Neural Network

Họ và tên: Mai Minh Quân

MSSV: 20225661

Nguồn tham khảo: Neural Networks and Deep Learning - Michael Nielsen

# 1. Perceptron

Perceptron là một loại nơ-ron nhân tạo sơ khai, được phát triển bởi Frank Rosenblatt vào những năm 1950 và 1960. Nó nhận nhiều đầu vào nhị phân và tạo ra một đầu ra nhị phân duy nhất.

1. Cấu trúc và Nguyên tắc hoạt động

Một perceptron bao gồm các thành phần sau:

* Đầu vào (): Các giá trị nhị phân (0 hoặc 1) được đưa vào perceptron.
* Trọng số (): Các số thực biểu thị tầm quan trọng của mỗi đầu vào đối với đầu ra.
* Giá trị ngưỡng (threshold): Một số thực quyết định liệu perceptron có kích hoạt hay không.
* Đầu ra: Một giá trị nhị phân (0 hoặc 1) được xác định dựa trên tổng có trọng số của các đầu vào so với ngưỡng.

A diagram of a circle with arrows

AI-generated content may be incorrect.

1. Mô hình Toán học

Đầu ra của perceptron được tính toán theo công thức sau:

1. Ví dụ về Quyết định

Xét ví dụ về việc quyết định có nên đi lễ hội phô mai hay không dựa trên các yếu tố: thời tiết tốt (), bạn bè muốn đi cùng (), và gần giao thông công cộng (). Chúng ta có thể gán trọng số cho từng yếu tố (ví dụ: cho thời tiết, và cho các yếu tố khác) và đặt một ngưỡng (ví dụ: ). Perceptron sẽ đưa ra quyết định dựa trên tổng có trọng số của các yếu tố này so với ngưỡng.

1. Mạng lưới Perceptron nhiều lớp

Các perceptron có thể được kết nối với nhau thành nhiều lớp để tạo thành một mạng lưới phức tạp hơn. Trong mạng lưới này, đầu ra của các perceptron ở lớp trước sẽ trở thành đầu vào cho các perceptron ở lớp sau. Mạng lưới perceptron nhiều lớp có khả năng giải quyết các vấn đề phức tạp hơn so với một perceptron đơn lẻ.

A network diagram of circles and lines

AI-generated content may be incorrect.

1. Đơn giản hóa ký hiệu

Để đơn giản hóa biểu thức, chúng ta có thể sử dụng ký hiệu tích vô hướng () thay cho tổng có trọng số () và giới thiệu khái niệm bias (). Với bias, quy tắc của perceptron trở thành:

Bias có thể được hiểu là mức độ dễ dàng để perceptron đưa ra đầu ra là 1.

# 2. Sigmoid Neuron

Mặc dù perceptron là một đơn vị tính toán hữu ích, nhưng mạng lưới perceptron gặp khó khăn trong việc học hiệu quả. Một thay đổi nhỏ ở trọng số hoặc bias của một perceptron có thể gây ra sự thay đổi đột ngột trong đầu ra của nó, dẫn đến việc khó điều chỉnh các tham số của mạng một cách trơn tru để đạt được hành vi mong muốn.

* 1. Giới thiệu Sigmoid Neuron

Để giải quyết vấn đề này, nơ-ron sigmoid được giới thiệu. Tương tự như perceptron, nơ-ron sigmoid nhận đầu vào, gán trọng số và có bias. Tuy nhiên, sự khác biệt chính nằm ở hàm kích hoạt. Thay vì hàm bước nhị phân, nơ-ron sigmoid sử dụng hàm sigmoid (σ):

Trong đó .

* 1. Hàm Sigmoid

Hàm sigmoid có dạng hình chữ S, nhận bất kỳ giá trị thực nào làm đầu vào và trả về một giá trị trong khoảng . Tính chất quan trọng của hàm sigmoid là nó là một hàm trơn và liên tục. Điều này có nghĩa là những thay đổi nhỏ ở đầu vào (trọng số hoặc bias) sẽ chỉ gây ra những thay đổi nhỏ tương ứng ở đầu ra của nơ-ron. Tính chất này cho phép mạng nơ-ron sigmoid học một cách hiệu quả hơn.

A diagram of a neural network

AI-generated content may be incorrect.

# 3. Kiến trúc của mạng nơ-ron

Một mạng nơ-ron điển hình bao gồm các lớp sau:

* Lớp đầu vào: Chứa các nơ-ron đầu vào, nhận dữ liệu đầu vào từ thế giới bên ngoài.
* Lớp ẩn: Một hoặc nhiều lớp nằm giữa lớp đầu vào và lớp đầu ra. Các nơ-ron trong lớp ẩn thực hiện các phép tính trung gian để trích xuất các đặc trưng phức tạp từ dữ liệu đầu vào.
* Lớp đầu ra: Chứa các nơ-ron đầu ra, tạo ra kết quả cuối cùng của mạng.

Mạng nơ-ron trong đó thông tin chỉ truyền theo một hướng từ lớp đầu vào đến lớp đầu ra được gọi là feedforward neural networks.

# 4. Học bằng Gradient Descent

Mục tiêu của việc học trong mạng nơ-ron là tìm ra các giá trị tối ưu cho trọng số và bias sao cho mạng có thể thực hiện tốt một nhiệm vụ cụ thể (ví dụ: phân loại hình ảnh). Để đạt được điều này, chúng ta sử dụng một hàm chi phí để đo lường hiệu suất của mạng. Hàm chi phí thường được định nghĩa sao cho nó có giá trị nhỏ khi mạng hoạt động tốt và giá trị lớn khi mạng hoạt động kém.

1. Hàm Chi phí

Một hàm chi phí phổ biến là hàm chi phí quadratic (hay còn gọi là mean squared error - MSE):

Trong đó:

* là tập hợp tất cả các trọng số trong mạng.
* là tập hợp tất cả các bias trong mạng.
* là số lượng mẫu huấn luyện.
* là một mẫu huấn luyện đầu vào.
* là đầu ra mong muốn cho đầu vào x.
* là đầu ra thực tế của mạng cho đầu vào x.

Mục tiêu là tìm các giá trị của và để giảm thiểu hàm chi phí .

1. Thuật toán Gradient Descent

Gradient descent là một thuật toán tối ưu hóa được sử dụng để tìm cực tiểu của một hàm số. Trong bối cảnh mạng nơ-ron, chúng ta sử dụng gradient descent để tìm các giá trị của trọng số và bias giúp giảm thiểu hàm chi phí.

Ý tưởng cơ bản của gradient descent là di chuyển các tham số (trọng số và bias) theo hướng ngược lại với gradient của hàm chi phí. Gradient cho biết hướng mà hàm chi phí tăng nhanh nhất. Bằng cách di chuyển theo hướng ngược lại, chúng ta hy vọng sẽ tiến gần hơn đến điểm cực tiểu của hàm chi phí.

Quy tắc cập nhật cho trọng số () và bias () trong thuật toán gradient descent là:

Trong đó là tốc độ học (learning rate), một tham số dương nhỏ quyết định kích thước của mỗi bước di chuyển.

1. Stochastic Gradient Descent

Khi tập dữ liệu huấn luyện rất lớn, việc tính toán gradient của hàm chi phí trên toàn bộ dữ liệu có thể tốn rất nhiều thời gian. Stochastic gradient descent (SGD) là một biến thể của gradient descent giúp tăng tốc độ học bằng cách ước tính gradient dựa trên một mẫu nhỏ ngẫu nhiên của dữ liệu huấn luyện, được gọi là mini-batch.

Thay vì tính toán gradient trên toàn bộ mẫu huấn luyện, SGD chỉ tính toán gradient trên một mini-batch gồm mẫu (). Điều này giúp giảm đáng kể thời gian tính toán cho mỗi bước cập nhật.

Để kết nối quá trình học trong mạng nơ-ron với phương pháp giảm dần gradient, giả sử ​ và ​ lần lượt là các trọng số và bias trong mạng nơ-ron. Phương pháp stochastic gradient descent hoạt động bằng cách chọn ra một mini-batch ngẫu nhiên từ các đầu vào huấn luyện, sau đó tiến hành huấn luyện với các dữ liệu này.

Trong đó, tổng số là trên tất cả các ví dụ huấn luyện ​ trong mini-batch hiện tại. Sau khi hoàn thành việc huấn luyện với một mini-batch, một mini-batch khác sẽ được chọn ngẫu nhiên và tiếp tục huấn luyện. Quá trình này sẽ được lặp lại cho đến khi tất cả các ví dụ huấn luyện đã được sử dụng, lúc đó ta hoàn thành một "epoch" huấn luyện. Sau mỗi epoch, quá trình huấn luyện sẽ bắt đầu lại với một bộ dữ liệu huấn luyện mới.

# 5. Backpropagation (Lan truyền ngược)

Backpropagation là một thuật toán hiệu quả để tính toán gradient của hàm chi phí đối với tất cả các trọng số và bias trong mạng nơ-ron. Đây là một thành phần cốt lõi của hầu hết các thuật toán học sâu.

* 1. Ký hiệu Ma trận

Để mô tả thuật toán backpropagation một cách hiệu quả, chúng ta sử dụng ký hiệu ma trận cho trọng số, bias và activation của các nơ-ron trong mạng.

* : Trọng số kết nối từ nơ-ron thứ ở lớp đến nơ-ron thứ ở lớp .
* : Bias của nơ-ron thứ ở lớp .
* : Activation của nơ-ron thứ ở lớp .
  1. Công thức Activation dưới dạng Ma trận

Activation của các nơ-ron ở lớp l có thể được tính toán từ các activations ở lớp l-1 bằng công thức ma trận:

Trong đó:

* , và là các vectơ cột chứa activations và biases của các lớp tương ứng.
* là ma trận trọng số kết nối lớp với lớp .
* là hàm sigmoid được áp dụng theo phần tử cho vectơ đầu vào.

Đại lượng được gọi là weighted input.

* 1. Lỗi của Nơ-ron

Trong thuật toán backpropagation, chúng ta định nghĩa lỗi () của một nơ-ron ở lớp có công thức như sau:

* 1. Bốn Phương trình cơ bản của Backpropagation

Thuật toán backpropagation dựa trên bốn phương trình cơ bản sau:

* Lỗi ở lớp đầu ra:

Trong đó là vectơ các đạo hàm riêng của hàm chi phí đối với các activations ở lớp đầu ra, và là tích Hadamard (tích theo phần tử).

* Lỗi ở lớp theo lỗi ở lớp :
* Tốc độ thay đổi của chi phí đối với bias:
* Tốc độ thay đổi của chi phí đối với trọng số:
  1. Thuật toán Backpropagation

Thuật toán backpropagation có thể được tóm tắt như sau:

1. Feedforward: Tính toán activations của tất cả các lớp trong mạng cho một đầu vào cụ thể.
2. Tính lỗi ở lớp đầu ra: Sử dụng Phương trình (BP1).
3. Lan truyền ngược lỗi: Tính toán lỗi cho các lớp trước đó bằng cách sử dụng Phương trình (BP2) lặp đi lặp lại từ lớp đầu ra trở về lớp thứ hai.
4. Tính gradient: Tính toán đạo hàm riêng của hàm chi phí đối với trọng số và bias bằng cách sử dụng Phương trình (BP3) và (BP4).
   1. Ảnh hưởng của Activation và Sự Bão hòa

Các phương trình backpropagation cho thấy rằng tốc độ học của một trọng số phụ thuộc vào activation của nơ-ron đầu vào và đạo hàm của hàm sigmoid tại nơ-ron đầu ra. Nếu activation của nơ-ron đầu vào gần 0, hoặc nếu nơ-ron đầu ra bị bão hòa (activation gần 0 hoặc 1), thì trọng số sẽ học chậm.