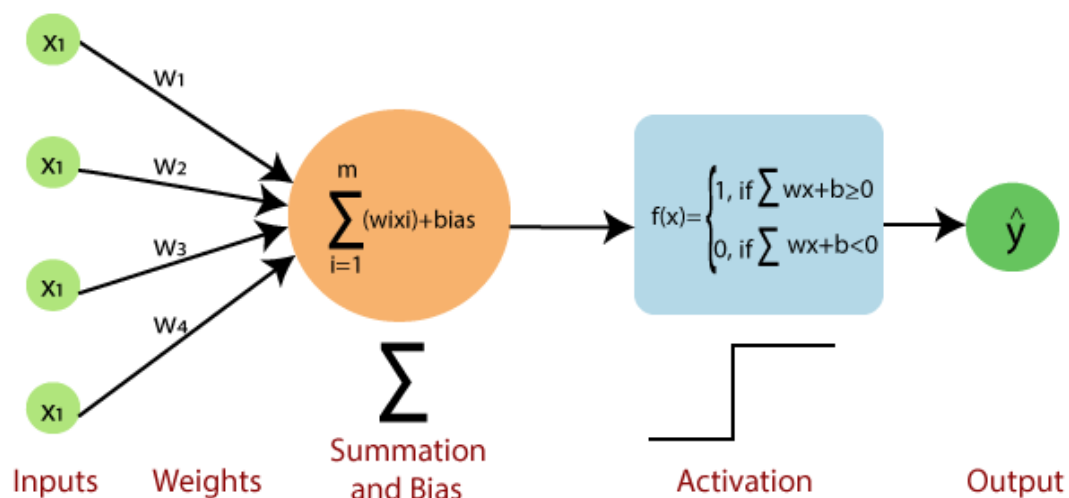
ANN có một số ưu điểm nhưng một trong những ưu điểm nổi bật nhất là một thực tế là nó thực sự *có thể học hỏi từ việc quan sát các tập dữ liệu*. Theo cách này, ANN được sử dụng như một hàm ngẫu nhiên công cụ xấp xỉ. Các loại công cụ này giúp ước tính các phương pháp hiệu quả nhất về mặt chi phí và lý tưởng để đến các giải pháp trong khi xác định các chức năng tính toán hoặc phân phối. ANN lấy mẫu dữ liệu thay vì toàn bộ tập dữ liệu để

đến các giải pháp, giúp tiết kiệm cả thời gian và tiền bạc. ANN được coi là các mô hình toán học khá đơn giản để nâng cao các công nghệ phân tích dữ liệu hiện có.

2. Cấu trúc mạng nơ-ron

2.1 Perceptron là gì?

Perceptrons được phát triển vào khoảng những năm 1950 – 1960 bởi nhà khoa học *Frank Rosenblatt* và được lấy cảm hứng từ những ý tưởng trước đó của *Warren McCulloch* và *Walter Pitts*. Ngày nay, nó phổ biến trong nhiều mô hình mạng nơ-ron khác nhau. Một perceptron có một số đầu vào (input) nhị phân, x_1, x_2, \dots , và tạo ra một đầu ra (output) nhị phân duy nhất.



Hình 8: Cấu trúc đầy đủ của một Perceptron

Cấu trúc bao gồm 3 thành phần:

Hệ thống ghép nối thần kinh (synapse)

Bộ cộng

Hàm kích hoạt

Một perceptron có thể bao gồm một hoặc nhiều đầu vào, mỗi **đầu vào (input)** là một giá trị x_n kèm theo **trọng số (weight)** của nó là w_n . Các giá trị này sẽ được đưa vào bộ cộng. Sau đó, bộ cộng sẽ làm nhiệm vụ lấy tổng của tích hai tham số **đầu vào(x)** và **trọng số(w)** sau đó cộng thêm một tham số **bias(*)** theo công thức :

$$z = x_1 * w_1 + x_2 * w_2 + \dots + \text{bias}$$

Cuối cùng, kết quả z thu được sẽ được đưa vào **hàm kích hoạt** để tính ra **giá trị đầu ra (output)** của perceptron. Thường sẽ là **Sigmoid** hoặc **ReLU**.

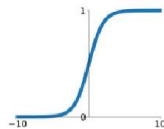
Bias: nghĩa là *độ lệch*, biểu thị *sự chênh lệch* giữa giá trị trung bình mà mô hình dự đoán và giá trị thực tế của dữ liệu. Đối với mạng nơ-ron đơn giản con số này có thể lấy ngẫu nhiên sau đó ta sẽ điều chỉnh dần bằng thuật toán.

2.2 Hàm kích hoạt

Hàm kích hoạt (activation function) là *một phép biến đổi phi tuyến tính* mà chúng ta thực hiện đối với tín hiệu đầu vào. Đầu ra được chuyển đổi này sẽ được sử dụng làm đầu vào của nơ-ron ở layer tiếp theo. Nếu không có hàm kích hoạt thì **trọng số (w)** và **bias** chỉ đơn giản như một hàm biến đổi tuyến tính. Giải một hàm tuyến tính sẽ đơn giản hơn nhiều nhưng sẽ khó có thể mô hình hóa và giải được những vấn đề phức tạp

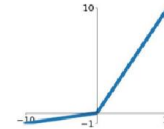
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



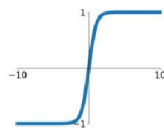
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

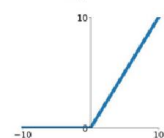


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

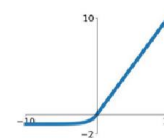
ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Hình 9: Một số hàm kích hoạt thường dùng

Mặc dù, trong thực tế còn nhiều *hàm kích hoạt* khác nữa, tuy nhiên phổ biến nhất vẫn là 3 hàm chính: **Sigmoid**, **ReLU**, **tanh**. Vì vậy chúng ta sẽ đi vào chi tiết 3 hàm này.

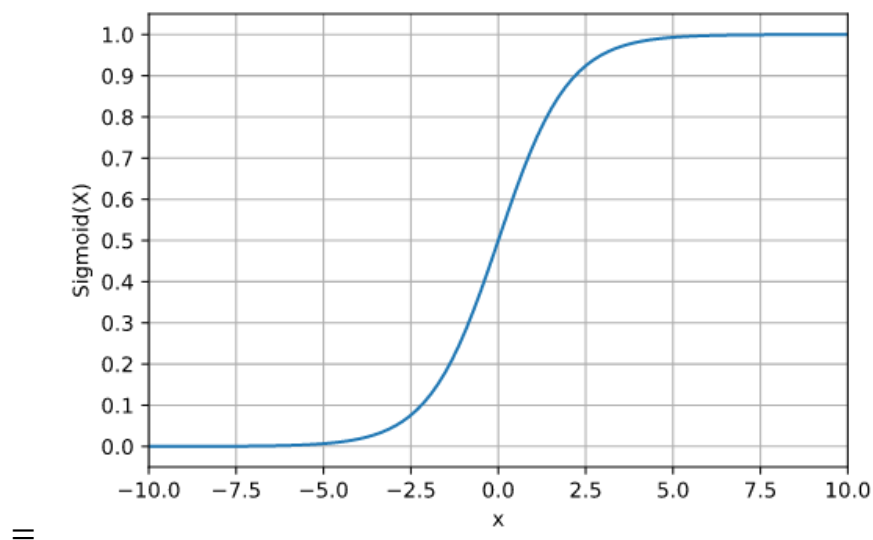
2.2.1 Hàm Sigmoid

Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0,1). Thường được dùng trong các bài toán đầu ra 2 output.

Công thức:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Đồ thị:



Hình 10: Đồ thị hàm Sigmoid

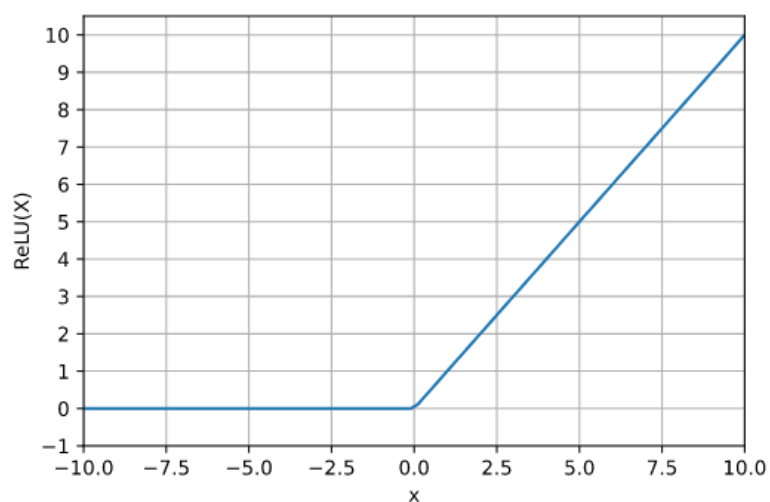
2.2.2 Hàm ReLU

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0 .

Công thức:

$$f(x) = \max(0, x)$$

Đồ thị:



Hình 11: Đồ thị hàm ReLU

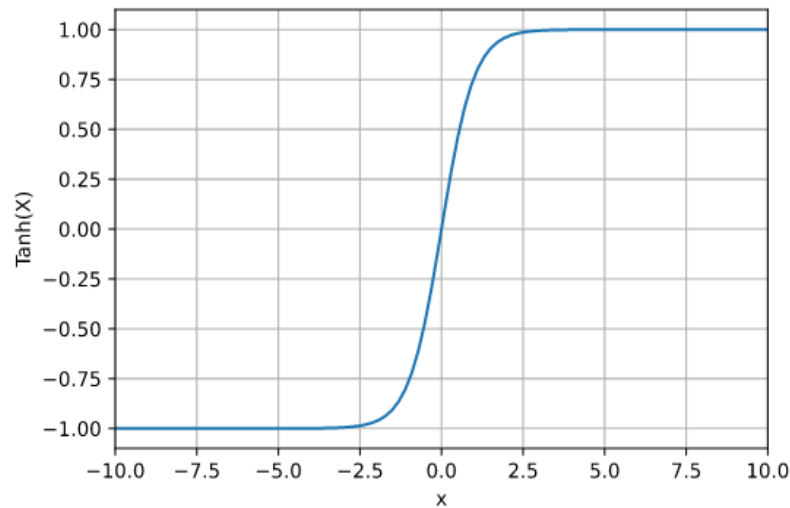
2.2.3. Hàm tanh

Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong *khoảng* $(-1; 1)$

Công thức:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Đồ thị:

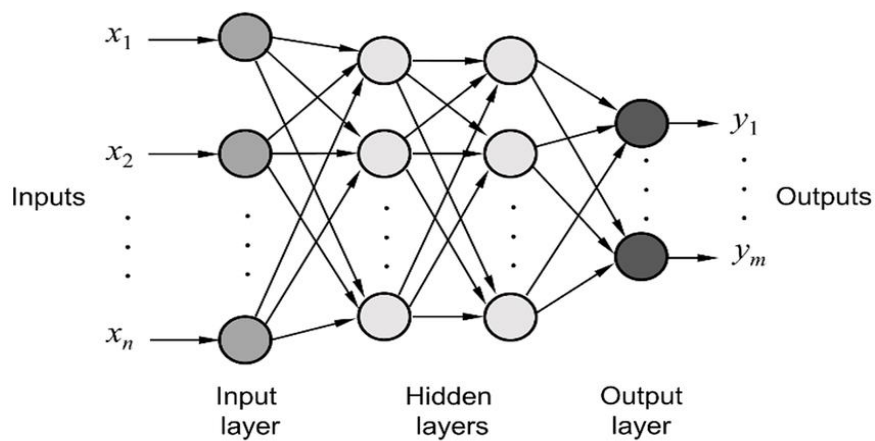


Hình 12: Đồ thị hàm tanh

Như vậy, chúng ta đã tìm hiểu qua cấu trúc cơ bản của một mạng neuron một lớp đơn giản. Tiếp theo, ta sẽ đi đến mạng neuron nhiều lớp, có cơ chế phức tạp hơn một chút.

2.3 Mạng nơ-ron nhiều lớp.

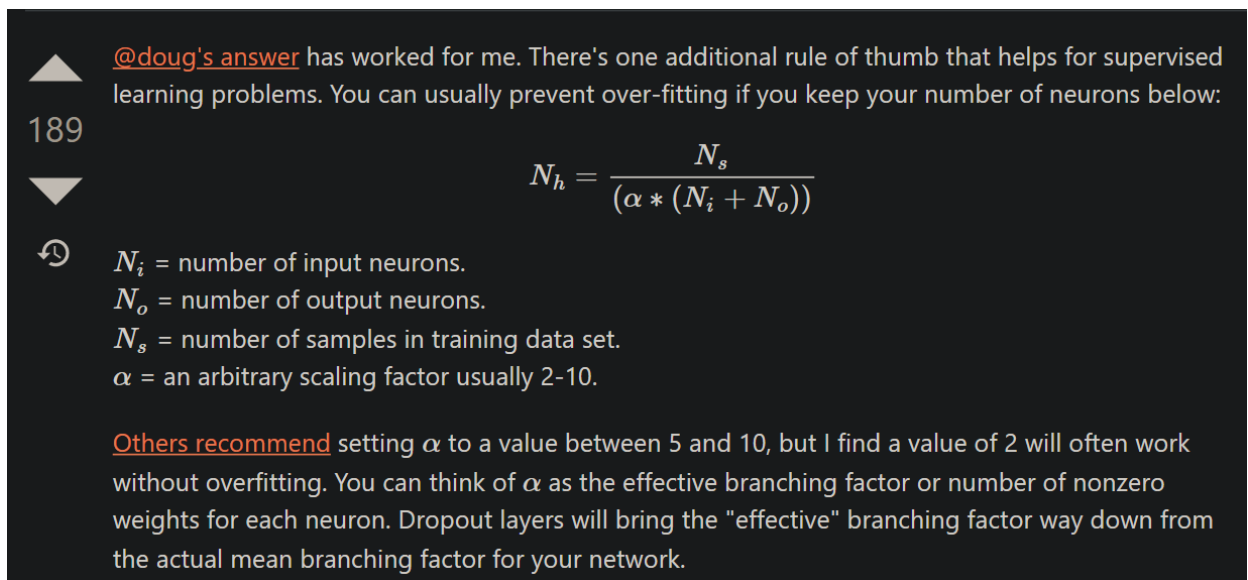
Mạng nơ-ron nhiều lớp (multilayer neural network), về cơ bản nó là sự kết hợp của nhiều perceptron liên kết với nhau, được phân chia thành 3 lớp (layer) chính: *lớp đầu vào (input layer)*, *lớp ẩn (hidden layer)*, *lớp đầu ra (output layer)*.



Hình 13: Cấu trúc của mạng nơ ron nhiều lớp

Trong đó, số lượng nút của lớp đầu vào (*input layer*) sẽ tương ứng với số lượng tham số được đưa vào mạng. Số lượng nút của lớp đầu ra (*output layer*) sẽ là số lượng giá trị trả về mà ta mong muốn.

Đối với lớp ẩn (*hidden layer*), trong một mạng có thể có từ một đến nhiều lớp ẩn tùy vào mục đích của người thiết kế. Số lượng nút của lớp ẩn được chọn dựa theo kinh nghiệm của người nghiên cứu mạng, tuy vậy số lượng đó ta có thể tìm ra bằng cách áp dụng công thức sau.



▲ @doug's answer has worked for me. There's one additional rule of thumb that helps for supervised learning problems. You can usually prevent over-fitting if you keep your number of neurons below:

189 ▼

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

↻ N_i = number of input neurons.
 N_o = number of output neurons.
 N_s = number of samples in training data set.
 α = an arbitrary scaling factor usually 2-10.

Others recommend setting α to a value between 5 and 10, but I find a value of 2 will often work without overfitting. You can think of α as the effective branching factor or number of nonzero weights for each neuron. Dropout layers will bring the "effective" branching factor way down from the actual mean branching factor for your network.

Hình 14: Cách chọn số lớp ẩn cho phù hợp

3. Cách thức hoạt động của mạng nơ-ron

3.1 Lan truyền thẳng

Lan truyền thẳng (Feedforward) trong mạng Nơ-ron nhân tạo cũng là sự kết hợp tuyến tính (hay tích vô hướng) của các giá trị input đầu vào với bộ trọng số tương ứng, các phép tính này được thực hiện trên các ma trận và véc-tơ.

Như đã biết, một mạng nơ-ron nhân tạo cơ bản gồm có 3 lớp: đầu vào (*input*), lớp giữa (*hidden*) và lớp đầu ra (*output*), kết quả lan truyền từ trái sang phải hay từ *input* →

$hidden \rightarrow output$ được gọi là **lan truyền thẳng**, còn ngược lại từ **phải sang trái** (hay từ $output \rightarrow hidden \rightarrow input$) được gọi là **lan truyền ngược**.

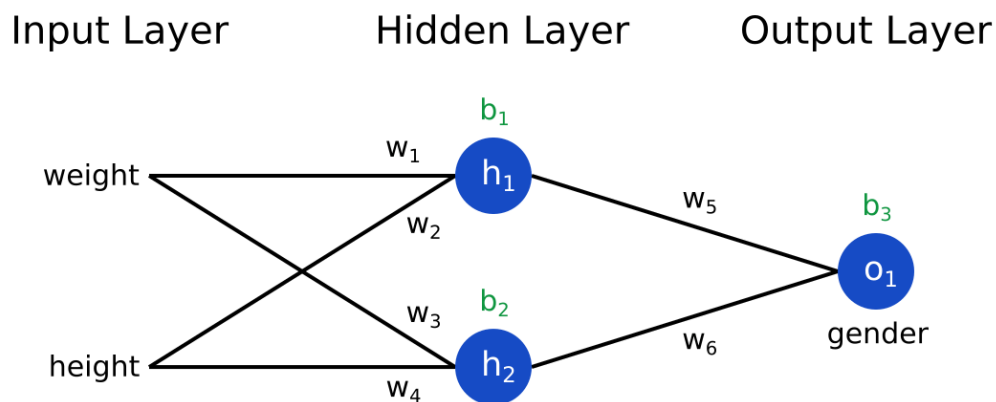
Nguyên lý hoạt động:

Như ta đã biết, một perceptron xử lý dữ liệu đầu vào bằng cách tính theo công thức sau:

$$z = x_1 * w_1 + x_2 * w_2 + \dots + \text{bias}$$

Sau đó sẽ tính $f(z)$, với $f()$ là một hàm kích hoạt, từ đây $f(z)$ sẽ là đầu ra (output) của perceptron. Lan truyền thẳng sử dụng cơ chế này, tuy nhiên khác với mạng perceptron một lớp thì mạng nhiều lớp sẽ tiếp tục lấy kết quả đầu ra (output) của layer đó để tính tiếp kết quả đầu ra (output) của layer tiếp theo cho đến khi hết lớp của mạng.

Ví dụ ta có mạng như sau:



Để dễ hình dung ta sẽ quy ước giá trị input đầu vào là: i_1 (**weight**), i_2 (**height**).

Giá trị h_1 sẽ được tính bằng: $h_1 = f(i_1 * w_1 + i_2 * w_2 + b_1) = f(z_1)$

Giá trị h_2 sẽ được tính bằng: $h_2 = f(i_1 * w_3 + i_2 * w_4 + b_2) = f(z_2)$

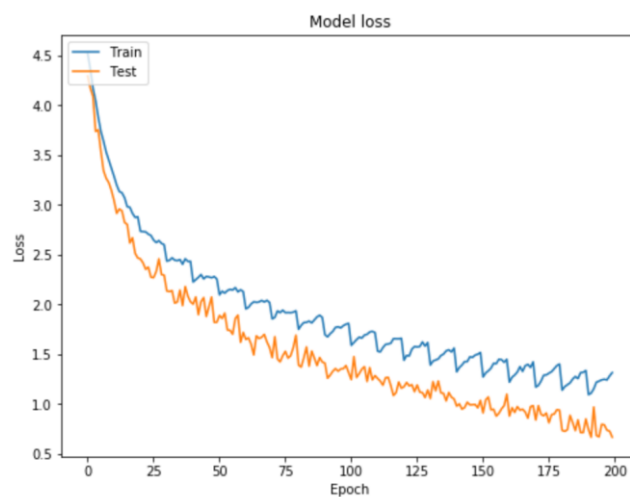
Giá trị o_1 sẽ được tính bằng: $o_1 = f(h_1 * w_5 + h_2 * w_6 + b_3) = f(z_0)$

Như vậy kết quả đầu ra cuối cùng sẽ là $f(z_0)$ đây chính là **kết quả dự đoán** của mạng, những bước trên đã giúp ta có được kết quả cuối này, đó chính là **lan truyền thẳng**. Sau

khi đã có kết quả đầu ra cuối cùng của mạng, chúng ta cần phải so sánh nó với **kết quả kỳ vọng**, vì vậy **hàm mất mát** ra đời để làm việc đó.

3.2 Hàm mất mát (Loss Function)

“Trong tối ưu hóa toán học và lý thuyết quyết định, **một hàm mất mát** (hay còn gọi là hàm tổn thất, tiếng Anh: loss function) là một hàm ánh xạ một biến cố trong xác suất hoặc các giá trị của một hay nhiều biến thành một số thực có tính trực giác thể hiện một vài "chi phí" (cost) liên quan đến sự kiện.” - Theo Wikipedia.



Hình 15: Hàm mất mát (Loss Function)

Chúng ta có thể hiểu đơn giản, một hàm mất mát ở đây sẽ đóng vai trò đánh giá độ “tốt” của mô hình với một bộ trọng số tương ứng. Mục đích của quá trình huấn luyện là tìm ra bộ số để độ lớn hàm mất mát (loss function) là nhỏ nhất (cực tiểu), tức độ lớn của nó càng nhỏ càng tốt. Như vậy ta có thể coi hàm mất mát là hàm mục tiêu trong quá trình huấn luyện.

3.3 Một số giải thuật hàm mất mát thường dùng:

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

Mean Squared Error (MSE)

$$\text{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Sau khi tính toán độ mất mát của *kết quả dự đoán* và *kết quả kỳ vọng*. Chúng ta có được độ lệch sai số của mô hình, từ số liệu sai số này chúng ta sẽ có cơ sở để điều chỉnh mô hình, sao cho mạng ngày càng thông minh hơn. Đó chính là nhiệm vụ của **lan truyền ngược**.

3.4 Lan truyền ngược

Lan truyền ngược là một phương pháp phổ biến để huấn luyện các mạng thần kinh nhân tạo được sử dụng kết hợp với một phương pháp tối ưu hóa như **Gradient Descent**. Phương pháp này tính toán độ dốc của hàm tổn thất với tất cả các trọng số có liên quan trong mạng nơ ron đó. Độ dốc này được đưa vào phương pháp tối ưu hóa, sử dụng nó để cập nhật các trọng số, để cực tiểu hóa hàm tổn thất.

Thuật toán học truyền ngược có thể được chia thành hai giai đoạn: lan truyền, cập nhật trọng số:

Giai đoạn 1: Lan truyền

Lan truyền thuận của một đầu vào của mô hình huấn luyện thông qua mạng nơ-ron để tạo ra các kích hoạt đầu ra của lan truyền này.

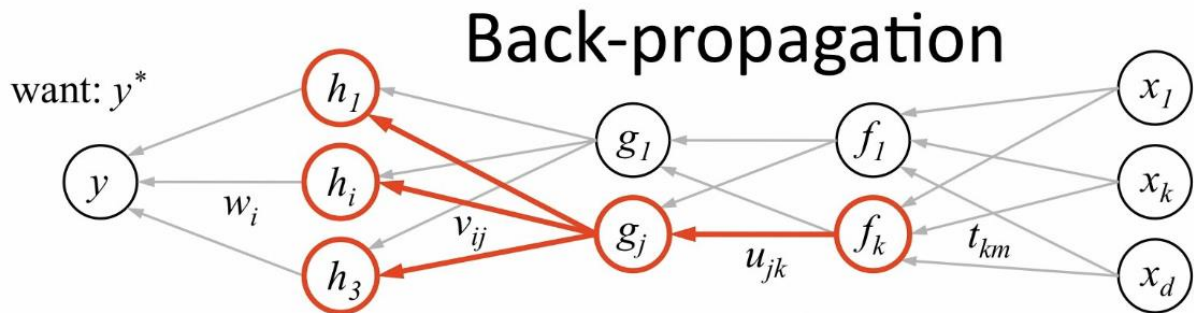
Truyền ngược của các kích hoạt đầu ra của lan truyền thông qua mạng lưới neural sử dụng mục tiêu huấn luyện mô hình để tạo ra các delta (sai lệch giữa giá trị mục tiêu và giá trị đầu ra thực tế) và tất cả đầu ra và các nơ-ron ẩn.

Giai đoạn 2: Cập nhật trọng số

Nhân các delta đầu ra và kích hoạt đầu vào để có được gradient của trọng số của nó.

Trừ một tỷ lệ (tỷ lệ phần trăm) từ gradient của trọng số.

Tỷ lệ này (tỷ lệ phần trăm) ảnh hưởng đến tốc độ và chất lượng học; nó được gọi là **tốc độ học (learning rate)**. Tỷ lệ này càng lớn, thì tốc độ huấn luyện nơ-ron càng nhanh; tỷ lệ này càng thấp, thì việc huấn luyện càng chậm. Dấu của gradient của một trọng số chỉ ra chỗ mà sai số đang gia tăng, đây là lý do tại sao trọng số phải được cập nhật theo hướng ngược lại.



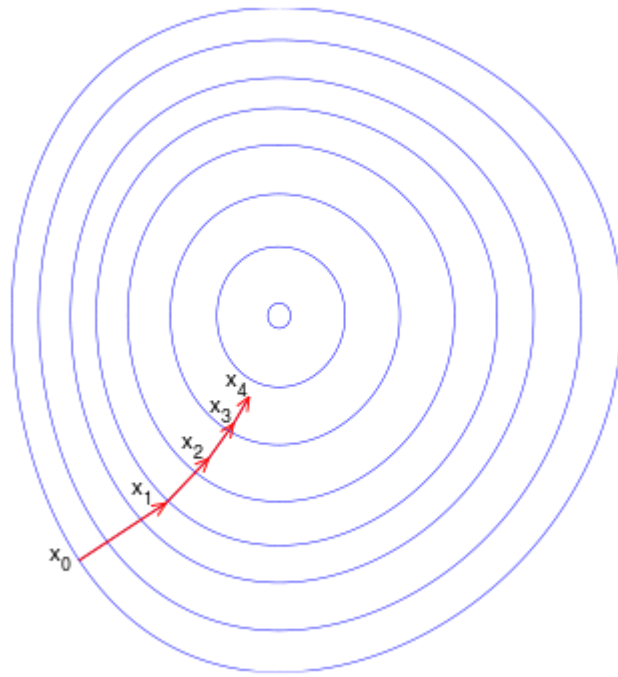
1. receive new observation $\mathbf{x} = [x_1 \dots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer $1 \dots L$
compute g_j based on units f_k from previous layer: $g_j = \sigma \left(u_{j0} + \sum_k u_{jk} f_k \right)$
3. get prediction y and error $(y - y^*)$
4. **back-propagate error:** for each unit g_j in each layer $L \dots 1$

<p>(a) compute error on g_j</p> $\frac{\partial E}{\partial g_j} = \sum_i \underbrace{\sigma'(h_i)}_{\substack{\text{should } g_j \\ \text{be higher} \\ \text{or lower?}}} \underbrace{v_{ij}}_{\substack{\text{how } h_i \text{ will} \\ \text{change as} \\ g_j \text{ changes}}} \underbrace{\frac{\partial E}{\partial h_i}}_{\substack{\text{was } h_i \text{ too} \\ \text{high or} \\ \text{too low?}}}$	<p>(b) for each u_{jk} that affects g_j</p> <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>(i) compute error on u_{jk}</p> $\frac{\partial E}{\partial u_{jk}} = \frac{\partial E}{\partial g_j} \underbrace{\sigma'(g_j)}_{\substack{\text{do we want } g_j \text{ to} \\ \text{be higher/lower}}} \underbrace{f_k}_{\substack{\text{how } g_j \text{ will change} \\ \text{if } u_{jk} \text{ is higher/lower}}}$ </div> <div style="width: 45%;"> <p>(ii) update the weight</p> $u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$ </div> </div>
---	---

Hình 16: Lan truyền ngược

3.5 Gradient Descent

Gradient Descent (GD) là một thuật toán tối ưu hóa lặp bậc nhất để tìm một cực trị của một hàm khả vi. Để tìm cực tiểu cục bộ của một hàm sử dụng suy giảm độ dốc, người ta có thể thực hiện các bước tỷ lệ thuận với âm của gradient (hoặc độ dốc xấp xỉ) của hàm tại điểm hiện tại. Hiểu đơn giản Gradient descent là thuật toán tìm **giá trị nhỏ nhất của hàm số $f(\mathbf{x})$** dựa trên đạo hàm. Mục đích của nó trong học máy là để tối ưu mô hình sao cho **sai số dự đoán (loss function)** được tính ở đầu ra là nhỏ nhất.



Hình 17: Quá trình suy giảm độ dốc đến khi tìm được cực tiểu

Thuật toán:

Bước 1: Khởi tạo giá trị $x = x_0$ tùy ý

Bước 2: Gán $x = x - \text{learning_rate} * f'(x)$ (learning_rate là hằng số không âm ví dụ $\text{learning_rate} = 0.001$)

Bước 3: Tính lại $f(x)$

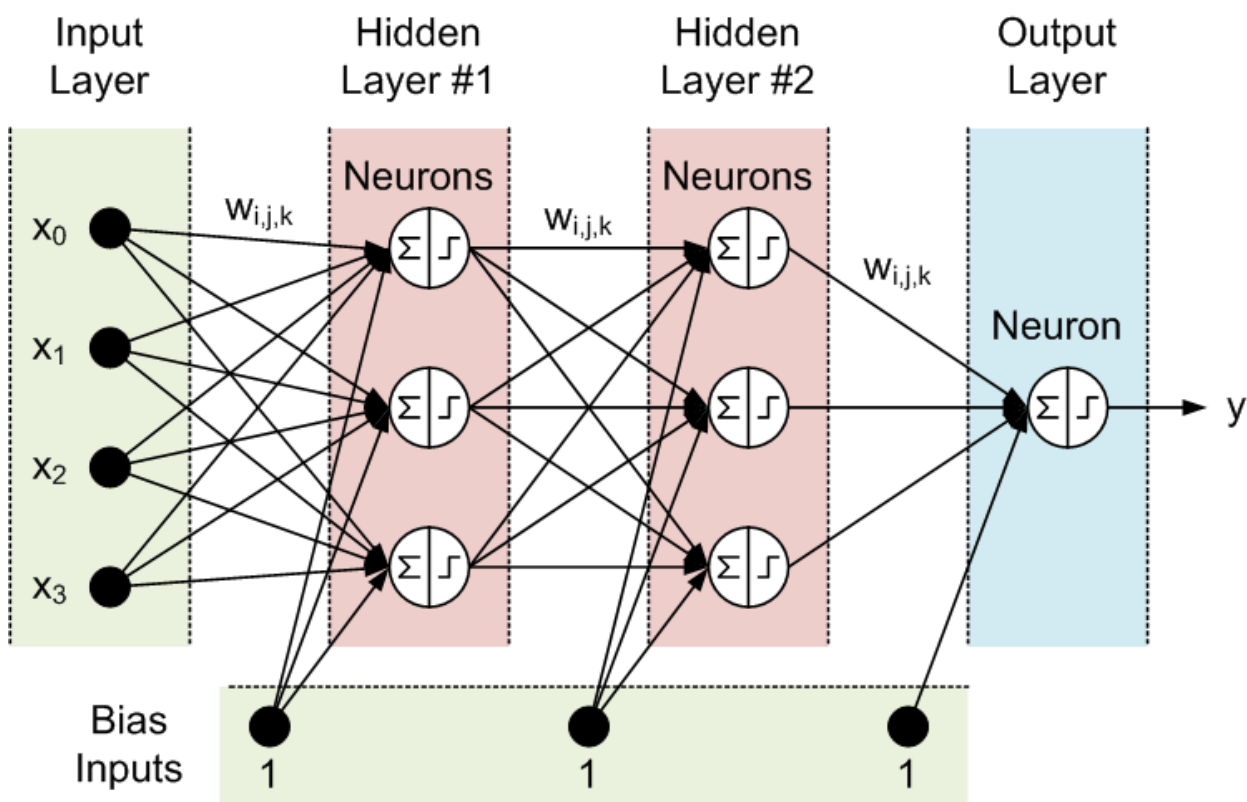
Nếu $f(x)$ đủ nhỏ thì dừng lại.

Ngược lại tiếp tục **Bước 2**.

4. Mạng thần kinh hồi quy (RNN)

Mạng thần kinh hồi quy (Recurrent Neural Network) là một lớp của mạng thần kinh nhân tạo, nơi kết nối giữa các nút để tạo thành đồ thị có hướng dọc theo một trình tự thời gian. Điều này cho phép mạng thể hiện hành vi động tạm thời.

Để có thể hiểu rõ về RNN, trước tiên chúng ta cùng nhìn lại mô hình Neural Network dưới đây:



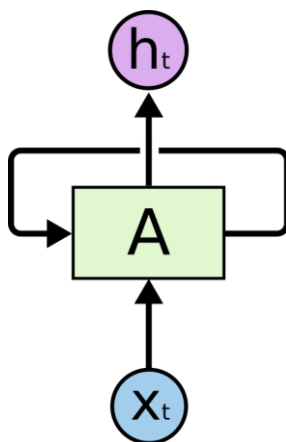
Hình 18: Cấu trúc mạng ANN truyền thống

Như đã biết, mạng nơ-ron bao gồm 3 phần chính: *lớp đầu vào*, *lớp ẩn* và *lớp đầu ra*, chúng ta có thể thấy rằng *đầu vào* và *đầu ra* của mạng nơ-ron này độc lập với nhau. Vì vậy, mô hình này không phù hợp với các bài toán có **dữ liệu liên tục móc nối nhau (sequence data)** như về chuỗi: mô tả, hoàn thành câu, v.v., bởi vì các dự đoán tiếp theo như từ tiếp theo phụ thuộc vào *vị trí của nó trong câu* và *các từ trước nó*.

Và thế là **RNN** ra đời với ý tưởng chính là dùng *một bộ nhớ để lưu trữ thông tin* từ các bước tính toán trước đó để dựa vào đó đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại.

4.1 Cấu trúc mạng RNN truyền thống

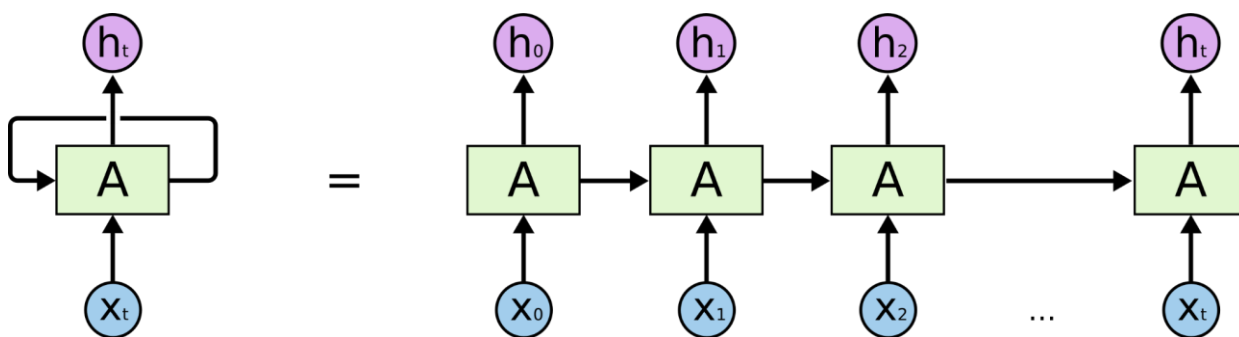
Mạng này chứa các vòng lặp bên trong giúp lưu thông tin.



Hình 19: Cấu trúc đơn giản của RNN

Hình trên đại diện cho một đoạn của mạng nơ-ron hồi quy **A** với đầu vào \mathbf{x}_t và đầu ra \mathbf{h}_t . Một vòng lặp được sử dụng để truyền thông tin từ bước này sang bước khác trong mạng nơ-ron.

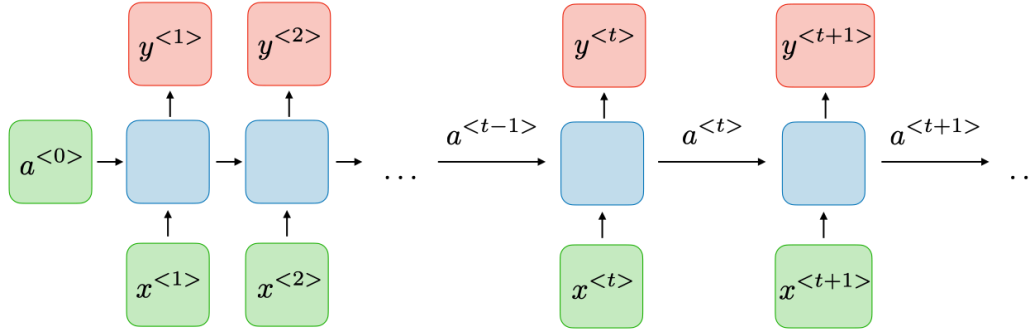
Các vòng lặp này khiến cho mạng nơ-ron hồi quy trông có vẻ khó hiểu. Tuy nhiên, nếu bạn để ý kỹ, nó không khác nhiều so với mạng nơ-ron thuần túy. Một mạng nơ-ron tuần hoàn có thể được coi là nhiều bản sao của cùng một mạng, trong đó mỗi đầu ra của mạng này là đầu vào của mạng khác.



Hình 20: Cấu trúc cơ bản của RNN

Chuỗi lặp lại các mạng này chính là phân giải của mạng nơ-ron hồi quy, các vòng lặp khiến chúng tạo thành một chuỗi danh sách các mạng sao chép nhau.

Các mạng neural hồi quy, còn được biến đến như là RNNs, là một lớp của mạng neural cho phép đầu ra được sử dụng như đầu vào trong khi có các trạng thái ẩn. Thông thường là như sau:



Hình 21: Cấu trúc chi tiết mạng RNN

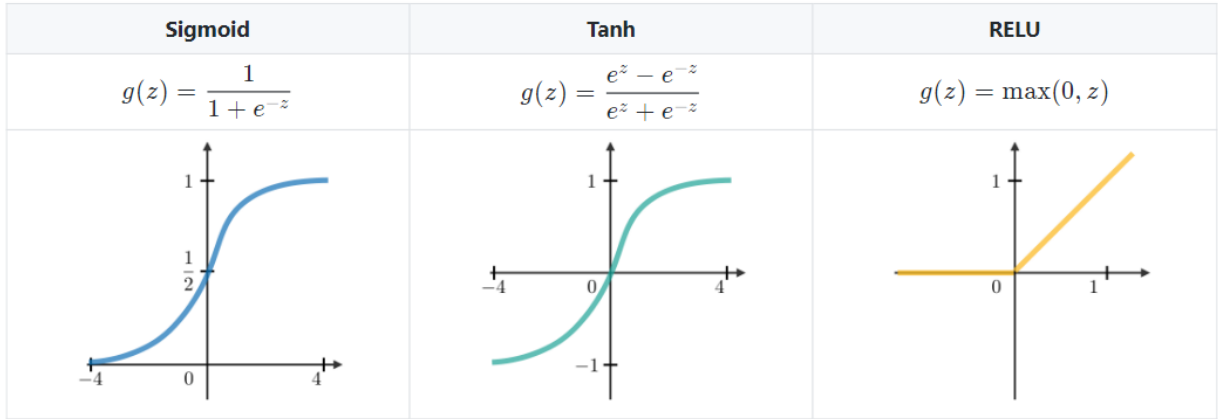
Tại mỗi bước t , giá trị kích hoạt $a^{<t>}$ và đầu ra $x^{<t>}$ được biểu diễn như sau:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

và

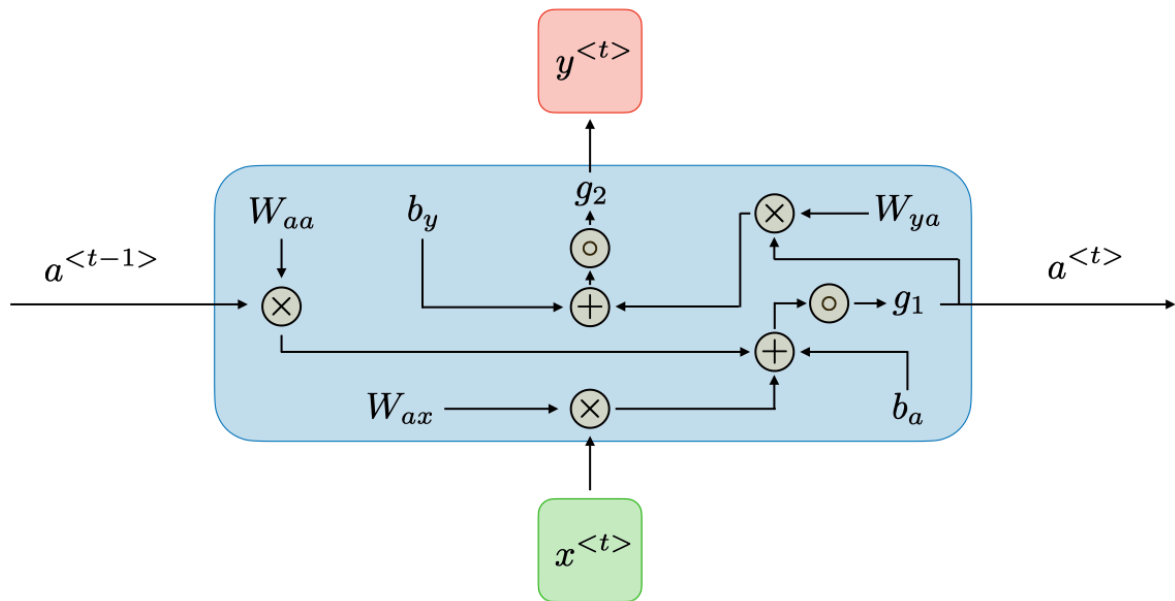
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Với W_{ax} , W_{aa} , W_{ya} , b_a , b_y , là các hệ số được chia sẻ tạm thời và g_1 , g_2 là các hàm kích hoạt. Các hàm kích hoạt thường dùng là **Sigmoid**, **Tanh**, **RELU**.



Hình 22: Công thức và đồ thị mô tả ba hàm kích hoạt thường dùng trong RNN

4.2 Cấu trúc chi tiết



Hình 23: Cấu trúc chi tiết 1 nút của mạng RNN

4.3 Các dạng bài toán RNN

One to one: có 1 input và 1 output, mạng neural truyền thống, ví dụ với input là ảnh động vật và output là phân loại ảnh đó có phải một con cá hay không.

One to many: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."

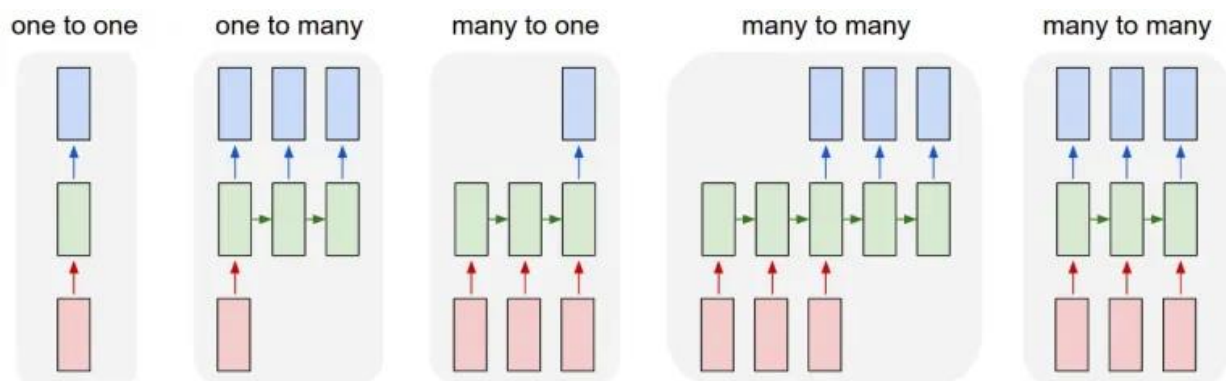


"two young girls are playing with lego toy."

Hình 24: Ví dụ về one to many

Many to one: bài toán có *nhiều input* nhưng chỉ có *1 output*, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, output là hành động trong video

Many to many: bài toán có *nhiều input* và *nhiều output*, ví dụ bài toán dịch từ tiếng Anh sang tiếng Việt, input là 1 câu gồm nhiều chữ: "I love Vietnam" và output cũng là 1 câu gồm nhiều chữ "Tôi yêu Việt Nam".



Hình 25: Các dạng bài toán RNN

4.4 Ưu và nhược điểm của một kiến trúc RNN thông thường

Ưu điểm	Hạn chế
<ul style="list-style-type: none">• Khả năng xử lý đầu vào với bất kì độ dài nào• Kích cỡ mô hình không tăng theo kích cỡ đầu vào• Quá trình tính toán sử dụng các thông tin cũ• Trọng số được chia sẻ trong suốt thời gian	<ul style="list-style-type: none">• Tính toán chậm• Khó để truy cập các thông tin từ một khoảng thời gian dài trước đây• Không thể xem xét bất kì đầu vào sau này nào cho trạng thái hiện tại

Bảng 1.: Ưu và nhược điểm của mạng RNN

4.5 Vấn đề Vanishing gradient trong RNN (Đạo hàm bị triệt tiêu)

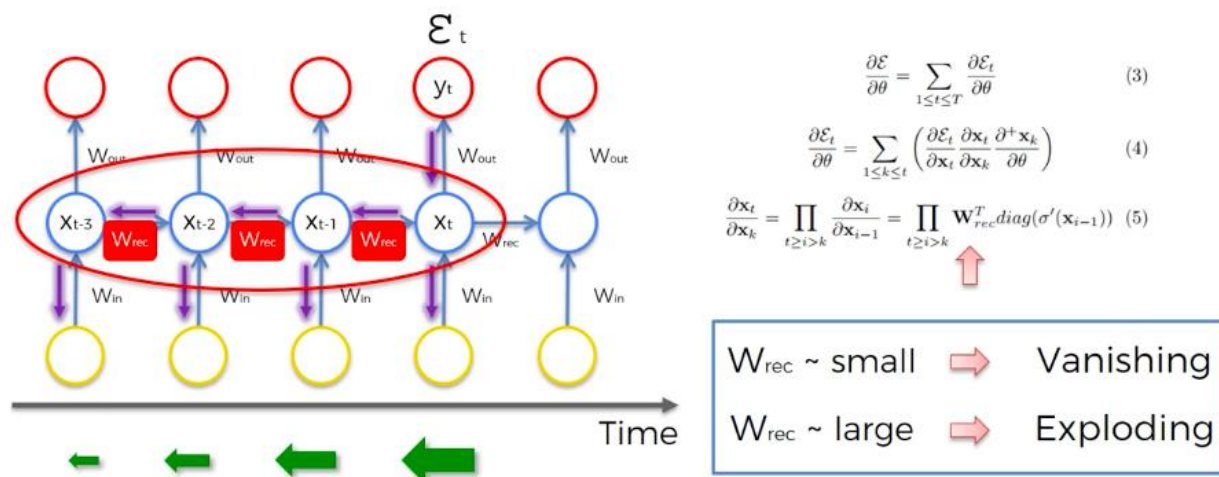
Vì hàm kích hoạt của chúng ta (*tanh* hoặc *sigmoid*) sẽ đưa ra đầu ra trong khoảng $[-1, 1]$ (với *sigmoid* là $[0, 1]$), đạo hàm sẽ là đóng trong khoảng $[0, 1]$ (với *sigmoid* là $[0, 0,25]$).

Ở trên, ta đã sử dụng quy tắc chuỗi để tính đạo hàm. Có một vấn đề ở đây, hàm *tanh* và hàm *sigmoid* **không có đạo hàm ở cả hai đầu**. Và khi đạo hàm bằng 0, nút mạng tương ứng y sẽ bão hòa. Lúc này, các nút chuyển tiếp cũng sẽ bị bão hòa. Vì vậy, đối với các giá trị nhỏ trong ma trận, khi chúng ta thực hiện phép nhân ma trận, đạo hàm tương ứng sẽ xảy ra *biến đổi gradient*, tức là đạo hàm bị hủy chỉ sau một vài bước nhân. Vì vậy, các bước từ xa sẽ không còn hoạt động với nút hiện tại, điều này sẽ ngăn RNN học các phụ thuộc từ xa. Sự cố này không chỉ xảy ra với RNN mà ngay cả các mạng nơ-ron nhiều lớp truyền thống cũng có hiện tượng này.

Với cách nhìn trên, ngoài **Vanishing gradient**, chúng ta còn gặp **Exploding Gradient**. Tùy thuộc vào chức năng kích hoạt và cài đặt mạng, sự cố này xảy ra khi các giá trị ma trận lớn (lớn hơn 1). Tuy nhiên, người ta thường nói về vấn đề biến mất hơn là phát nổ, vì 2 lý do sau đây.

Đầu tiên, sự bùng nổ của đạo hàm có thể theo dõi được bởi vì khi đạo hàm bùng nổ, chúng ta nhận được một giá trị không phải số của NaN khiến chương trình của chúng ta dừng lại.

Thứ hai, sự bùng nổ đạo hàm có thể tránh được khi chúng ta đặt giá trị ngưỡng cao hơn (tham khảo kỹ thuật Gradient Clipping). Việc theo dõi sự mất mát của các công cụ phát sinh và tìm cách xử lý vẫn còn rất khó khăn.



Hình 26: Vanishing Gradient

Để xử lý Vanishing Gradient, có hai cách phổ biến:

Cách thứ nhất, thay vì sử dụng hàm *tanh* và kích hoạt *sigmoid*, chúng ta thay thế nó bằng *ReLU* (hoặc các biến thể như *Leaky ReLU*). Đạo hàm của *ReLU* là 0 hoặc 1, do đó vấn đề mất đạo hàm có thể được kiểm soát phần nào.

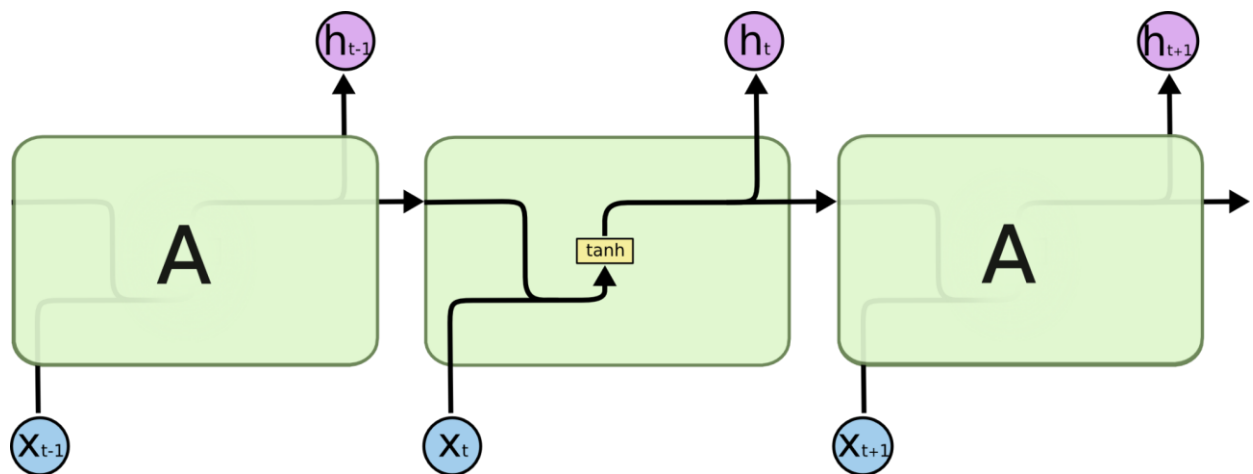
Thứ hai, chúng ta thấy rằng RNN thuần túy không được thiết kế để lọc ra những thông tin không cần thiết. Chúng ta *cần thiết kế một kiến trúc có thể nhớ lâu hơn*, đó là **LSTM** và **GRU**.

5. Bộ nhớ dài-ngắn hạn (LSTM)

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là **LSTM** - là một dạng đặc biệt của **RNN**, nó có khả năng học được các phụ thuộc xa. **LSTM** được giới thiệu bởi *Hochreiter & Schmidhuber (1997)*, và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

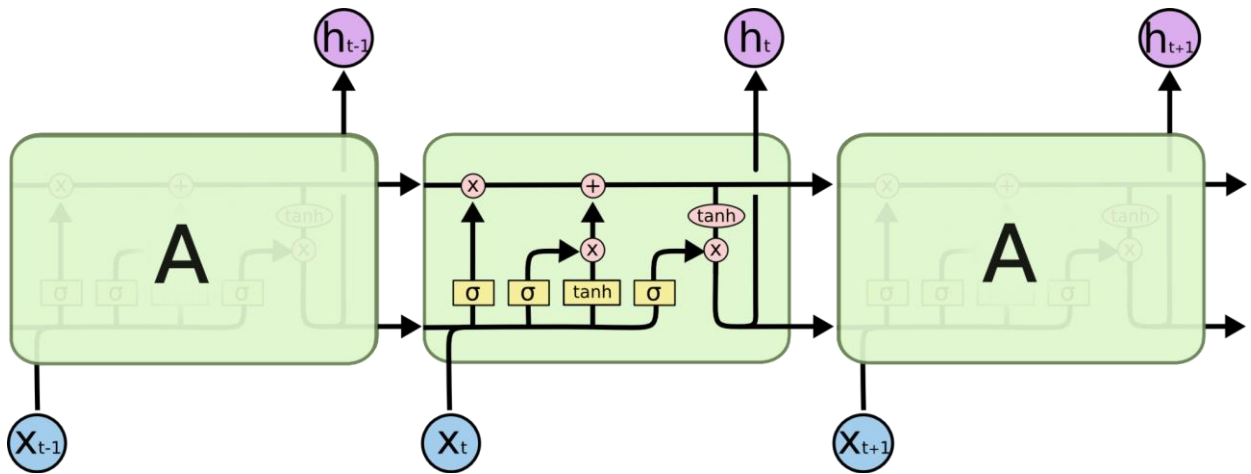
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (*long-term dependency*). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó *đã có thể ghi nhớ được mà không cần bất kì can thiệp nào*.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-đun này có cấu trúc rất đơn giản, thường là một tầng *tanh*.



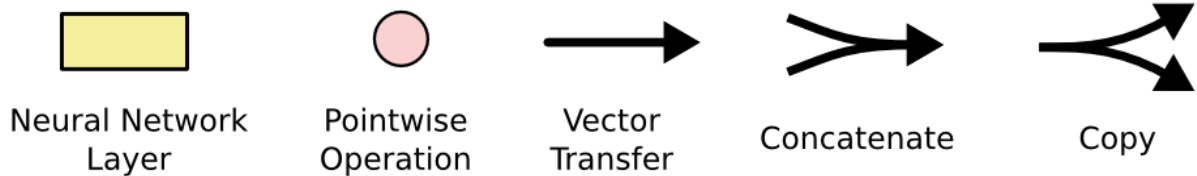
Hình 27: Cấu trúc mạng RNN truyền thống

LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng **RNN chuẩn**. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình 28: Cấu trúc mạng LSTM được cải tiến từ RNN

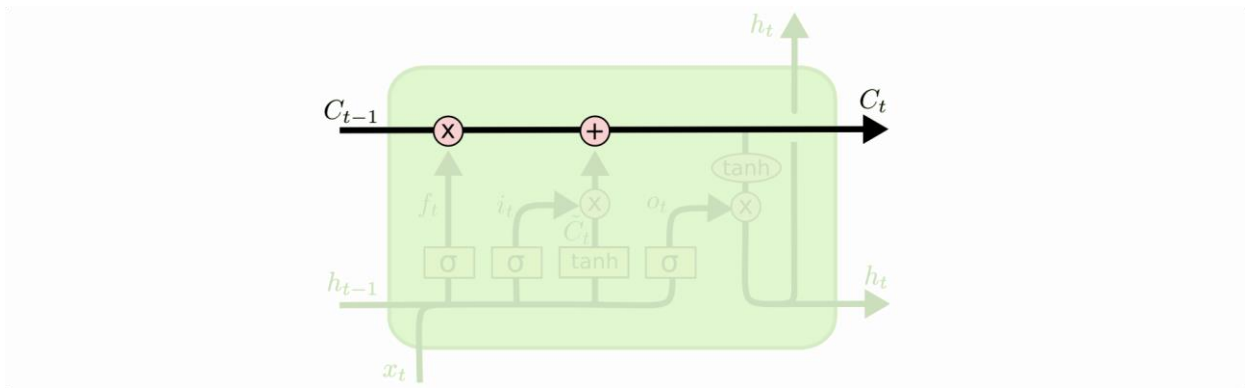
Các kí hiệu có ý nghĩa như sau:



Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các tầng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

5.1 Ý tưởng cốt lõi của LSTM

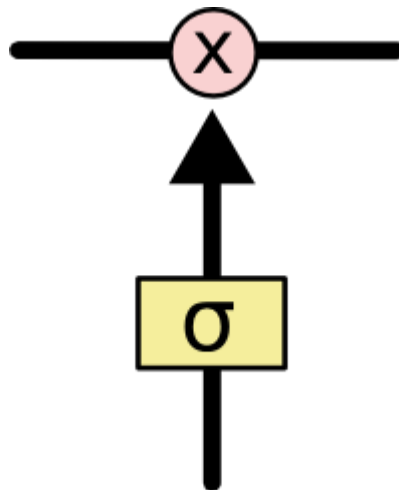
Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.



Hình 29: Mô tả dạng băng chuyền thể hiện trạng thái tế bào trong LSTM

Trạng thái tế bào là một dạng giống như băng truyền. Nó *chạy xuyên suốt tất cả các mắt xích* (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.

LSTM có khả năng *bỏ đi* hoặc *thêm vào* các thông tin cần thiết cho trạng thái tế bào, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là *cổng* (gate). Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng *sigmoid* và *một phép nhân*.

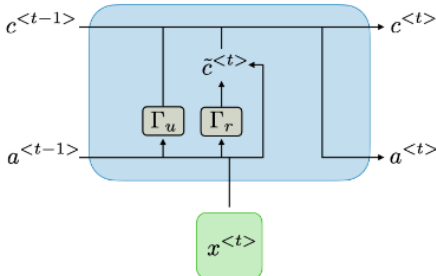
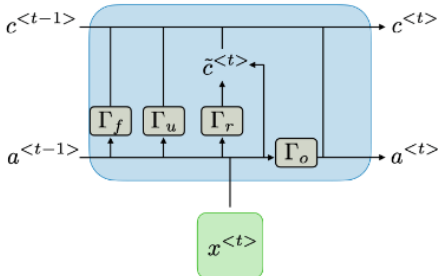


Hình 30: Cổng sàng lọc thông tin LSTM

Tầng sigmoid sẽ cho đầu ra là một số trong khoản $[0, 1]$, mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 0 thì có nghĩa là không cho thông tin nào qua cả, còn

khi là 1 thì có nghĩa là cho tất cả các thông tin đi qua nó. Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

5.2 Cấu trúc chi tiết của GRU/LSTM

Đặc tính	Gated Recurrent Unit (GRU)	Bộ nhớ dài-ngắn hạn (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Các phụ thuộc		

Chú ý: kí hiệu \star chỉ phép nhân từng phần tử với nhau giữa hai vectors.

Hình 31: Cấu trúc chi tiết của GRU và LSTM

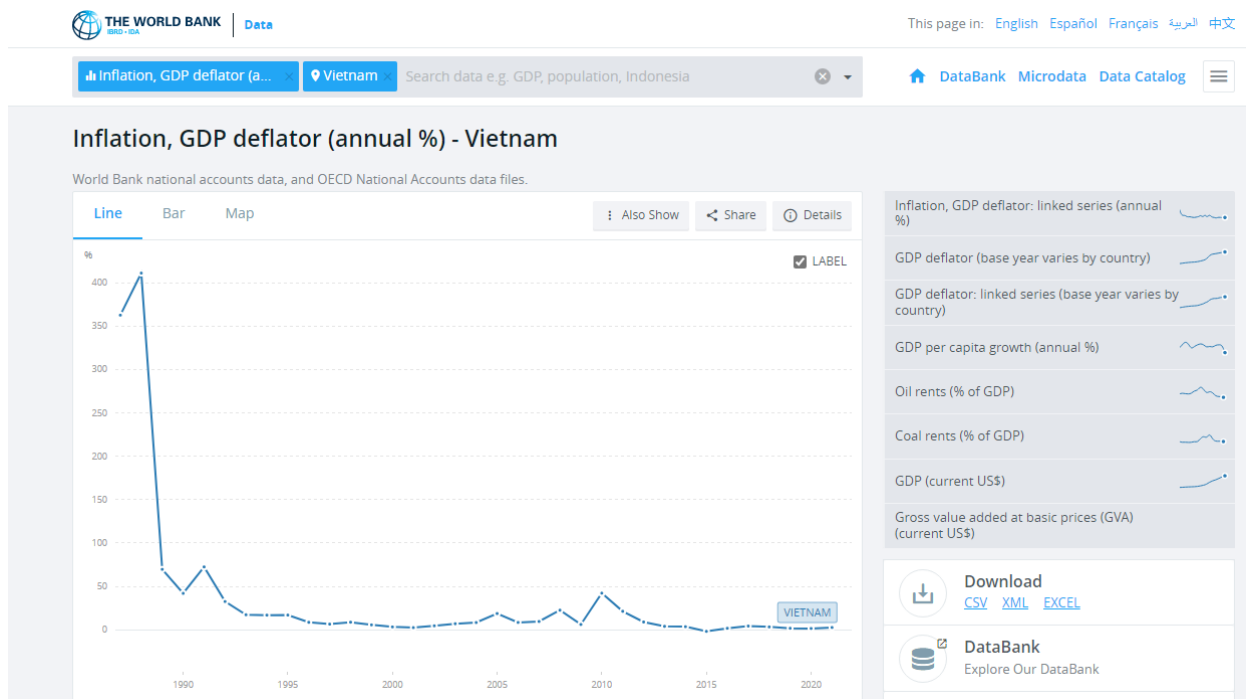
Các ứng dụng của LSTM bao gồm: Quản lý robot, dự đoán chuỗi thời gian, nhận dạng tiếng nói, học giai điệu (nhịp điệu), sáng tác nhạc, học ngữ pháp, phiên dịch,...vân vân.

Vậy là chúng ta đã tìm hiểu qua các kiến thức về mạng, cấu trúc, cách hoạt động và các nguyên lý máy học căn bản. Tiếp theo chúng ta sẽ đi đến phần thực nghiệm để áp dụng những kiến thức đã biết vào xây dựng mô hình dự đoán lạm phát của Việt Nam để giải quyết yêu cầu của đề tài.

Chương 3: THỰC NGHIỆM

1. Thu thập dữ liệu

Để có được dữ liệu lạm phát ở Việt Nam ta sẽ lấy từ nguồn ***data.worldbank.org*** đây là một nơi chuyên tổng hợp các số liệu kinh tế của các quốc gia khác nhau.



Hình 32: Số liệu từ *data.worldbank.org*

Tuy nhiên, do tỷ lệ lạm phát được tính theo đơn vị % cho nên sẽ rất khó cho mô hình tìm ra quy luật để thực hiện việc dự đoán. Theo như tìm hiểu thì ta biết được dữ liệu lạm phát được tính dựa trên **chỉ số CPI (Chỉ số giá tiêu dùng)** và tính theo công thức:

Tỷ lệ lạm phát kỳ hiện tại = (Giá trị CPI cuối cùng / Giá trị CPI ban đầu) x 100

Ví dụ: Tỷ lệ lạm phát năm 2020 so với năm 2019 = (Giá trị chỉ số CPI năm 2020 / Giá trị CPI năm 2019) x 100

Giả sử chỉ số CPI năm 2019 và 2020 lần lượt là 98 và 105. Như vậy, tỷ lệ lạm phát năm 2020 so với năm 2019 sẽ là: $(105 / 98) \times 100 = 107,14\%$



Hình 33: Dữ liệu CPI Việt Nam

Như vậy, ta chỉ cần dự đoán số liệu CPI rồi dựa theo công thức trên sẽ suy ra được chỉ số lạm phát.

Chỉ số CPI: Chỉ số giá tiêu dùng (Consumer Price Index) là chỉ số tính theo phần trăm để phản ánh mức thay đổi tương đối của giá hàng tiêu dùng theo thời gian. Sở dĩ chỉ là thay đổi tương đối vì chỉ số này chỉ dựa vào một giỏ hàng hóa đại diện cho toàn bộ hàng tiêu dùng.

Để lấy dữ liệu từ trang này, ta nhấn vào download file dữ liệu tổng hợp dạng **CSV** phía bên góc dưới bên phải sẽ lấy được dữ liệu về máy. Vì là file dữ liệu tổng hợp nên nó sẽ bao gồm **dữ liệu CPI của hơn 200 quốc gia khác nhau** (giai đoạn 1960 - 2021). Ta sẽ đổi tên của file này lại thành **world_CPI.csv**.

	A	B	C	D	E	F	G	H	I
1	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
2	Aruba	ABW	Consumer Price Index	FP.CPI.TOTL					
3	Africa Eastern & S.	AFE	Consumer Price Index	FP.CPI.TOTL					
4	Afghanistan	AFG	Consumer Price Index	FP.CPI.TOTL					
5	Africa Western & C.	AFW	Consumer Price Index	FP.CPI.TOTL					
6	Angola	AGO	Consumer Price Index	FP.CPI.TOTL					
7	Albania	ALB	Consumer Price Index	FP.CPI.TOTL					
8	Andorra	AND	Consumer Price Index	FP.CPI.TOTL					
9	Arab World	ARB	Consumer Price Index	FP.CPI.TOTL					
10	United Arab Emirates	ARE	Consumer Price Index	FP.CPI.TOTL					
11	Argentina	ARG	Consumer Price Index	FP.CPI.TOTL					
12	Armenia	ARM	Consumer Price Index	FP.CPI.TOTL					
13	American Samoa	ASM	Consumer Price Index	FP.CPI.TOTL					
14	Antigua and Barbuda	ATG	Consumer Price Index	FP.CPI.TOTL					
15	Australia	AUS	Consumer Price Index	FP.CPI.TOTL	7.960458	8.14256	8.116545	8.168574	8.402706
16	Austria	AUT	Consumer Price Index	FP.CPI.TOTL	17.82417	18.45554	19.26423	19.78605	20.55149

Hình 34: Mô tả dữ liệu trong file *world_CPI.csv*

Vì chúng ta chỉ quan tâm đến dữ liệu CPI của Việt Nam nên ta sẽ tiến hành đưa nó vào ngôn ngữ Python để xử lý và lọc ra giá trị cần thiết.

2. Xử lý dữ liệu

Ta tiến hành import thư viện và dùng hàm `read_csv` của Pandas để lấy dữ liệu từ file csv.

```
# Import thư viện
import pandas as pd
import numpy as np
import math
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

# Lấy dữ liệu dạng Dataframe từ file dữ liệu
DATASET_inflation = pd.read_csv('world_CPI.csv')
```

Từ file Excel ta thấy số liệu Việt Nam đứng ở **dòng thứ 257** ta sẽ lấy số liệu dòng này ra. Tuy nhiên, số liệu của Việt Nam chỉ được thống kê từ năm 1995, những năm trước đó không có nên ta sẽ loại bỏ những hàng rỗng đi bằng hàm **dropna()**.

```
# Lấy số liệu của Việt Nam
VN_inflation = DATASET_inflation.T[257] # Vietnam
# Loại bỏ các năm có số liệu bị rỗng
VN_inflation = VN_inflation[4:].dropna()
print(VN_inflation)
```

1995	40.16572
1996	42.445125
1997	43.807412
1998	46.990546
1999	48.025105

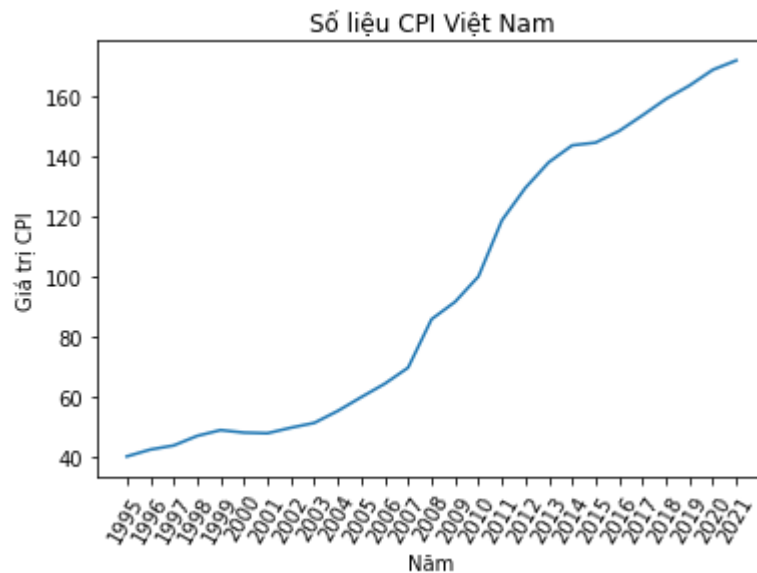
Như vậy, ta đã lọc ra được dữ liệu lạm phát của Việt Nam từ lúc được ghi nhận đến nay, tức giai đoạn 1995 - 2021. Ta sẽ tiến hành trực quan hoá dữ liệu này để có cái nhìn toàn diện nhất.

3. Trực quan hoá dữ liệu

Ta tiến hành vẽ biểu đồ đường với thư viện **matplotlib.pyplot** để mô tả.

```
# Vẽ biểu đồ trực quan hoá dữ liệu
import matplotlib.pyplot as plt
plt.plot(VN_inflation)
plt.title('Số liệu CPI Việt Nam')
plt.xlabel("Năm")
plt.ylabel("Giá trị CPI")
plt.xticks(rotation=60)
plt.show()
```

Kết quả trực quan:



Ta thấy dữ liệu giá tiêu dùng tăng dần theo từng năm thể hiện giá hàng hoá tăng dần, đồng nghĩa với việc có sự giảm giá trị đồng tiền xảy ra (lạm phát). Tiếp theo, ta sẽ tiến hành xây dựng mô hình dự đoán.

4. Xây dựng mô hình dự đoán

Trước khi bắt đầu xây dựng mô hình, ta tiến hành chuẩn hoá dữ liệu. Vì dữ liệu là những con số lớn, nếu ta để trực tiếp những giá trị đó vào thì vẫn có thể dự đoán được, tuy nhiên sẽ làm tốn tài nguyên bộ nhớ cũng như làm chậm trong việc tính toán mô hình, để tối ưu hơn ta sẽ dùng hàm **MinMaxScaler*** để Scale dữ liệu về còn trong khoảng [0,1].

```
# Chuẩn hoá dữ liệu, scale dữ liệu về khoảng [0, 1]
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
VN_inflation = sc.fit_transform(VN_inflation.to_numpy()
                                .reshape(-1,1).tolist())
VN_inflation = VN_inflation.flatten()
```

Hàm MinMaxScaler là gì?

Ta có thể hiểu đơn giản, một số bình thường trong tập dữ liệu sẽ nằm **trong khoảng** $[-\infty; +\infty]$, khi ta thực hiện Scale số đó về [0;1] thì hàm MinMaxScaler sẽ tính toán theo công thức dưới để thu nhỏ số này về trong [0;1] mà thôi thay vì $[-\infty; +\infty]$. Nếu số sau khi scale **bằng 1** tương ứng với số đó là **số lớn nhất trong tập** và nếu **bằng 0** thì số đó là **số bé nhất trong tập**.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Hình 35: Công thức hàm Minmaxscaler

Ý tưởng dự đoán

Để tiến hành xây dựng mô hình huấn luyện ta cần biết được *đầu vào* và *đầu ra mong muốn* là gì. Ta gọi **X là giá trị đầu vào** và **Y là giá trị đầu ra mong muốn**, ta cần xác định hai giá trị này từ dữ liệu trên. Do dữ liệu dạng theo thời gian liên tục nên ta sẽ tách ra bằng

cách lấy giá trị của 5 năm trước đó để dự đoán năm tiếp theo và tiếp tục tăng lên mỗi một đơn vị.

D	E	F	G	H	I	J	K
Train 1					Test 1		
40.1657	42.4451	43.8074	46.9905	48.9252	48.0884	47.8809	49.7151
	Train 2					Test 2	

Hình 36: Mô tả cách phân tập X và Y từ dữ liệu CPI

Hiện thực hoá trên Python:

```
# Phân lớp dữ liệu ra làm 2 phần X, Y
X,Y=[],[]
range_X = 5
for i in range(range_X,len(VN_inflation)):
    X.append(VN_inflation[i-range_X:i].tolist())
    Y.append(VN_inflation[i])

Y = np.array(Y).reshape(-1,1).tolist()
```

Sau khi đã tách được tập **X**, **Y** ta sẽ cắt cặp dữ liệu ra làm 2 phần theo tỷ lệ **80/20**, với **80%** để huấn luyện mô hình và **20%** còn lại là dữ liệu mô hình chưa từng nhìn thấy trước đó, mục đích là để test xem mô hình có hoạt động hiệu quả hay là không.

```
from sklearn.model_selection import train_test_split
# Cắt tập dữ liệu X, Y theo tỷ lệ 80/20 (80% training, 20% test)
x_training, x_test, y_training, y_test = train_test_split(X,Y,test_size=0.20,
                                                         shuffle=True,
                                                         random_state=0)
```

Vậy là ta đã có bốn biến bao gồm **x_training**, **y_training** phục vụ cho việc huấn luyện và **x_test**, **y_test** phục vụ cho việc kiểm thử mô hình.

Xây dựng mô hình mạng

Ta sẽ lựa chọn cấu trúc mạng LSTM để xây dựng mô hình bởi vì cấu trúc này có nhiều ưu điểm hỗ trợ cho việc xử lý dữ liệu dạng time-series. Mô hình ta xây dựng sẽ bao gồm 4 lớp:

Lớp thứ nhất: LSTM, lớp đầu vào, bao gồm **128 units** với, hàm kích hoạt là **ReLU**.

Lớp thứ hai: LSTM, lớp ẩn gồm **64 units**.

Lớp thứ ba: Dropout, có **giá trị 0.5** để loại bỏ một vài unit trong mạng nhằm **tránh over-fitting**.

Lớp thứ tư: Dense, lớp đầu ra, gồm **1 units** nhận giá trị đầu vào của tất cả các nút trước đó.

Hiện thực hoá trên Python:

```
from keras.models import Sequential
from keras.layers import LSTM,Dropout,Dense
lstm_model=Sequential()
lstm_model.add(LSTM(units=128,return_sequences=True,
                    activation='relu',
                    input_shape=(range_X,1)))
lstm_model.add(LSTM(units=64))
lstm_model.add(Dropout(0.5))
lstm_model.add(Dense(1))

lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_training,y_training,epochs=150,batch_size=5,verbose=2)
```

Tiếp theo, chúng ta sẽ compile mô hình với hàm tối ưu hoá là **adam** và hàm mất mát (*loss function*) là **MSE**.

Cuối cùng ta sẽ huấn luyện mô hình bằng hàm **.fit()** với lượng *epochs = 150*, *batch_size = 5*, *verbose = 2*.

Quá trình chạy training:

```
lstm_model.compile(loss= mean_squared_error ,optimizer=adam)
lstm_model.fit(x_training,y_training,epochs=150,batch_size=4)
```

✕ Epoch 1/150
4/4 - 3s - loss: 0.3290 - 3s/epoch - 869ms/step
Epoch 2/150
4/4 - 0s - loss: 0.2513 - 109ms/epoch - 27ms/step
Epoch 3/150
4/4 - 0s - loss: 0.1878 - 118ms/epoch - 30ms/step
Epoch 4/150
4/4 - 0s - loss: 0.1213 - 70ms/epoch - 17ms/step
Epoch 5/150
4/4 - 0s - loss: 0.0713 - 91ms/epoch - 23ms/step
Epoch 6/150
4/4 - 0s - loss: 0.0268 - 79ms/epoch - 20ms/step

Vậy là chúng ta đã hoàn tất việc xây dựng và huấn luyện mô hình dự đoán, tiếp theo chúng ta sẽ tiến hành đưa dữ liệu test, dữ liệu mà mô hình chưa nhìn thấy trước đó vào để đánh giá mức độ chính xác của mô hình này.

5. Tiến hành dự đoán, đánh giá.

Ta tiến hành đưa giá trị **x_test** vào hàm **.predict()** để tiến hành dự đoán mô hình sau đó kết quả dự đoán sẽ được lưu vào biến **pred_y_test**, lúc này để dễ hình dung ta sẽ vẽ biểu đồ đường ra để dễ so sánh **giá trị dự đoán** và **giá trị thực tế** có sự tương đồng hay là không.

Thực nghiệm trên Python:

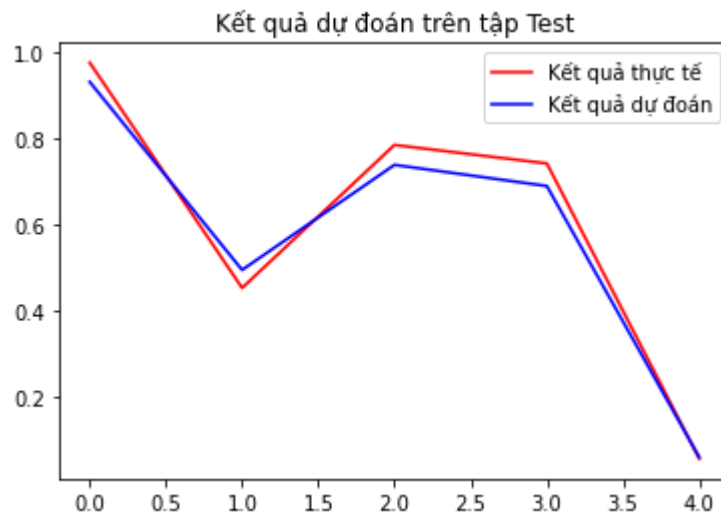
```

pred_y_test = lstm_model.predict(x_test)

x_ax=range(len(x_test))
plt.plot(x_ax, y_test, lw=1.5, color="red",
         label="Kết quả thực tế")
plt.plot(x_ax, pred_y_test, lw=1.5, color="blue",
         label="Kết quả dự đoán")
plt.title("Kết quả dự đoán trên tập Test")
plt.legend()
plt.show()

```

Kết quả biểu đồ:



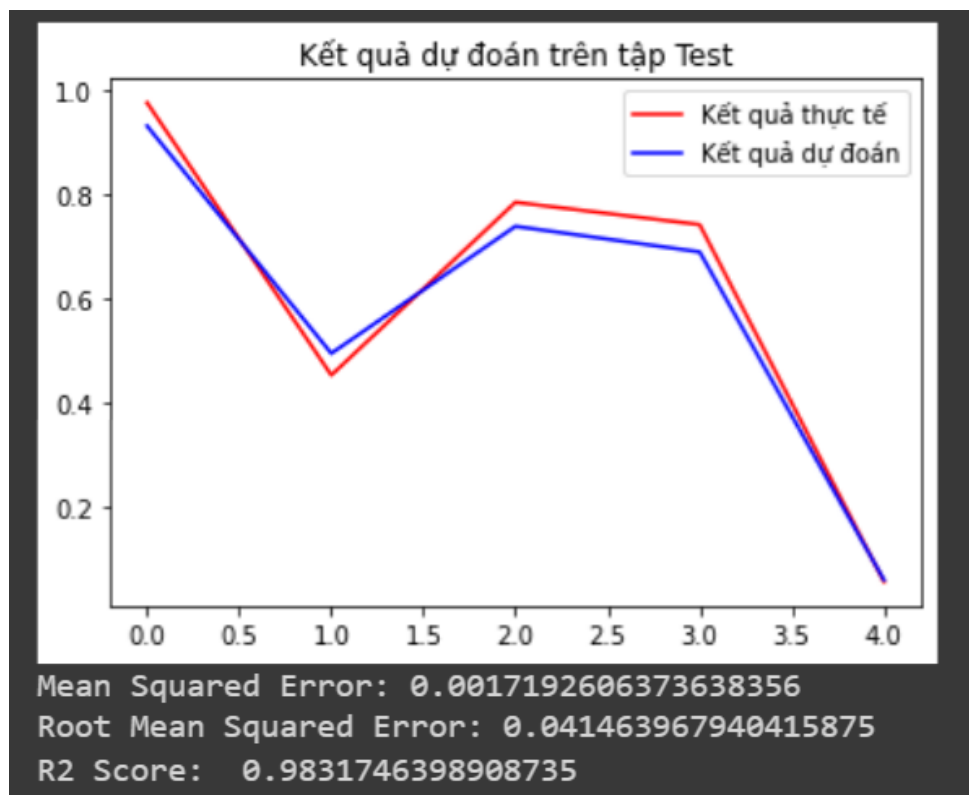
Từ kết quả trên ta thấy được có sự *trùng đồng đáng kể* giữa kết quả thực tế và dự đoán khi nhìn bằng mắt thường, tuy nhiên để chính xác hơn ta sẽ dùng các *hàm tính sai số* để xác định mức độ tương quan của mô hình dự đoán này. Cụ thể ta sẽ dùng hai hàm tính sai số **MSE**, **RMSE** và hàm **R² Score** để ước lượng độ chính xác của mô hình.

Hiện thực hoá trên Python:

```
# Hiển thị sai số.
pred_y_test = pred_y_test.tolist()

mse = mean_squared_error(y_test, pred_y_test)
print("Mean Squared Error:",mse)
rmse = math.sqrt(mse)
print("Root Mean Squared Error:", rmse)
print("R2 Score: ",r2_score(y_test, pred_y_test))
```

Kết quả dự đoán:



Đánh giá kết quả thử nghiệm (Test): Ta thấy kết quả khá ấn tượng. Giá trị **MSE** và **R-MSE** gần bằng 0 và điểm số **R² Score** tương đương **98%** đồng nghĩa kết quả dự đoán giống đến **98%** kết quả kỳ vọng. Đây là một con số khả quan, nó cho thấy mô hình này khả thi khi dự đoán trong thực tế.

Vậy là chúng ta đã hoàn thành việc xây dựng mô hình dự đoán chỉ số CPI, từ các chỉ số dự đoán được này chúng ta có thể suy ra và tính được mức độ lạm phát của Việt Nam những năm tới.

6. Mã nguồn hoàn chỉnh:

Source Code:

https://github.com/phatjkk/world_inflation_data/blob/main/DACS.ipynb

QR Code:



Chương 4: KẾT LUẬN

Trong thời đại ngày nay, vấn đề lạm phát và ảnh hưởng của lạm phát đối với tăng trưởng kinh tế là một đề tài vô cùng nóng, đặc biệt là trong bối cảnh Việt Nam đang trong quá trình hội nhập và phát triển kinh tế. Mô hình dự báo lạm phát do chúng em thực hiện có thể giúp ích cho việc dự đoán tỷ lệ lạm phát ở Việt Nam trong những năm tiếp theo, từ đó có thể đưa ra những chính sách, kế hoạch phù hợp nhằm ổn định theo hướng có lợi cho nền kinh tế của đất nước.

1. Ưu điểm

Mô hình xây dựng đơn giản, dễ sử dụng

Tỷ lệ dự đoán có độ chính xác cao (98%)

Đề tài có sức ảnh hưởng lớn, vô cùng quan trọng với sự phát triển của một quốc gia nên có tiềm năng phát triển mạnh mẽ.

2. Nhược điểm

Chưa phân tích chuyên sâu được vấn đề do khả năng hiểu biết còn hạn chế.

Dữ liệu chưa phong phú, do đây là vấn đề tương đối bảo mật của nước ta nên khó thu thập hoàn chỉnh.

3. Hướng phát triển

Nếu được đầu tư thêm thời gian và công sức, mô hình này hoàn toàn khả thi và có thể phát triển hoàn thiện trong tương lai với tỷ lệ dự đoán chính xác cao hơn, dự đoán được nhiều trường hợp hơn...

Xây dựng thêm các mô hình dự đoán tương tự với bộ dữ liệu chi tiết, để tìm kiếm như dự đoán giá vàng, chứng khoán, dự báo thời tiết...

TÀI LIỆU THAM KHẢO

Blankespoor, B., Baffes, J., Erman, A., & Erman, A. (2022, September 7). World Bank Open Data. Data. Retrieved September 12, 2022, from <https://data.worldbank.org/>

Công Thức Tính Lạm Phát là gì? Những kiến thức Cơ Bản Về Lạm Phát. Entrade X by DNSE. (n.d.). Retrieved September 12, 2022, from <https://www.dnse.com.vn/hoc/cong-thuc-tinh-lam-phat-la-gi>

Dominhai. (n.d.). [RNN] LSTM là gì? Hai's Blog. Retrieved September 12, 2022, from <https://dominhhai.github.io/vi/2017/10/what-is-lstm/#3-2-b%C3%AAn-trong-lstm>

Huyen, N. T. (2022, September 12). Recurrent neural network: TỪ RNN đến LSTM. Viblo. Retrieved September 12, 2022, from <https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2>

Mạng Neural Hồi quy cheatsheet star. CS 230 - Mạng nơ-ron hồi quy cheatsheet. (n.d.). Retrieved September 12, 2022, from <https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-recurrent-neural-networks>

nttuan8, |. (2019, November 7). Bài 13: Recurrent neural network. Deep Learning cơ bản. Retrieved September 12, 2022, from <https://nttuan8.com/bai-13-recurrent-neural-network/>

Wikimedia Foundation. (2022, September 2). Main page. Wikipedia. Retrieved September 12, 2022, from <https://www.wikipedia.org/>