BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC VĂN HIẾN

KHOA KỸ THUẬT – CÔNG NGHỆ

HỌC MÁY

ĐỀ TÀI 33: HỌC CÁCH XÂY DỰNG MẠNG NƠ-RON TỪ ĐẦU BẰNG NUMPY

GVGD: HỒ NHẬT MINH

SVTH: Trần Trung Lâm - 221A020047

Nguyễn Đức Hoàng - 221A020016

Phan Minh Tuấn- 221A020044

Võ Duy Thanh - 221A020035

Trần Trí Thức - 221A020011

Nguyễn Minh Hùng – 221A020005

TP. HỒ CHÍ MINH - 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC VĂN HIẾN

KHOA KỸ THUẬT – CÔNG NGHỆ

HỌC MÁY

ĐỀ TÀI 33: HỌC CÁCH XÂY DỰNG MẠNG NƠ-RON TỪ ĐẦU BẰNG NUMPY

STT	HỌ VÀ TÊN SINH VIÊN	MSSV	CHỮ KÝ
1	Trần Trung Lâm	221A020047	
2	Nguyễn Đức Hoàng	221A020016	
3	Phan Minh Tuấn	221A020044	
4	Võ Duy Thanh	221A020035	
5	Trần Trí Thức	221A020011	
6	Nguyễn Minh Hùng	221A020005	

TP. HỒ CHÍ MINH - 2024

MỤC LỤC

LỜI CẢM ƠN

Lời nói đầu tiên nhóm em xin phép được gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Hồ Nhật Minh vì những bài giảng vô cùng bổ ích và tâm huyết mà thầy đã truyền đạt trong suốt thời gian qua. Qua mỗi buổi học, chúng em cảm nhận được sự nhiệt tình và lòng đam mê của thầy trong việc giảng dạy, cũng như sự kiên nhẫn và tận tâm trong việc giải đáp những thắc mắc của chúng em. Những phương pháp giảng dạy sinh động và gần gũi của thầy đã biến những bài học khô khan trở nên thú vị và dễ tiếp thu hơn.

Chúng em hiểu rằng, để có được những bài giảng chất lượng và bổ ích như vậy, thầy đã phải đầu tư rất nhiều công sức và thời gian trong việc nghiên cứu và chuẩn bị. Chúng em vô cùng biết ơn những nỗ lực và sự cống hiến hết mình của thầy. Nhờ có sự hướng dẫn của thầy mà chúng em đã vượt qua được nhiều khó khăn và tự tin hơn trong quá trình học tập. Nhóm em xin hứa sẽ tiếp tục cố gắng học tập, rèn luyện và áp dụng những kiến thức đã học được từ thầy vào thực tế một cách hiệu quả nhất. Chúng em cũng mong rằng trong tương lai sẽ có thêm nhiều cơ hội được học hỏi và nhận được sự chỉ dẫn từ thầy.

Một lần nữa, nhóm em xin chân thành cảm ơn thầy và chúc thầy luôn mạnh khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy. Mong rằng thầy sẽ tiếp tục truyền lửa và cảm hứng học tập cho nhiều thế hệ sinh viên sau.

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

TP. Hồ Chí Minh, ngày…...tháng ……năm…

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

TỔNG QUAN

Mạng nơ-ron là một tập hợp các thuật toán nhằm nhận diện các mô hình tiềm ẩn trong dữ liệu. Chúng là các tập hợp con của học máy và là cốt lõi của các thuật toán học sâu. Mạng nơ-ron được lấy cảm hứng từ cách hoạt động của não bộ con người. Mạng nơ-ron được ứng dụng trong nhiều lĩnh vực thực tế như xe tự lái, nhận diện giọng nói, chẩn đoán y khoa, và nhiều ứng dụng khác. Trong đề tài này, chúng ta sẽ xây dựng một mạng nơ-ron từ đầu chỉ sử dụng NumPy mà không sử dụng bất kỳ khung học sâu nào như TensorFlow hoặc PyTorch. Mạng nơ-ron được xây dựng sẽ dự đoán giá nhà ở Pune, Ấn Độ. Đề tài này cũng sẽ cung cấp cho bạn một cái nhìn sâu sắc về cách hoạt động của mạng nơ-ron.

A. Mục tiêu

Hiểu cách hoạt động của mạng nơ-ron

Xây dựng một mạng nơ-ron từ đầu sử dụng NumPy

B. Mô tả dữ liệu

Tập dữ liệu chứa thông tin về giá nhà dựa trên các thuộc tính khác nhau. Tập dữ liệu này bao gồm giá nhà từ Pune, Maharashtra (Ấn Độ).

C. Công nghệ sử dụng

➔ Ngôn ngữ: Python

➔ Thư viện: pandas, sci-kit learn, numpy

D. Cách tiếp cận

Tiền xử lý dữ liệu

Loại bỏ các giá trị null

Chuẩn hóa các đặc trưng số học

Mã hóa One-Hot cho dữ liệu phân loại

Huấn luyện mô hình

Huấn luyện mô hình mạng nơ-ron

Đánh giá mô hình

Đánh giá mô hình trên dữ liệu kiểm tra

E. Tổng quan mã code theo mô-đun:

Input

└── Real_Estate_Data.xlsx

MLPipeline

├── Neural_Network.py

└── Preprocessing.py

Notebook

└── nn_demo. ipynb

Resources

├── graphics

├── 01_real_world. ipynb

├── 02_history. ipynb

└── 03_neural_net. ipynb

Engine.py

Readme.md

Requirements.txt

F. Khi bạn giải nén tệp neural_network.zip, bạn sẽ tìm thấy các thư mục sau trong đó:

1. Input
2. ML_Pipeline
3. Notebook
4. Resources
5. Engine.py
6. Readme.md
7. requirements.txt

1. Thư mục Input chứa dữ liệu mà chúng ta sẽ sử dụng để phân tích. Trong trường hợp này, nó chứa tập dữ liệu dự đoán giá nhà.

2. Thư mục Notebook chứa tệp jupyter notebook của dự án.

3. ML_pipeline là một thư mục chứa tất cả các hàm được phân loại vào các tệp Python khác nhau với tên gọi tương ứng. Những hàm Python này sau đó được gọi trong tệp Engine.py.

4. Thư mục Resources chứa các tài liệu sử dụng trong dự án.

5. Tệp requirements.txt có tất cả các thư viện cần thiết với các phiên bản tương ứng. Vui lòng cài đặt bằng lệnh: `pip install -r requirements.txt`.

6. Tất cả hướng dẫn để chạy mã đều có trong tệp Readme.md.

KẾT LUẬN

1. Ứng dụng thực tế của học sâu (deep learning)

1.1 Phát hiện gian lận

Thuật toán deep learning có thể xác định các vấn đề bảo mật để giúp bảo vệ chống gian lận. Ví dụ, thuật toán deep learning có thể phát hiện các nỗ lực đăng nhập đáng ngờ vào tài khoản của bạn và thông báo cho bạn, cũng như cảnh báo nếu mật khẩu bạn chọn không đủ mạnh.

1.2 Dịch vụ khách hàng

Xuất hiện nhiều dịch vụ hỗ trợ khách hàng trực tuyến và tương tác với chatbot để trả lời các câu hỏi hoặc sử dụng trợ lý ảo trên điện thoại thông minh. Deep learning cho phép các hệ thống này học hỏi theo thời gian để phản hồi tốt hơn.

1.3 Dịch vụ tài chính

Nhiều dịch vụ tài chính có thể dựa vào sự hỗ trợ từ deep learning. Phân tích dự đoán giúp hỗ trợ danh mục đầu tư và giao dịch tài sản trên thị trường chứng khoán, cũng như cho phép các ngân hàng giảm thiểu rủi ro liên quan đến phê duyệt khoản vay.

1.4 Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên là một phần quan trọng của các ứng dụng deep learning dựa vào việc hiểu văn bản và lời nói. Chatbot dịch vụ khách hàng, bộ dịch ngôn ngữ và phân tích cảm xúc là các ví dụ về các ứng dụng hưởng lợi từ xử lý ngôn ngữ tự nhiên.

1.5 Nhận diện khuôn mặt

Một lĩnh vực của deep learning được gọi là thị giác máy tính cho phép các thuật toán deep learning nhận diện các đặc điểm cụ thể trong hình ảnh và video. Với kỹ thuật này, bạn có thể sử dụng deep learning để nhận diện khuôn mặt, giúp xác định danh tính của bạn qua các đặc điểm độc đáo.

1.6 Xe tự lái

Các phương tiện tự lái sử dụng deep learning để học cách vận hành và xử lý các tình huống khác nhau khi lái xe, đồng thời cho phép các phương tiện phát hiện đèn giao thông, nhận diện biển báo và tránh người đi bộ.

1.7 Phân tích dự đoán

Các mô hình deep learning có thể phân tích một lượng lớn thông tin lịch sử để đưa ra dự đoán chính xác về tương lai. Phân tích dự đoán giúp các doanh nghiệp trong nhiều lĩnh vực, bao gồm dự báo doanh thu, phát triển sản phẩm, ra quyết định và sản xuất.

1.8 Hệ thống đề xuất

Các dịch vụ trực tuyến thường sử dụng hệ thống đề xuất với khả năng nâng cao nhờ các mô hình deep learning. Với đủ dữ liệu, các mô hình deep learning này có thể dự đoán xác suất của các tương tác nhất định dựa trên lịch sử của các tương tác trước đó. Các ngành công nghiệp như dịch vụ phát trực tuyến, thương mại điện tử và mạng xã hội đều sử dụng hệ thống đề xuất.

1.9 Chăm sóc sức khỏe

Các ứng dụng deep learning trong ngành chăm sóc sức khỏe phục vụ nhiều mục đích. Không chỉ có thể hỗ trợ phát triển các giải pháp điều trị, các thuật toán deep learning còn có khả năng hiểu các hình ảnh y tế và giúp bác sĩ chẩn đoán bệnh nhân bằng cách phát hiện các tế bào ung thư.

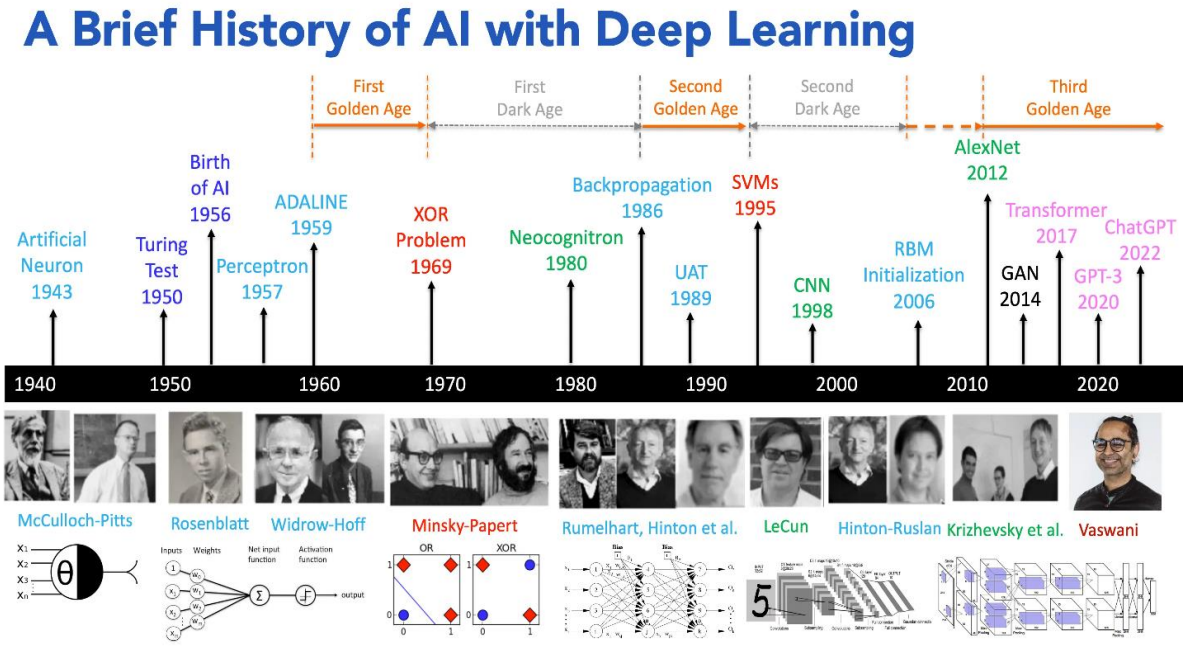
1.10 Công nghiệp

Các ứng dụng deep learning trong tự động hóa công nghiệp giúp bảo vệ an toàn cho công nhân trong các nhà máy bằng cách cho phép máy móc phát hiện các tình huống nguy hiểm, chẳng hạn như khi các vật thể hoặc con người ở quá gần máy móc.

2. Hiểu biết về mạng nơ-ron sâu

Mạng nơ-ron sâu là sự mở rộng của các mạng nơ-ron nhân tạo truyền thống. So với mạng nơ-ron truyền thống, có hai điểm khác biệt chính mà mạng nơ-ron sâu có. Mạng nơ-ron truyền thống thường chỉ có một hoặc hai lớp ẩn, khiến chúng khá nông. Trong khi đó, mạng nơ-ron sâu có rất nhiều lớp ẩn. Ví dụ, dự án Google Brain đã sử dụng một mạng nơ-ron với hàng triệu nơ-ron. Có rất nhiều mô hình cho mạng nơ-ron sâu, từ DNNs, CNNs, RNNs, cho đến LSTMs. Các nghiên cứu gần đây còn mang đến mạng dựa trên sự chú ý (attention-based networks) tập trung vào các phần cụ thể của một mạng nơ-ron sâu. Mạng càng lớn và càng có nhiều lớp, thì mạng càng phức tạp, đòi hỏi nhiều tài nguyên và thời gian huấn luyện hơn. Mạng nơ-ron sâu hoạt động tốt nhất với kiến trúc dựa trên GPU, vì GPU cần ít thời gian huấn luyện hơn so với các CPU truyền thống, trong khi các phát triển gần đây đã rút ngắn đáng kể thời gian huấn luyện.

3. Dòng thời gian của học sâu



Hình1.1 Lịch sử deep learning

1943 – **Mô hình nơ-ron sinh học**: Warren McCulloch và Walter Pitts giới thiệu mô hình toán học của nơ-ron sinh học, được xem là nền tảng cho các mạng nơ-ron nhân tạo sau này.

1957 – **Perceptron**: Frank Rosenblatt phát minh ra perceptron, một thuật toán cơ bản của mạng nơ-ron nhân tạo. Đây là bước đầu tiên hướng tới việc mô phỏng cách nơ-ron hoạt động trong não.

1960s-1970s – **Stagnation Period**: Perceptron bị hạn chế bởi khả năng xử lý các bài toán phi tuyến tính. Mối quan tâm đối với mạng nơ-ron suy giảm do các mô hình khác như SVM và các phương pháp thống kê trở nên phổ biến.

1986 – **Thuật toán lan truyền ngược (Backpropagation)**: Geoffrey Hinton, David Rumelhart và Ronald J. Williams phát triển thuật toán lan truyền ngược, giúp huấn luyện mạng nơ-ron có nhiều lớp ẩn. Đây là một cột mốc quan trọng trong sự phát triển của học sâu.

1990s – **Mạng nơ-ron hồi tiếp (RNNs)** và **LSTM**: RNNs được phát triển để xử lý dữ liệu tuần tự như chuỗi thời gian hoặc ngôn ngữ tự nhiên. Hochreiter và Schmidhuber giới thiệu **LSTM (Long Short-Term Memory)**, một biến thể của RNNs để xử lý các vấn đề về vanishing gradient.

2006 – **Sự trở lại của học sâu (Deep Learning Revival)**: Geoffrey Hinton và các đồng nghiệp phát triển kỹ thuật huấn luyện mạng nơ-ron sâu hiệu quả bằng cách sử dụng

mạng Boltzmann hạn chế (Restricted Boltzmann Machine), đánh dấu sự trở lại của deep learning.

2012 – **ImageNet và CNNs**: Mạng nơ-ron tích chập (CNNs) của Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton đạt thành tích đột phá trong cuộc thi ImageNet, giảm đáng kể lỗi phân loại hình ảnh. Sự kiện này là một cột mốc quan trọng đưa deep learning vào trung tâm của AI.

2014 – **GANs (Generative Adversarial Networks)**: Ian Goodfellow giới thiệu mạng GANs, một trong những đột phá lớn trong lĩnh vực tạo nội dung. GANs được dùng để tạo ra hình ảnh, âm thanh và video nhân tạo.

2017 – **Transformers và Attention Mechanism**: Google phát triển mô hình **Transformer** trong bài báo "Attention is All You Need", cách mạng hóa xử lý ngôn ngữ tự nhiên (NLP) và hình thành nền tảng cho các mô hình lớn như GPT.

2020s – Sự phát triển của các mô hình lớn (Large-scale models): Các mô hình ngôn ngữ lớn (LLMs) như GPT-3, GPT-4, và BERT của Google đã đạt được những thành công lớn trong NLP, tiếp tục mở rộng khả năng của deep learning.

4. Học sâu và học máy

Học máy và học sâu đều là những loại của trí tuệ nhân tạo (AI). Nói ngắn gọn, học máy là AI có thể tự động thích ứng với sự can thiệp tối thiểu từ con người. Học sâu là một nhánh con của học máy, sử dụng các mạng nơ-ron nhân tạo để bắt chước quá trình học hỏi của não người.

Học máy	Học sâu
Một nhánh con của AI	Một nhánh con của học máy
Có thể huấn luyện trên các tập dữ liệu nhỏ hơn	Yêu cầu lượng dữ liệu lớn
Yêu cầu cần nhiều sự can thiệp của con người để điều chỉnh và học hỏi	Tự học từ môi trường và các sai sót trước đó mà không cần can thiệp từ con người
Thời gian huấn luyện ngắn hơn và độ chính xác thấp hơn	Thời gian huấn luyện dài hơn và độ chính xác cao hơn
Tạo ra các mối tương quan đơn giản và tuyến tính	Tạo ra các mối quan hệ phức tạp và phi tuyến
Có thể huấn luyện trên CPU (bộ vi xử lý trung tâm)	Cần một GPU chuyên dụng (bộ xử lý đồ họa) để huấn luyện

5. Tìm hiểu về cách hoạt động của mạng nơ-ron

Bộ não con người chính là nguồn cảm hứng cho kiến trúc mạng nơ-ron. Các tế bào não của con người, còn được gọi là nơ-ron, tạo thành một mạng lưới phức tạp, có tính liên kết cao và gửi các tín hiệu điện đến nhau để giúp con người xử lý thông tin. Tương tự, một mạng nơ-ron nhân tạo được tạo ra từ các tế bào nơ-ron nhân tạo, cùng nhau phối hợp để giải quyết một vấn đề. Nơ-ron nhân tạo là các mô đun phần mềm, được gọi là nút và mạng nơ-ron nhân tạo là các chương trình phần mềm hoặc thuật toán mà về cơ bản, sử dụng hệ thống máy tính để giải quyết các phép toán.

5.1 Một mạng nơ-ron cơ bản bao gồm các nơ-ron nhân tạo liên kết theo 3 lớp:

5.1.1 Lớp đầu vào

Thông tin từ thế giới bên ngoài đi vào mạng nơ-ron nhân tạo qua lớp đầu vào. Các nút đầu vào xử lý dữ liệu, phân tích hoặc phân loại và sau đó chuyển dữ liệu sang lớp tiếp theo.

5.1.2 Lớp ẩn

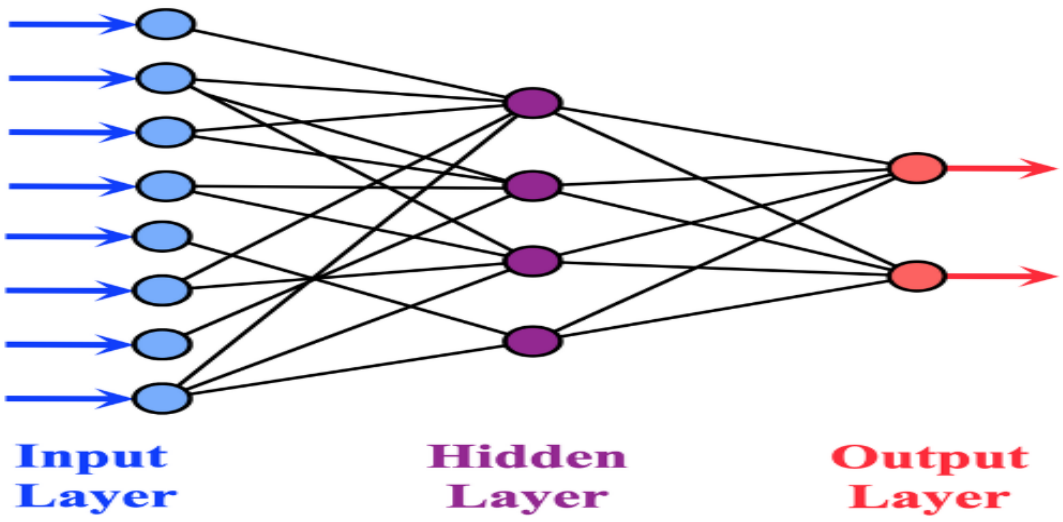
Dữ liệu đi vào lớp ẩn đến từ lớp đầu vào hoặc các lớp ẩn khác. Mạng nơ-ron nhân tạo có thể có một số lượng lớn lớp ẩn. Mỗi lớp ẩn phân tích dữ liệu đầu ra từ lớp trước, xử lý dữ liệu đó sâu hơn và rồi chuyển dữ liệu sang lớp tiếp theo.

5.1.3 Lớp đầu ra

Lớp đầu ra cho ra kết quả cuối cùng của tất cả dữ liệu được xử lý bởi mạng nơ-ron nhân tạo. Lớp này có thể có một hoặc nhiều nút. Ví dụ: giả sử chúng ta gặp phải một vấn đề phân loại nhị phân (có/không), lớp đầu ra sẽ có một nút đầu ra, nút này sẽ cho kết quả 1 hoặc 0. Tuy nhiên, nếu chúng ta gặp phải vấn đề phân loại nhiều lớp, lớp đầu ra sẽ có thể bao gồm nhiều hơn một nút đầu ra.

6. Mạng nơ-ron nông (shallow neural network) là gì?

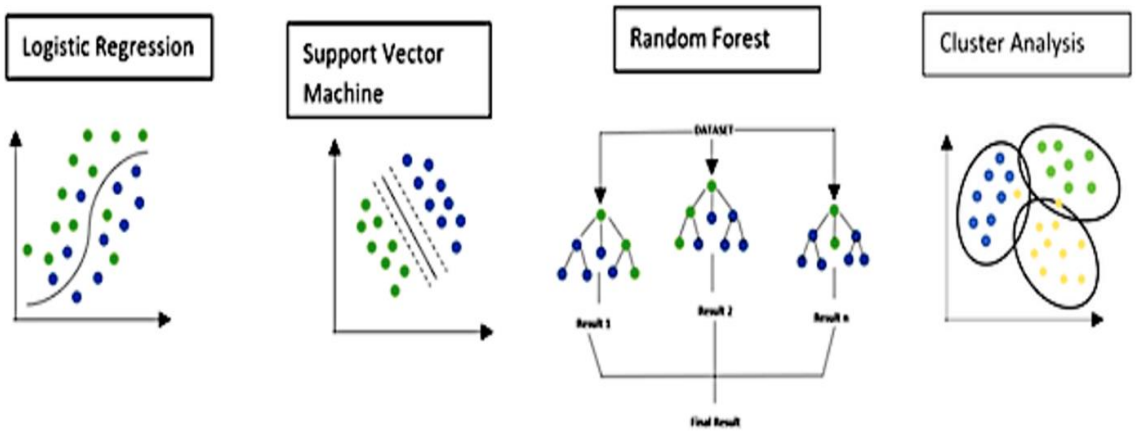
Mạng nơ-ron nông chỉ có một (hoặc chỉ một vài) lớp ẩn giữa lớp đầu vào và lớp đầu ra. Lớp đầu vào nhận dữ liệu, lớp ẩn (các lớp ẩn) xử lý dữ liệu đó, và lớp cuối cùng tạo ra đầu ra. Mạng nơ-ron nông đơn giản hơn, dễ huấn luyện hơn và có hiệu suất tính toán cao hơn so với mạng nơ-ron sâu, mà có thể có hàng nghìn đơn vị ẩn trong hàng chục lớp. Mạng nơ-ron nông thường được sử dụng cho các nhiệm vụ đơn giản như hồi quy tuyến tính, phân loại nhị phân, hoặc trích xuất đặc trưng trong không gian thấp.



Hình1.2 Shallow Neural Network

Về mặt kỹ thuật, một mạng nơ-ron có nhiều hơn một lớp ẩn không còn được coi là mạng nơ-ron nông nữa. Tuy nhiên, trong một số trường hợp, một mạng nơ-ron có 2 hoặc 3 lớp ẩn, mỗi lớp có một số lượng nhỏ đơn vị ẩn và có kết nối đơn giản giữa chúng có thể tạo ra đầu ra đơn giản và vẫn được coi là một mạng "nông".

Bốn kỹ thuật học máy nông phổ biến thường được sử dụng trong các ứng dụng xử lý hình ảnh cho MRI:



Hình1.3 Various Shallow Neural Networks

7. Hàm kích hoạt (activation function) là gì và tại sao lại sử dụng nó?

Hàm kích hoạt là một thành phần quan trọng trong mạng nơ-ron, xác định xem một nơ-ron có nên được kích hoạt dựa trên đầu vào của nó. Các hàm này sử dụng các phép toán toán học để đánh giá xem đầu vào có quan trọng cho việc dự đoán hay không. Nếu đầu vào được coi là quan trọng, hàm sẽ "kích hoạt" nơ-ron.

Một hàm kích hoạt tạo ra đầu ra từ một tập hợp các giá trị đầu vào được cung cấp cho một nút hoặc lớp. Trong mạng nơ-ron, một nút tương tự như một nơ-ron trong não người, nó nhận các tín hiệu đầu vào (các kích thích bên ngoài) và phản ứng lại. Não bộ xử lý các tín hiệu này và quyết định xem có kích hoạt nơ-ron hay không dựa trên các con đường đã hình thành theo thời gian và cường độ phát tín hiệu của nơ-ron.

Trong học sâu, hàm kích hoạt thực hiện vai trò tương tự. Mục tiêu chính của hàm kích hoạt là biến đổi đầu vào đã được tổng hợp và trọng số từ một nút thành giá trị đầu ra, sau đó được truyền tới lớp ẩn tiếp theo hoặc được sử dụng làm đầu ra cuối cùng.

Hầu hết các hàm kích hoạt là phi tuyến, điều này cho phép mạng nơ-ron "học" các đặc trưng của tập dữ liệu, chẳng hạn như cách các pixel khác nhau tạo thành một đặc trưng trong hình ảnh. Nếu không có các hàm kích hoạt phi tuyến, mạng nơ-ron chỉ có thể học các hàm tuyến tính và phép biến đổi affine. Điều này là một vấn đề vì các hàm tuyến tính và hàm affine không thể nắm bắt được các mẫu phức tạp và phi tuyến thường xuất hiện trong dữ liệu thực tế.

Ngoài ra, các hàm kích hoạt còn giúp ánh xạ một dòng đầu vào có phân phối và quy mô không xác định sang một phân phối đã biết. Điều này làm ổn định quá trình huấn luyện

và ánh xạ các giá trị đến đầu ra mong muốn cho các tác vụ không hồi quy như phân loại, mô hình sinh, hoặc học củng cố. Các hàm kích hoạt phi tuyến làm tăng độ phức tạp cho mạng nơ-ron, cho phép chúng thực hiện các tác vụ nâng cao hơn.

8. Các loại hàm kích hoạt

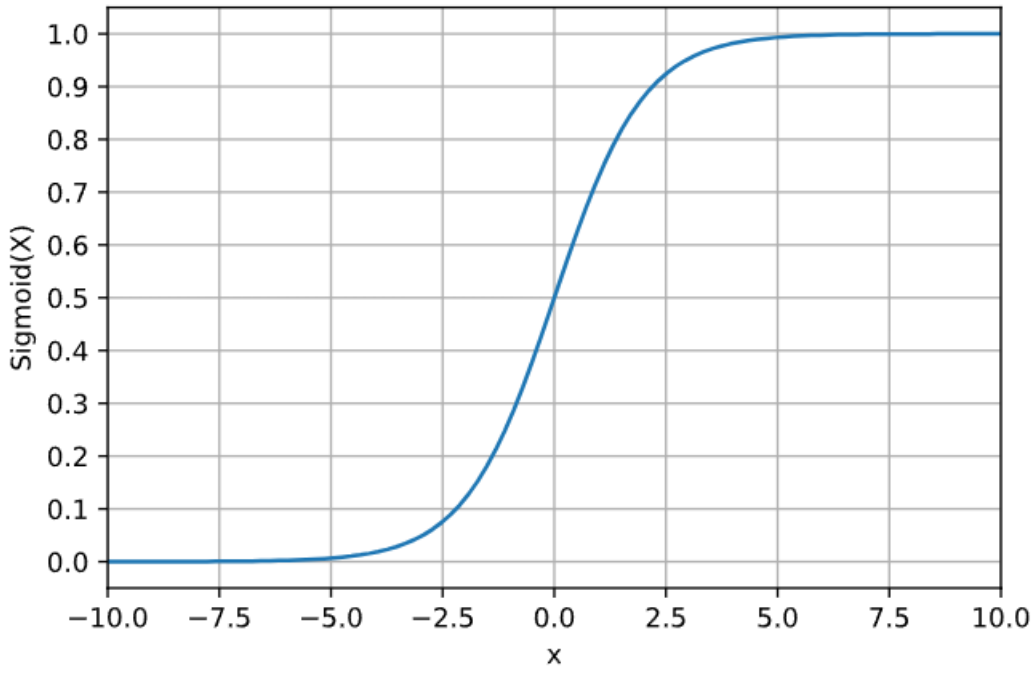
Các hàm kích hoạt trong mạng nơ-ron giúp thêm tính phi tuyến vào mô hình, cho phép nó học được các mẫu phức tạp. Các loại hàm kích hoạt phổ biến là:

8.1 Hàm Sigmoid:

Công thức:

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

Phân tích:



Hình1.4 Biểu đồ của 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 số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1. Trong quá khứ hàm Sigmoid hay được dùng vì có đạo hàm rất đẹp. Tuy nhiên hiện nay hàm Sigmoid rất ít được dùng vì những nhược điểm sau:

Hàm Sigmod: Đây là một hàm được biểu diễn dưới dạng đồ thị hình chữ **'S'.** Thường được sử dụng ở lớp đầu ra của phân loại nhị phân, trong đó kết quả là 0 hoặc 1, vì giá trị của hàm sigmoid chỉ nằm giữa 0 và 1, do đó, kết quả có thể dễ dàng được dự đoán là ***1*** nếu giá trị lớn hơn **0,5** và ***0*** nếu không.

Hàm Sigmoid bão hào và triệt tiêu gradient: Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).

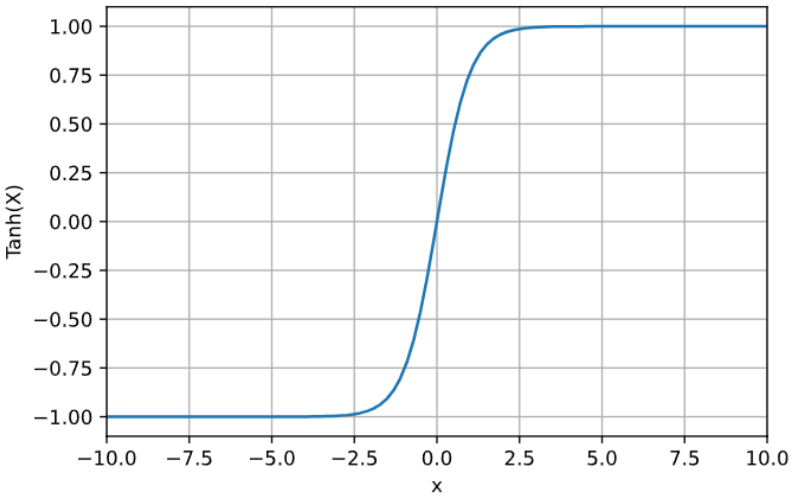
Chuyển đổi giá trị đầu vào thành một giá trị trong khoảng từ 0 đến 1.

8.2 Hàm Tanh (Hyperbolic Tangent):

Công thức:

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

Phân tích:



Hình1.5. Biểu đồ của hàm tanh

Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1). Cũng như Sigmoid, hàm Tanh bị bão hoà ở 2 đầu (gradient thay đổi rất ít ở 2 đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid.

Hàm tanh còn có thể được biểu diễn bằng hàm sigmoid như sau:

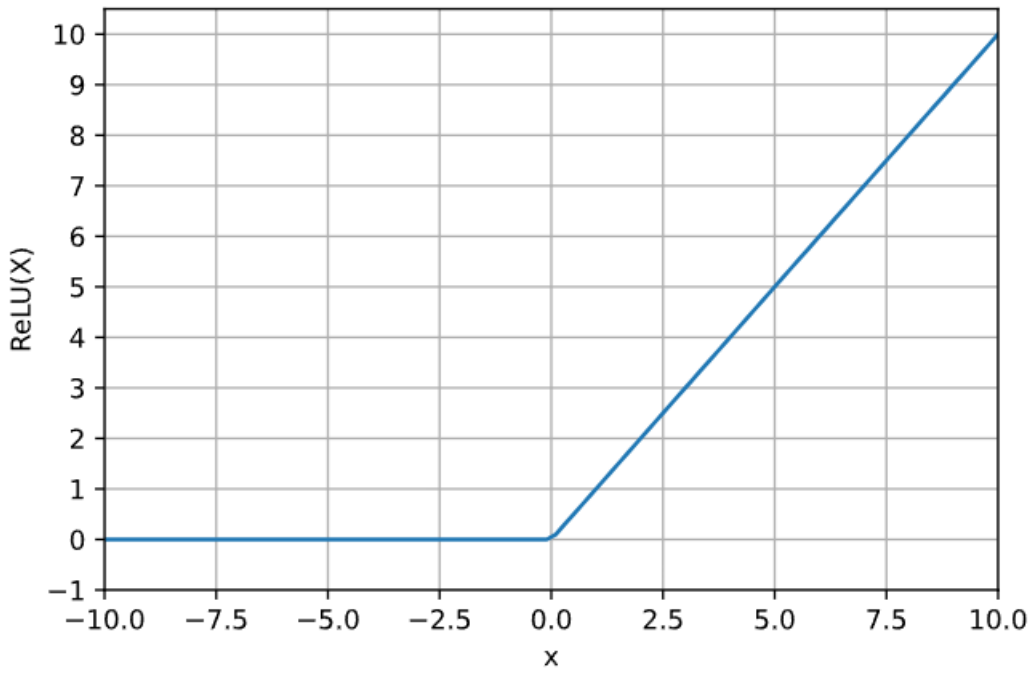
tanh(x)=2σ(2x) −1tanh(*x*)=2*σ*(2*x*) −1

8.3 ReLU (Rectified Linear Unit):

Công thức:

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

Phân tích:



Hình1.6 Biểu đồ của 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. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó

- Đặt các giá trị âm thành 0 và giữ nguyên các giá trị dương.

8.4 Leaky ReLU:

Công thức:

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x) \text{ với } \alpha \text{ là hằng số nhỏ.}$$

Phân tích:

Leaky ReLU là một cố gắng trong việc loại bỏ "dying ReLU". Thay vì trả về giá trị 0 với các đầu vào < 0 thì Leaky ReLU tạo ra một đường xiên có độ dốc nhỏ (xem đồ thị). Có nhiều báo cáo về việc hiệu Leaky ReLU có hiệu quả tốt hơn ReLU, nhưng hiệu quả này vẫn chưa rõ ràng và nhất quán.

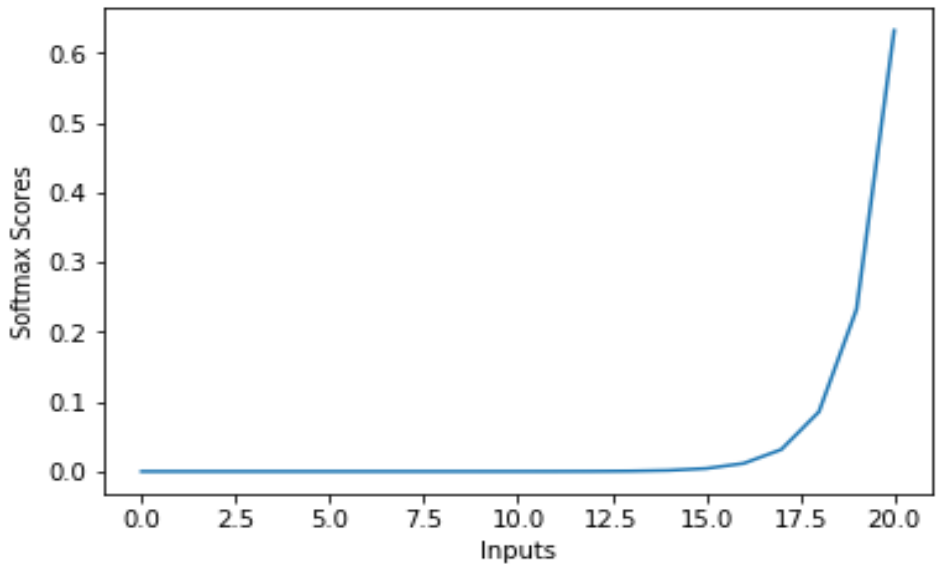
Ngoài Leaky ReLU có một biến thể cũng khá nổi tiếng của ReLU là PReLU. PReLU tương tự Leaky ReLU nhưng cho phép neuron tự động chọn hệ số $\alpha\alpha$ tốt nhất.

8.5 Hàm Softmax:

Công thức:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Phân tích:



Hình1.7 Biểu đồ của hàm Softmax

Hàm Softmax: Hàm softmax cũng là một loại hàm sigmoid nhưng rất tiện dụng khi chúng ta cố gắng xử lý các vấn đề phân loại đa lớp. Hàm softmax được sử dụng lý tưởng ở lớp đầu ra của bộ phân loại, nơi chúng ta thực sự cố gắng đạt được xác suất để xác định lớp của từng đầu vào.

Được sử dụng trong lớp đầu ra cho các bài toán phân loại.

9. Hàm mất mát là gì?

Loss function (hàm mất mát) là một thành phần quan trọng trong các mô hình học máy, đặc biệt là trong quá trình huấn luyện mạng nơ-ron. Nó được sử dụng để đo lường độ chính xác của mô hình bằng cách tính toán sự khác biệt giữa giá trị dự đoán của mô hình và giá trị thực tế.

Loss function (hàm mất mát) là hàm cho phép xác định mức độ sai khác của kết quả dự đoán so với giá thực thực cần dự đoán. Nó là một phương pháp đo lường chất lượng của mô hình dự đoán trên tập dữ liệu quan sát. Nếu mô hình dự đoán sai hiều thì giá trị của loss function sẽ càng lớn và ngược lại nếu nó dự đoán càng đúng thì giá trị của loss function sẽ càng thấp.

Loss function kí hiệu là L, là thành phần cốt lõi của evaluation function và objective function. Cụ thể, trong công thức thường gặp:

$$\mathcal{L}_D(f_w) = \frac{1}{|D|} \sum_{(x,y)\in D} L\left(f_w(x), y\right)$$

Hàm mất mát (loss function) trả về một giá trị thực không âm, thể hiện sự chênh lệch giữa hai đại lượng: y’ (giá trị dự đoán) và y (giá trị thực). Hàm mất mát có thể được coi như một hình thức phạt cho mô hình mỗi khi nó dự đoán sai. Mức phạt này tỉ lệ thuận với độ nghiêm trọng của sai sót.

Trong mọi bài toán học có giám sát (supervised learning), mục tiêu chính của chúng ta là giảm thiểu tổng mức phạt mà mô hình phải chịu. Trong trường hợp lý tưởng, khi y’= y, hàm mất mát sẽ trả về giá trị cực tiểu là 0, cho thấy rằng mô hình đã dự đoán chính xác.\

10. Các loại hàm mất mát

Hàm mất mát trong học máy có thể được phân loại dựa trên các nhiệm vụ học máy mà chúng áp dụng. Hầu hết các hàm mất mát đều áp dụng cho các bài toán hồi quy và phân loại trong học máy. Đối với các nhiệm vụ học máy hồi quy, mô hình được kỳ vọng sẽ dự

đoán các giá trị đầu ra liên tục. Ngược lại, đối với các nhiệm vụ phân loại, mô hình được kỳ vọng sẽ cung cấp các nhãn rời rạc tương ứng với các lớp của tập dữ liệu.

Các hàm mất mát phổ biến bao gồm hàm ma cơ bản dành cho phân loại nhị phân

10.1 0-1 Los:

Là hàm mất mát đơn giản nhất, nó trả về giá trị 1 nếu dự đoán sai và 0 nếu dự đoán đúng. Thường được sử dụng trong các bài toán phân loại nhị phân.

10.2 Perceptron Loss:

Hàm mất mát này được sử dụng trong thuật toán perceptron. Nó phạt mô hình khi dự đoán sai bằng cách tính toán khoảng cách giữa điểm dữ liệu và mặt phẳng phân cách. Nếu dự đoán đúng, hàm mất mát trả về 0.

$$L_{perceptron}(\hat{y}, y) = \max(0, -y \cdot \hat{y})$$

10.3 Hinge Loss:

Thường được sử dụng trong máy vector hỗ trợ (SVM), hàm mất mát này cho phép tạo ra một khoảng cách giữa các lớp. Nó phạt những dự đoán sai và cả những dự đoán đúng nhưng không đủ mạnh (không vượt qua một ngưỡng nhất định).

$$L_{hinge}(\hat{y}, y) = \max(0, 1 - y \cdot \hat{y})$$

10.4 Logistic Loss (hay Log Loss):

Được sử dụng trong hồi quy logistic, hàm mất mát này tính toán xác suất mà một mẫu thuộc về một lớp cụ thể. Nó phạt các dự đoán dựa trên xác suất và cho phép tối ưu hóa mô hình trong các bài toán phân loại.

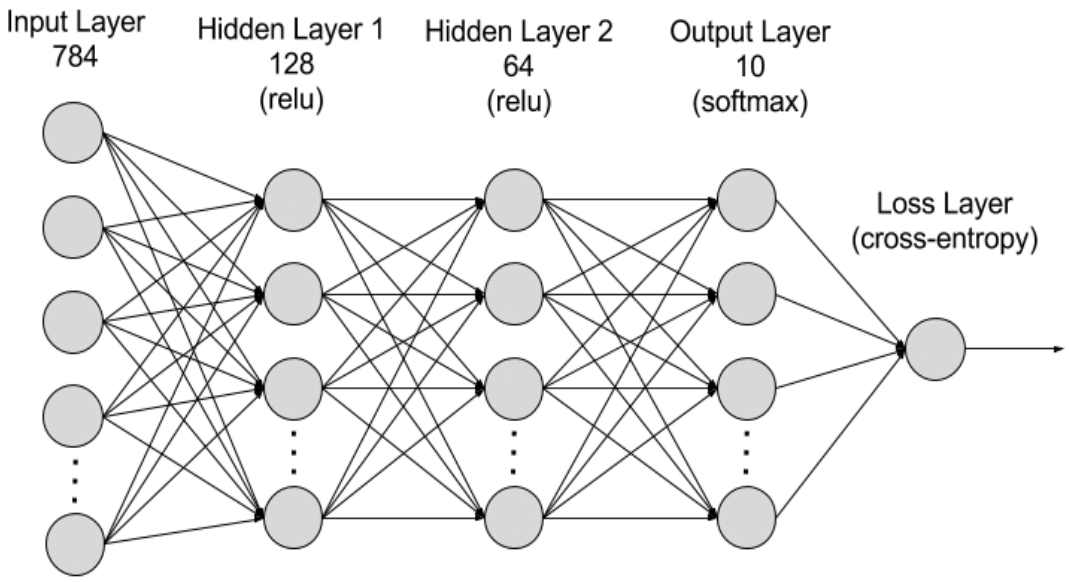
$$L_{log}(\hat{y}, y) = \log_2(1 + \exp(-y \cdot \hat{y}))$$

11. Học sâu hoạt động như thế nào?

Các thuật toán học sâu là các mạng nơ-ron được lập mô hình theo bộ não con người. Ví dụ: một bộ não con người chứa hàng triệu nơ-ron được kết nối với nhau, làm việc cùng nhau để tìm hiểu và xử lý thông tin. Tương tự, các mạng nơ-ron học sâu, hay mạng nơ-ron nhân tạo, được tạo thành từ nhiều lớp nơ-ron nhân tạo hoạt động cùng nhau bên trong máy tính.

Các nơ-ron nhân tạo là những mô-đun phần mềm được gọi là nút, sử dụng các phép toán để xử lý dữ liệu. Các mạng nơ-ron nhân tạo là những thuật toán học sâu sử dụng các nút này để giải quyết các vấn đề phức tạp.

Học sâu dựa trên mạng nơ-ron với nhiều lớp. Mỗi lớp bao gồm các nơ-ron, và thông tin được truyền từ lớp này sang lớp khác, với các trọng số và độ chệch được áp dụng ở mỗi bước. Quá trình huấn luyện bao gồm hai bước chính: truyền xuôi và truyền ngược. Trong truyền xuôi, dữ liệu đầu vào được đưa qua mạng để tạo ra dự đoán. Truyền ngược điều chỉnh các trọng số và độ chệch dựa trên sai số được tính toán từ hàm mất mát, giúp mạng học dần qua thời gian.



Hình1.8. Lớp mất mát (cross-entropy)

12. Tiền xử lý dữ liệu

Trước khi huấn luyện mạng nơ-ron, dữ liệu cần được chuẩn bị. Các bước tiền xử lý phổ biến bao gồm:

- Chuẩn hóa/Tiêu chuẩn hóa: Tỷ lệ hóa dữ liệu đầu vào trong một khoảng cụ thể (ví dụ: từ 0 đến 1) hoặc tiêu chuẩn hóa để có trung bình bằng 0 và độ lệch chuẩn bằng 1.

- Xử lý giá trị thiếu: Điền vào hoặc loại bỏ các giá trị thiếu.

- Mã hóa One-Hot: Chuyển đổi các biến phân loại thành định dạng nhị phân.

- Chia dữ liệu: Chia tập dữ liệu thành tập huấn luyện, xác thực, và kiểm tra.

13. Truyền xuôi

Truyền xuôi là quá trình truyền dữ liệu đầu vào qua mạng nơ-ron để tạo ra dự đoán. Trong mạng nơ-ron, đầu vào được nhân với các trọng số và qua các lớp, với hàm kích hoạt được áp dụng tại mỗi lớp, để đưa ra đầu ra cuối cùng.

13.1 Quy trình:

Dữ liệu đầu vào được đưa vào mạng.

Các trọng số và độ chệch được áp dụng lên đầu vào.

Hàm kích hoạt xử lý đầu ra từ mỗi lớp.

Kết quả cuối cùng là đầu ra dự đoán của mạng.

14. Truyền ngược (Backward propagation)

Truyền ngược là quá trình điều chỉnh trọng số dựa trên sai số giữa đầu ra dự đoán và giá trị thực. Thông qua việc lan truyền gradient ngược lại qua các lớp, mạng sẽ cập nhật trọng số để giảm thiểu sai số trong các lần dự đoán tiếp theo.

Backpropagation , viết tắt của "backward propagation of errors", là một thuật toán để học có giám sát các mạng nơ-ron nhân tạo sử dụng gradient descent . Với một mạng nơ-ron nhân tạo và một hàm lỗi , phương pháp này tính toán gradient của hàm lỗi theo trọng số của mạng nơ-ron. Đây là một khái quát của quy tắc delta cho perceptron thành mạng nơ-ron truyền thẳng nhiều lớp.

Phần "ngược" của tên bắt nguồn từ thực tế là phép tính gradient diễn ra ngược qua mạng, với gradient của lớp trọng số cuối cùng được tính trước và gradient của lớp trọng số đầu tiên được tính sau. Các phép tính một phần của gradient từ một lớp được sử dụng lại trong phép tính gradient cho lớp trước đó. Luồng thông tin lỗi ngược này cho phép tính hiệu quả gradient ở mỗi lớp so với cách tiếp cận ngây thơ là tính gradient của từng lớp riêng biệt.

Sự phổ biến của backpropagation đã có sự hồi sinh gần đây do việc áp dụng rộng rãi các mạng nơ-ron sâu để nhận dạng hình ảnh và nhận dạng giọng nói. Nó được coi là một

thuật toán hiệu quả và các triển khai hiện đại tận dụng GPU chuyên dụng để cải thiện hiệu suất hơn nữa.

14.1 Quy trình:

Tính toán lỗi dựa trên đầu ra và giá trị thực.

Lan truyền lỗi ngược qua các lớp của mạng.

Cập nhật trọng số để giảm thiểu sai số.

15. Xây dựng mạng nơ-ron sử dụng NumPy

Các bước xây dựng mạng nơ-ron sử dụng NumPy bao gồm 10 bước

1. Chuẩn bị dữ liệu.
2. Xác định kiến trúc mạng.
3. Khởi tạo trọng số và độ chệch.
4. Thực hiện truyền xuôi để tạo ra đầu ra dự đoán.
5. Tính toán sai số giữa dự đoán và nhãn thực tế.
6. Truyền ngược sai số để tính toán gradient.
7. Cập nhật trọng số.
8. Lặp lại quá trình này qua nhiều lần huấn luyện (epochs).
9. Đánh giá mô hình.
10. Tối ưu hóa và tinh chỉnh.

15.1 Chuẩn bị dữ liệu đầu vào

Thu thập và chuẩn bị dữ liệu. Mỗi mẫu dữ liệu sẽ có một tập các đặc trưng (input features) và một giá trị mục tiêu (target/output).

Đảm bảo rằng dữ liệu đã được chuẩn hóa hoặc tiêu chuẩn hóa nếu cần, ví dụ như giá trị đầu vào trong khoảng [0,1] hoặc [-1,1].

15.2 Xác định kiến trúc mạng

Xác định số lượng lớp (layers) của mạng nơ-ron.

Xác định số lượng nơ-ron trong mỗi lớp (input layer, hidden layers, output layer). Chọn hàm kích hoạt cho mỗi lớp (ví dụ: Sigmoid, ReLU, Tanh, Softmax).

15.3 Khởi tạo trọng số và độ chệch (bias)

Khởi tạo các trọng số và độ chệch ngẫu nhiên với giá trị nhỏ. Trọng số là các hệ số liên kết giữa các nơ-ron của các lớp.

Số lượng trọng số phụ thuộc vào số lượng kết nối giữa các lớp.

15.4 Truyền xuôi (Forward Propagation)

Dữ liệu đầu vào được nhân với trọng số tương ứng và cộng với độ chệch.

Áp dụng hàm kích hoạt (ví dụ: Sigmoid, ReLU) để tạo ra đầu ra cho từng lớp.

Kết quả đầu ra của một lớp sẽ trở thành đầu vào của lớp tiếp theo, tiếp tục cho đến lớp cuối cùng để tạo ra đầu ra dự đoán.

15.5 Tính toán sai số (Loss Calculation)

Tính toán sai số giữa giá trị dự đoán và giá trị thực tế (target) bằng cách sử dụng một hàm mất mát (loss function), ví dụ: Mean Squared Error (MSE) cho bài toán hồi quy, hoặc Cross-Entropy cho bài toán phân loại.

Sai số cho biết mức độ khác biệt giữa dự đoán của mô hình và giá trị thực.

15.6 Truyền ngược (Backward Propagation)

Sử dụng thuật toán lan truyền ngược để tính gradient của hàm mất mát đối với các trọng số (đạo hàm của hàm mất mát).

Lan truyền gradient ngược qua các lớp của mạng, từ đầu ra về phía đầu vào, để xác định mức độ ảnh hưởng của các trọng số đến sai số.

15.7 Cập nhật trọng số

Dùng gradient tính được để cập nhật trọng số và độ chệch, thường thông qua một thuật toán tối ưu hóa như Gradient Descent.

Công thức cập nhật: Trọng số mới = Trọng số cũ - learning rate gradient.

Hệ số học (learning rate) điều chỉnh tốc độ thay đổi của trọng số.

15.8 Lặp lại quá trình huấn luyện (Training Loop)

Thực hiện truyền xuôi và truyền ngược nhiều lần qua toàn bộ dữ liệu (nhiều epoch) cho đến khi sai số hội tụ hoặc đạt được mức mong muốn.

Sau mỗi lần lặp, trọng số được cập nhật và mô hình trở nên chính xác hơn.

15.9 Đánh giá mô hình

Sau khi hoàn tất huấn luyện, sử dụng một tập dữ liệu kiểm tra (test set) để đánh giá độ chính xác của mô hình.

Tính toán các chỉ số hiệu suất như độ chính xác (accuracy), độ chính xác dự đoán (precision), và độ nhạy (recall) để đánh giá mô hình.

15.10 Tối ưu hóa và điều chỉnh

Tinh chỉnh các siêu tham số như số lớp ẩn, số lượng nơ-ron, learning rate, và số epoch để cải thiện hiệu suất của mô hình.

Kiểm tra và điều chỉnh để tránh hiện tượng overfitting hoặc underfitting.

TÀI LIỆU THAM KHẢO

1. https://getthematic.com/insights/what-is-deep-learning/

2. https://levity.ai/blog/difference-machine-learning-deep-learning

3. https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide

4. https://cloud.google.com/discover/deep-learning-vs-machine-learning

5. https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,that%20resembles%20the%20human%20brain.

6. https://mize.tech/blog/how-does-a-neural-network-work-implementation-and-5-examples/

7. https://viblo.asia/p/shallow-neural-networks-Do7546y0ZM6

8. https://khanh-personal.gitbook.io/ml-book-vn/chapter1/ham-mat-mat