IL TEOREMA DI APPROSSIMAZIONE UNIVERSALE

Il **teorema di Cybenko** o **teorema di approssimazione universale** è un risultato importante nel campo delle reti neurali, in particolare per le reti neurali feed-forward a strato singolo, chiamate anche perceptron a strato singolo.

> **Il teorema afferma che una rete neurale a strato singolo con una funzione di attivazione sigmoide (o qualsiasi funzione di attivazione continua e non costante) può approssimare qualsiasi funzione continua su un intervallo chiuso e limitato, purché abbia un numero sufficiente di neuroni nel suo strato nascosto.**

In altre parole, se abbiamo una funzione continua definita su un intervallo, **possiamo trovare una rete neurale con un solo strato nascosto che, con abbastanza neuroni, può rappresentare questa funzione** con qualsiasi grado di precisione desiderato.

Il teorema di Cybenko è fondamentale perché **fornisce una base teorica per l'uso delle reti neurali nell'apprendimento automatico.** Prima di questo teorema, non era chiaro se le reti neurali potessero effettivamente apprendere e rappresentare una vasta gamma di funzioni. **Il teorema dimostra che, in teoria, le reti neurali semplici sono potenti strumenti di approssimazione, il che ha spianato la strada al loro sviluppo e utilizzo in molti campi.**

Il teorema di Cybenko può essere visto come un caso particolare del più generale teorema di Stone-Weierstrass, che riguarda l'approssimazione di funzioni continue. Questo teorema afferma che ***ogni funzione continua definita su un intervallo chiuso e limitato può essere approssimata arbitrariamente bene da una combinazione lineare di funzioni più semplici*** (polinomi o altre funzioni elementari).

Il teorema di Cybenko utilizza l'idea di base del teorema di Stone-Weierstrass, ma la specifica per le reti neurali. Invece di usare polinomi o altre funzioni elementari, il teorema di Cybenko dimostra che le combinazioni lineari di funzioni di attivazione sigmoidi (o altre funzioni di attivazione non costanti e continue) possono approssimare qualsiasi funzione continua. In altre parole, **le unità sigmoidi nel teorema di Cybenko giocano un ruolo simile ai polinomi nel teorema di Stone-Weierstrass.**

Una rete neurale è costituita da **strati di neuroni**. I neuroni sono organizzati in strati: uno strato di input, uno o più strati nascosti e uno strato di output. **Ogni neurone in un dato strato è collegato ai neuroni del successivo strato tramite dei pesi.** Ogni neurone elabora una **combinazione lineare** degli input (pesi moltiplicati per gli input e sommati a un bias) e passa il risultato attraverso una **funzione di attivazione non lineare (come la sigmoide, ReLU, tanh, etc.**). La non linearità è cruciale perché permette alla rete di approssimare funzioni non lineari.

Durante il processo di addestramento, i pesi e i bias della rete vengono aggiornati per minimizzare l'errore tra l'output della rete e i valori target. Questo viene fatto utilizzando un algoritmo di ottimizzazione come la discesa del gradiente, che si basa sul calcolo del gradiente dell'errore rispetto ai pesi (backpropagation).

Grazie alla combinazione di pesi ottimizzati e funzioni di attivazione non lineari, una rete neurale può modellare relazioni complesse tra gli input e gli output. La rete apprende a generare una mappa dagli input agli output che si avvicina alla funzione target.

Il teorema di Cybenko si applica a funzioni continue su spazi di dimensione arbitraria, quindi se la rete ha **n** input, la rete può approssimare funzioni continue **f:Rn→R**

In questo contesto, la rete neurale può essere vista come una mappa che trasforma un vettore di input **x**=**(x1,x2,x3,…,xn)** in un output y che è una funzione delle variabili di input. Questa funzione approssimata può essere espressa come:

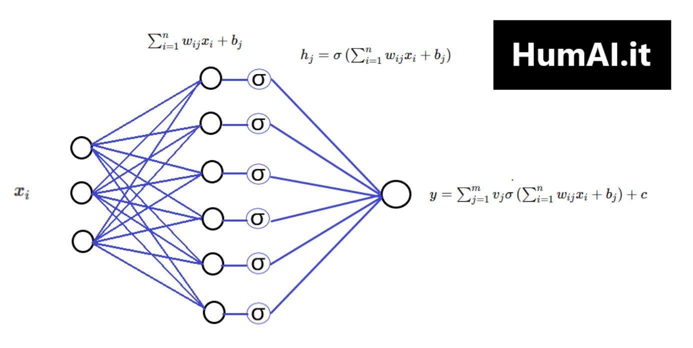
y≈f(x1,x2,x3,…,xn)

Dove f è una funzione continua di n variabili.

Per illustrare, supponiamo di avere una rete neurale feed-forward con tre neuroni di input (x1,x2,x3), un singolo strato nascosto con neuroni che utilizzano la funzione di attivazione sigmoide, e un neurone di output. La rete può essere addestrata per approssimare una funzione continua che mappa i tre input ad un output:

y=f(x1,x2,x3)

Il teorema di Cybenko garantisce che, con un numero sufficiente di neuroni nel **livello nascosto**, la rete può approssimare f arbitrariamente bene. (cioè non esattamente)



y è l'output della rete.

v_j è il peso della connessione tra il neurone nascosto j e il neurone di output.

h_j è l'output del neurone nascosto j (già calcolato come sopra).

c è il bias del neurone di output.

Formule per i Neuroni del Livello Nascosto

Per ogni neurone j nel livello nascosto, il calcolo è il seguente:

$$h_j = \sigma\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right)$$

La funzione sigmoidea è una funzione matematica da R ad R crescente che produce una curva sigmoide, ovvero una curva avente un andamento ad "S". Questa funzione mappa qualsiasi valore reale in un intervallo compreso tra 0 e 1. Nelle reti neurali, la funzione sigmoide introduce non linearità nel modello, permettendo di apprendere relazioni complesse nei dati.

Dove:

- h_j è l'output del neurone nascosto j.
- $\sigma(z) = \frac{1}{1+e^{-z}}$ è la funzione di attivazione sigmoide.
- w_{ij} è il peso della connessione tra l'input x_i e il neurone nascosto j.
- b_j è il bias del neurone nascosto j.

Formula per l'Output della Rete

L'output finale della rete è una combinazione lineare delle uscite dei neuroni del livello nascosto:

$$y = \sum_{j=1}^{m} v_j h_j + c$$

Dove:

Dove:

- y è l'output della rete.
- v_j è il peso della connessione tra il neurone nascosto j e il neurone di output.
- h_j è l'output del neurone nascosto j (già calcolato come sopra).
- c è il bias del neurone di output.

Insieme, le Formule Sono

Mettiamo tutto insieme per vedere come si arriva all'output finale partendo dagli input:

1. Calcolo dell'output di ogni neurone nascosto:
$h_j = \sigma\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right), \quad \text{per } j = 1, 2, \ldots, m$
2. Calcolo dell'output finale della rete:
$y = \sum_{j=1}^{m} v_j\sigma\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right) + c$

Questa sommatoria di funzioni sigmoidi permette alla rete di approssimare una vasta gamma di funzioni continue, grazie alla capacità di combinare in modo non lineare gli input e i pesi della rete.

Approfondimento matematico

Il teorema afferma che per ogni funzione continua f(x) definita su [0,1]^n e per ogni valore positivo di epsilon, esiste una rete neurale con un numero finito di neuroni capace di approssimare f(x) con errore inferiore ad epsilon. (L'epsilon viene introdotto nel Teorema per esprimere il grado di accuratezza con cui la rete neurale approssima la funzione continua)

Enunciato del Teorema di Approssimazione Universale di Cybenko:

Sia $\sigma : \mathbb{R} \to \mathbb{R}$ una funzione di attivazione sigmoide, cioè una funzione continua e limitata tale che $\lim_{x\to-\infty}\sigma(x) = 0$ e $\lim_{x\to+\infty}\sigma(x) = 1$. Allora, per ogni funzione continua $f \in C([0,1]^n)$ e per ogni $\epsilon > 0$, esiste una rete neurale feedforward con un singolo strato nascosto con un numero finito di neuroni, coefficienti a_i, b_i e pesi α_i tali che:
$\left|f(x) - \sum_{i=1}^{N} \alpha_i\sigma(a_i \cdot x + b_i)\right| < \epsilon$
per ogni $x \in [0,1]^n$.

Enunciato del Teorema di Stone-Weierstrass

Sia K uno spazio compatto e A un'~~algebra~~ insieme di funzioni reali continue su K. Se A soddisfa le seguenti condizioni:

1. A contiene la funzione costante 1. esiste almeno una funzione in A che è sempre uguale a 1
2. A separa i punti di K, ossia per ogni coppia di punti distinti $x, y \in K$, esiste $f \in A$ tale che $f(x) \neq f(y)$. Per ogni coppia di punti distinti (x,y), esiste una funzione in A che assume valori diversi in quei due punti.

Allora, A è densa in $C(K)$, lo spazio delle funzioni continue su K, rispetto alla norma sup.
cioè ogni funzione continua su K può essere approssimata arbitrariamente bene da funzioni in A.

> La dimostrazione del teorema di Cybenko si basa sulla tesi che l'insieme delle funzioni sigmoidi forma un'algebra di funzioni che soddisfa le condizioni del teorema di Stone-Weierstrass.

Premesse

Norma Sup

La norma sup di una funzione f definita su un intervallo I è data da:
$\|f\|_\infty = \sup_{x\in I}|f(x)|$

Chiusura di A

La chiusura di A rispetto alla norma sup è l'insieme di tutte le funzioni $f \in C(I)$ tali che per ogni $\epsilon > 0$ esiste una funzione $g \in A$ per cui:
$\|f - g\|_\infty < \epsilon$

In altre parole, una funzione f appartiene alla chiusura di A se e solo se possiamo trovare una sequenza di funzioni $\{f_n\} \subset A$ che converge uniformemente a f. Formalmente, possiamo scrivere:
$\overline{A} = \{f \in C(I) \mid \forall\epsilon > 0, \exists g \in A \text{ tale che } \|f - g\|_\infty < \epsilon\}$

Esempio

Supponiamo che A sia l'insieme delle polinomi su $[a,b]$. La chiusura di A rispetto alla norma sup è l'insieme di tutte le funzioni continue su $[a,b]$, cioè:
$\overline{A} = C([a,b])$

Questo segue dal teorema di Weierstrass, che afferma che ogni funzione continua su un intervallo chiuso e limitato può essere approssimata uniformemente da polinomi.

Importanza della Chiusura

La nozione di chiusura è fondamentale perché consente di estendere le proprietà di un insieme A a tutte le funzioni che possono essere approssimate dalle funzioni in A. Nel contesto del teorema di Stone-Weierstrass, dimostrare che A è denso in $C(I)$ significa che $\overline{A} = C(I)$, cioè ogni funzione continua può essere approssimata da funzioni in A.

Dimostrazione del Teorema di Stone-Weierstrass

Passo 1: Preliminari e definizioni

- $C(K)$ è uno spazio normato con la norma sup $\|f\| = \sup_{x\in K}|f(x)|$.
- A è un'algebra di funzioni continue che separa i punti e contiene la funzione costante 1.

Passo 2: Densità nell'approssimazione di funzioni reali

Vogliamo dimostrare che per ogni funzione continua $f \in C(K)$ e per ogni $\epsilon > 0$, esiste una funzione $g \in A$ tale che $\|f - g\| < \epsilon$.

Passo 3: Approssimazione delle funzioni continue

Utilizziamo il fatto che A separa i punti di K per costruire un'approssimazione. Consideriamo $f \in C(K)$ e sia $\epsilon > 0$.

Per ogni punto $x \in K$, esiste un intorno aperto U_x di x tale che per ogni $y \in U_x$, $|f(y) - f(x)| < \frac{\epsilon}{2}$. Poiché K è compatto, esiste un numero finito di tali intorni $U_{x_1}, U_{x_2}, \ldots, U_{x_n}$ che coprono K.

Passo 4: Funzioni di partizione dell'unità

Costruiamo una partizione dell'unità subordinata a questa copertura. Poiché A separa i punti, possiamo costruire funzioni $\varphi_i \in A$ tali che:

- $\varphi_i \geq 0$.
- $\varphi_i(x_j) = \delta_{ij}$, dove δ_{ij} è il delta di Kronecker.

Passo 5: Combinazione lineare di funzioni

Definiamo la funzione $g \in A$ come:
$g = \sum_{i=1}^{n} f(x_i)\varphi_i$.

Questa funzione g è una combinazione lineare di funzioni in A e quindi appartiene ad A.

Passo 6: Stima dell'errore

Per ogni $x \in K$, $x \in U_{x_j}$ per qualche j, quindi:
$|f(x) - g(x)| = \left|f(x) - \sum_{i=1}^{n} f(x_i)\varphi_i(x)\right|$.

Poiché $\varphi_i(x)$ sono funzioni di partizione dell'unità, $\sum_{i=1}^{n}\varphi_i(x) = 1$ e $\varphi_i(x) \geq 0$. Quindi:
$|f(x) - g(x)| = \left|f(x) - \sum_{i=1}^{n} f(x_i)\varphi_i(x)\right| \leq \sum_{i=1}^{n}|f(x) - f(x_i)|\varphi_i(x)$.

Dato che $x \in U_{x_j}$, $|f(x) - f(x_j)| < \frac{\epsilon}{2}$, quindi:
$|f(x) - g(x)| < \frac{\epsilon}{2}\sum_{i=1}^{n}\varphi_i(x) = \frac{\epsilon}{2} \cdot 1 = \frac{\epsilon}{2}$.

Pertanto:
$\|f - g\| = \sup_{x\in K}|f(x) - g(x)| < \epsilon$.

Conclusione

Abbiamo dimostrato che per ogni $f \in C(K)$ e ogni $\epsilon > 0$, esiste $g \in A$ tale che $\|f - g\| < \epsilon$. Quindi, A è densa in $C(K)$, completando così la dimostrazione del teorema di Stone-Weierstrass.

Proprietà delle Funzioni φ_i

1. **Non negatività**: $\varphi_i(x) \geq 0$ per ogni i e x.
2. **Partizione dell'unità**: $\sum_{i=1}^{n}\varphi_i(x) = 1$ per ogni x.

Consideriamo il termine $|f(x) - g(x)|$:

$|f(x) - g(x)| = \left|f(x) - \sum_{i=1}^{n} f(x_i)\varphi_i(x)\right|$.

Applichiamo l'identità triangolare inversa:

$\left|f(x) - \sum_{i=1}^{n} f(x_i)\varphi_i(x)\right| = \left|\sum_{i=1}^{n}(f(x) - f(x_i))\varphi_i(x)\right|$.

Usiamo la disuguaglianza triangolare per sommare i termini all'interno del valore assoluto:

$\left|\sum_{i=1}^{n}(f(x) - f(x_i))\varphi_i(x)\right| \leq \sum_{i=1}^{n}|f(x) - f(x_i)|\,|\varphi_i(x)|$.

Poiché $\varphi_i(x) \geq 0$, possiamo scrivere $|\varphi_i(x)| = \varphi_i(x)$. Quindi, otteniamo:

$\left|\sum_{i=1}^{n}(f(x) - f(x_i))\varphi_i(x)\right| \leq \sum_{i=1}^{n}|f(x) - f(x_i)|\,\varphi_i(x)$.

Questa è l'ineguaglianza che volevamo dimostrare:

$\left|f(x) - \sum_{i=1}^{n} f(x_i)\varphi_i(x)\right| \leq \sum_{i=1}^{n}|f(x) - f(x_i)|\,\varphi_i(x)$.

Dimostrazione del teorema di Cybenko

1. **Definizione delle Funzioni Sigmoidi:**
Una funzione sigmoide $\sigma : \mathbb{R} \to \mathbb{R}$ è una funzione continua che soddisfa:
   - $\lim_{x\to-\infty}\sigma(x) = 0$
   - $\lim_{x\to\infty}\sigma(x) = 1$

   Un esempio comune di funzione sigmoide è la funzione logistica: $\sigma(x) = \frac{1}{1+e^{-x}}$.
2. **Spazio delle Funzioni Continue:**
Consideriamo lo spazio $C([a,b])$ delle funzioni continue definite su un intervallo chiuso e limitato $[a,b]$.
3. **Algebra delle Combinazioni Lineari:**
Consideriamo l'algebra $\mathcal{A}$ generata dalle combinazioni lineari finite di funzioni sigmoidi della forma $\sigma(w \cdot x + \theta)$, dove w e θ sono parametri reali e $x \in [a,b]$.
4. **Condizioni del Teorema di Stone-Weierstrass:**
Per applicare il teorema di Stone-Weierstrass, dobbiamo verificare che $\mathcal{A}$ soddisfa le seguenti condizioni:
   - **Algebra**: $\mathcal{A}$ è un'algebra di funzioni reali continue su $[a,b]$.
   - **Separazione dei Punti**: $\mathcal{A}$ separa i punti di $[a,b]$.
   - **Approssimazione delle Costanti**: $\mathcal{A}$ contiene una funzione che approssima arbitrariamente bene la funzione costante.
5. **Separazione dei Punti:**
Dobbiamo mostrare che, per ogni coppia di punti $x_1, x_2 \in [a,b]$ con $x_1 \neq x_2$, esiste una funzione $\sigma(w \cdot x + \theta) \in \mathcal{A}$ tale che $\sigma(w \cdot x_1 + \theta) \neq \sigma(w \cdot x_2 + \theta)$.

   Consideriamo la funzione sigmoide $\sigma(x)$ e i punti x_1 e x_2. Poiché $x_1 \neq x_2$, possiamo scegliere w e θ tali che $w \cdot x_1 + \theta \neq w \cdot x_2 + \theta$. Dato che la funzione sigmoide è strettamente monotona, $\sigma(w \cdot x_1 + \theta) \neq \sigma(w \cdot x_2 + \theta)$.
6. **Approssimazione delle Costanti:**
Consideriamo la funzione costante $c \in \mathbb{R}$. Possiamo approssimare c utilizzando una combinazione lineare di funzioni sigmoidi. Ad esempio, per una funzione sigmoide logistica $\sigma(x)$, possiamo considerare $c = \sigma(0)$. Con opportuni valori di w e θ, possiamo ottenere una combinazione lineare di funzioni sigmoidi che approssima c arbitrariamente bene.
7. **Applicazione del Teorema di Stone-Weierstrass:**
Dato che $\mathcal{A}$ è un'algebra che separa i punti di $[a,b]$ e approssima le costanti, possiamo applicare il teorema di Stone-Weierstrass. Questo teorema ci dice che $\mathcal{A}$ è densa in $C([a,b])$. Quindi, per ogni funzione continua $f \in C([a,b])$ e per ogni $\epsilon > 0$, esiste una combinazione lineare di funzioni sigmoidi che approssima f con un errore massimo inferiore a ϵ.

Una rete neurale profonda è costruita con più livelli di trasformazione successivi, ognuno dei quali prende in input l'output dello strato precedente (si prendono gli output dello strato nascosto precedente come se fossero input di una rete con un solo strato nascosto)

Perchè in generale le reti neurali sono costituite da più livelli nascosti?

L'idea che un singolo strato nascosto sia sufficiente per approssimare qualsiasi funzione continua su un intervallo chiuso e limitato, come abbiamo visto, deriva dal teorema di approssimazione universale. Tuttavia, nella pratica dell'apprendimento automatico e delle reti neurali, ci sono diverse ragioni per cui si preferisce utilizzare reti con più strati nascosti, anche noti come reti neurali profonde (deep neural networks).

1 Le reti profonde possono sfruttare la capacità di costruire funzioni complesse componendo funzioni più semplici. Questo permette di rappresentare in modo più efficiente relazioni complesse rispetto a un singolo strato nascosto. 2 **Reti con più strati possono apprendere gerarchie di caratteristiche**. Strati più bassi possono catturare caratteristiche di basso livello (ad esempio, bordi nelle immagini), mentre strati più alti possono catturare caratteristiche di alto livello (ad esempio, forme o oggetti interi).

3 Strati multipli con un numero adeguato di neuroni possono contribuire a una migliore generalizzazione rispetto a un singolo strato con molti neuroni, riducendo il rischio di sovradattamento ai dati di addestramento (overfitting).

4 Le reti profonde possono rappresentare funzioni che sono esponenzialmente più complesse rispetto a reti con un singolo strato nascosto. Questo significa che alcune funzioni possono essere rappresentate con meno neuroni complessivi in una rete profonda rispetto a una rete con un solo strato nascosto.

Anche se un singolo strato nascosto può approssimare una qualsiasi funzione, la quantità di neuroni necessari potrebbe essere impraticabile. **Reti profonde possono approssimare le stesse funzioni con meno risorse computazionali.**

In compiti di visione artificiale, reti profonde come le **CNN (Convolutional Neural Networks)** hanno dimostrato di essere estremamente efficaci nel riconoscimento di immagini e nella rilevazione di oggetti.

Nella **NLP (Natural Language Processing)**, reti come le **RNN (Recurrent Neural Networks) e le trasformatori (Transformers)** con più strati sono utilizzate per catturare dipendenze temporali e strutture linguistiche complesse.

Lezioni di Python

Python: un'ottima scelta per gli algoritmi di intelligenza artificiale

Integrazione di SQL in Python e l'approccio dinamico

Cerca

Addestramento dei modelli di traduzione automatica

Aiuta il progetto HumAI a crescere

Algoritmi di disambiguazione (Word Sense Disambiguation)

Architettura del sistema encoder-decoder con reti RNN

Capire l'intelligenza artificiale

Chiarezza sulle reti neurali

Come l'AI corregge i propri errori: Gradient descent e backpropagation

Come si addestra un algoritmo generativo del linguaggio?

Cross entropy: la funzione che permette ai modelli di AI di migliorarsi.

Fare previsioni con i modelli di Markov

Glossario

HumAI: ChatGPT sul computer

Il processo di information retrieval (IR)

Il significato delle parole con i modelli NLP

Il teorema di approssimazione universale

Immagini di fantasia con DALL-E3

Interpretabilità delle reti neurali

L'AI e l'analisi della mente umana

L'albero delle decisioni e il machine learning

La computazione quantistica

La regolarizzazione, un modo per gestire l'overfitting

La self-attention delle reti transformer

La sintesi vocale con le reti neurali ricorrenti

Le reti neurali adatte ad elaborare sequenze di dati

Le scienze cognitive e l'intelligenza artificiale

Lessico neurale

Modelli di AI più efficienti

Principal Component Analysis (PCA): ridurre la dimensioni salvando l'informazione

Quando l'AI "comprende" quello che dici

Semplifichiamo

Trading finanziario con l'intelligenza artificiale

Tutti gli approfondimenti

Un compromesso tra coerenza e creatività dei modelli LLM

Word embedding e il modello Skip-gram

Ultimi articoli

L'intelligenza artificiale può comprendere?

Librerie python basate su LLM

Modelli addestrati per generare critiche efficaci: CriticGPT

I "Fatti" sono una questione di statistica?

Chiarezza sui concetti di fine-tuning, transfer learning e prompt engineering

Argomenti principali

Conversazioni con l'AI

L'intelligenza artificiale