Carlo Niccolai (carlopavana)

Costruire una rete neurale da zero

17 July 2025

Questo materiale è pensato per tutti soprattutto per le persone interessate all'apprendimento automatico ma che non hanno mai programmato, tutti oggi potranno imparare a costruire una rete neurale.

In altre parole questo documento dovrebbe farci costruire una rete neurale partendo da zero, ma se non dovessimo riuscirci perché troppo complicato, sicuramente miglioreremo le nostre conoscenze in queste tre aree:

- Maggiore familiarità con una non facile matematica.
- Capire come funziona l'apprendimento profondo su un livello profondo (scusate il gioco di parole).
- Comprendere come rendere il codice di programmazione più efficiente usando la matematica vettoriale.

Il nostro futuro?

Il nostro futuro?

Io vorrei che questo documento fosse utilizzabile da tutti, anche principianti totali ed ho cercato di prepararlo con questo obiettivo, unico prerequisito necessario è quello di essere in grado di risolvere questa equazione, io non posso spiegare tutta l'algebra.

```
8 + x5 = 18
```

In ogni caso il lettore deve sentirsi libero di saltare qualsiasi sezione elencata di seguito se ha già qualche conoscenza del machine learning (ML). I titoli segnati da (*) sono fondamentali.

Costruire una rete neurale

1)*L'apprendimento automatico.

2)*Corso intensivo sulle matrici.

3)*Corso intensivo sulle derivate.

4)*Corso intensivo sulle derivate parziali.

5) Il modello del perceptron

6) Notazione della rete neurale

7) Propagazione in avanti (Feed Forward)

8) Vettorializzare, Vettorializzare, Vettorializzare

Seconda parte – programmazione della rete (non in questo articolo)

9) Il codice

10) Il costo

11) La retropropagazione

12) Costruiamo questa rete

Quando mi sembrerà opportuno indicherò delle risorse esterne per approfondire un determinato argomento dato che quello che sto presentando sono solo gli elementi essenziali; ho escluso tutto quello che, sebbene utile, non è importante per costruire una solida conoscenza sulle reti neurali.

Questo materiale deve rafforzare questo principio: i concetti e la matematica sono importanti il codice di programmazione non lo è.

Indice

- 1 1)*L'apprendimento automatico.
- 2 2)*Corso intensivo sulle matrici.
- 3 3)*Corso intensivo sulle derivate.
- 4 4)*Corso intensivo sulle derivate parziali.
- 5 5) Il modello del perceptron
- 6 6) Notazione della rete neurale
- 7 7) Propagazione in avanti (Feed Forward)
- 8 8) Vettorializzare, Vettorializzare, Vettorializzare

1)*L'apprendimento automatico.

Vi siete mai chiesti perché ChatGPT sembra che sia in grado di capirvi, o come faccia gemini pro vision a riconoscere un gatto da una panchina quando gli vengono fornite

delle immagini da esaminare?
Per quanto le macchine sembrino umane, e per quanto sembri impossibile che sia finto un tale tipo di conoscenza, è proprio questo che fanno le macchine: fingono e provano fino a quando non riescono a superare la prova.
Facciamo una simulazione mentale.
Ripensiamo a quando eravamo piccoli e non sapevamo nulla, ma poi abbiamo imparato a riconoscere le persone, gli animali, gli alberi, e tutti gli altri oggetti che ci circondavano, ma come abbiamo fatto?
Se siamo nati in una famiglia di persone bionde e le uniche persone con cui interagivamo erano i nostri genitori pensavamo sicuramente che tutte le persone fossero bionde finché non abbiamo visto altri esseri umani, avevano le stesse caratteristiche dei nostri genitori, ma i capelli erano neri, rossi o bianchi.
Non assomigliavano al nostro cane o al nostro gatto e noi dovevamo espandere la nostra definizione di umano per incapsulare sempre più persone: basse, alte, magre, grasse con o senza gambe,
Avevamo ampliato le nostre conoscenze perché avevamo ottenuto più dati, ed è esattamente così che impara anche una macchina.
Se forniamo ad un computer un'immagine di Mario Rossi e lo classifichiamo come un essere umano, il computer penserà che Mario Rossi sia l'unico umano che esiste al mondo. Non riuscirà a classificare qualsiasi altra persona come essere umano.
Ma se diamo alla macchina un'immagine di 100.000 esseri umani, dicendo ogni volta "questo è un essere umano" il computer costruirà una definizione più ampia per come appare un essere umano: viso, braccia, circa 160/180 centimetri di altezza, vestiti, colore della pelle diverso, ecc.
Le macchine imparano dai dati, proprio come fanno gli umani. Più dati ha una macchina, più la sua "visione del mondo" e la conoscenza si espanderà.
Ma anche la la qualità dei dati è importante. Se diciamo ad un bambino che ogni banana che vede è in realtà una mela, ogni volta che vedrà una banana nella sua mente crederà davvero che il lungo frutto giallo sia chiamato mela.
Ma poi troverà a scuola un compagno di classe che gli dirà: "amico questo frutto è una banana". Corretto abbastanza volte il nostro sfortunato bambino imparerà a riconoscere le banane e le mele.
Ma esaminiamo cos'è successo: Il bambino ha imparato dai suoi errori e li ha corretti.
Ancora una volta, esattamente come succede ad una macchina.
Possiamo così dire che l'apprendimento automatico consiste di questi passaggi:

- Raccogliere correttamente i dati definiti per la macchina da allenare.
- Creare una misura per descrivere il valore dell'errore fatto dalla macchina quando cerca di prevedere cosa sia un oggetto.
- Allenamento in modo iterativo per ridurre la misura dell'errore.
  Quando si utilizza per la prima volta una macchina per l'apprendimento automatico, i risultati che si ottengono sono solo spazzatura.

Allora noi "puniamo" il modello nello stesso modo in cui "puniamo" un bambino dicendogli che si sbaglia, ma per le macchine, facciamo un ulteriore passo avanti:

diciamo loro di quanto si sbagliano, con un valore che chiamiamo il costo del modello.
Supponiamo di allenare un modello per classificare cani e gatti, e lo interroghiamo dandogli tre immagini di cani, il modello invece li classifica tutti come gatti.
Quindi, quanti ne ha indovinati?
Ha un punteggio di 0/3, quindi ovviamente è andato piuttosto male. Ma 1/3 è un punteggio migliore di 0/3, e 2/3 ancora di più.
Possiamo quantificare il tasso di errore attraverso il costo, con un'implementazione di base come segue:
Costo = numero di errori / numero di previsioni totali
Quindi, nel caso del nostro modello super spazzatura, su 3 previsioni ha fatto 3 ipotesi sbagliate, quindi il costo = 3/3 = 1, che è il costo più elevato possibile per il modello.
Tutto il machine learning consiste nel cercare di minimizzare il più possibile il valore del costo.
Un costo pari a 0 significa che otteniamo 0 previsioni sbagliate su un totale di 3 previsioni (0/3), il che significa che il nostro modello ha indovinato tutto.
Basta capire questo:

- basso costo = buon modello
- alto costo = cattivo modello.
  Puoi pensare al costo come sinonimo di un voto scolastico: se ottieni un 3 nella tua classe significa che non conosci assolutamente la materia, ma se ottieni un 9 stai andando alla grande.
  Ma come possiamo minimizzare questo costo?
  Qui entra in gioco una matematica abbastanza complicata e presto la impareremo.
  In questo momento disegniamo un’analogia per le reti neurali.
  Come suggerisce il nome, le reti neurali stanno cercando di copiare ciò che il cervello umano fa nella speranza di creare l’intelligenza artificiale alla pari di quella dell’umanità.
  Si stima che il cervello umano abbia oltre 100 miliardi di neuroni, quindi un modello di macchina del cervello umano (una rete neurale) dovrebbe avere 100 miliardi di strumenti di calcolo, che per semplicità chiameremo parametri.
  Se il costo è una misura di quanto male sta andando un modello, allora gli serve prima l’output del modello, che richiede 100 miliardi di parametri per calcolare il risultato finale. Questo rende la funzione di costo un vero e proprio colosso: adopera 100 miliardi di variabili e restituisce un solo numero.
- Si può anche solo capire, per non dire minimizzare, una funzione di tali dimensioni?
- Si può minimizzare il costo manualmente?

Ecco le risposte alle due domande, rispettivamente:

- Sì, si può.
- No, non si può, ed è per questo che usiamo i computer.

Riepilogo dell'apprendimento automatico. Ora sappiamo cos'è l'apprendimento automatico.
Quando cerchiamo di far sì che un modello di apprendimento automatico classifichi elementi come immagini o dati, gli indicheremo quali sono le risposte corrette e lo proveremo sui dati.
Da questi test otteniamo la misura del costo che indica quanto il modello faccia previsioni errate sui dati, ed eseguiremo un processo iterativo che preveda calcoli matematici complessi per minimizzare tale costo.
Quando il costo sarà ridotto al minimo significa che il modello prevede correttamente le cose quasi come, o addirittura meglio, di un essere umano.

2)*Corso intensivo sulle matrici.

Le matrici sono un modo conciso per rappresentare sistemi di equazioni.
Quando si devono elaborare centinaia o migliaia di numeri, come spesso accade nell'apprendimento automatico, le matrici sono la chiave per rendere tutto comprensibile.
Ad esempio come si potrebbe rappresentare in modo conciso

```
(6 * 5) + (2 * 1) + (6 * 3)
```

In questo modo:

```
(6,2,6) . (5,1,3)
```

Prodotto a punti: prodotto scalare tra due vettori
Qui stiamo moltiplicando due vettori (lo spiegherò più avanti) in un’operazione chiamata prodotto a punti, che lo calcoliamo con questi passaggi:

- Il primo elemento del primo vettore si moltiplica per il primo elemento del secondo vettore.
- Il secondo elemento del primo vettore si moltiplica per il secondo elemento del secondo vettore
- Il terzo elemento del primo vettore si moltiplica per il terzo elemento del secondo vettore.
- Poi si sommano questi tre numeri e si ottiene il prodotto scalare.

Si accoppia ogni elemento da entrambi i vettori in ordine, si moltiplicano e poi si sommano i numeri risultanti, quindi:

- La prima coppia: 6*5
- Seconda coppia: 2*1
- Terza coppia: 6*3

Sommando il tutto, si ottiene 30 + 2 + 18 = 50: quando si effettua il prodotto di due vettori si ottiene una somma, un singolo numero, che chiamiamo scalare.
Tenete a mente che non è possibile calcolare due vettori di prodotto se hanno lunghezze diverse.
Cosa succede se un vettore ha 2 elementi e il secondo vettore ha tre elementi? Quel terzo elemento nel secondo vettore non ha altro elemento a cui accoppiarsi quindi potrebbe far bruciare la calcolatrice? NO, scherzo! per fortuna la cosa non è così drammatica.
I vettori sono solo matrici unidimensionali, il che significa che agiscono proprio come un elenco di numeri.
Per ottenere una più profonda comprensione dei vettori è possibile seguire su internet moltissimi tutorial in ogni lingua della terra, questa ulteriore conoscenza non è necessaria per costruire una rete, ma può essere estremamente utile.

Le matrici sono più complicate.
Sono bidimensionali e possono avere molte righe e molte colonne. Ecco un esempio di matrice:

Una matrice con 2 righe e 2 colonne

Una matrice con 2 righe e 2 colonne

Rivediamo la terminologia:

- Quando si pensa a uno scalare, si pensi ad un singolo numero.
- Quando si pensa a un vettore, si pensa ad un elenco di numeri.
- Quando si pensa a una matrice, si pensa ad una tabella di numeri.

Perché ci preoccupiamo così tanto delle matrici?
Perché la moltiplicazione di matrici può trasformare molti calcoli in espressioni semplici.
Ad esempio, cosa succede se voglio moltiplicare molti numeri tra loro, ma non conosco una somma finale?
Voglio una nuova matrice come risultato?

Un esempio di moltiplicazione di due matrici.

Un esempio di moltiplicazione di due matrici.

Un esempio di moltiplicazione di due matrici.
La moltiplicazione della matrice funziona come segue:

1. Nella prima matrice, prendi la prima riga della prima matrice (nota che è un vettore perché è monodimensionale, solo una lista di numeri) e la prima colonna della seconda matrice (anch'essa un vettore!) e calcola il prodotto scalare tra loro. Questo è il primo elemento della matrice risultante. Questo è il calcolo (6*8 + 3*3).

2. Nella prima matrice, prendi la prima riga della prima matrice e la seconda colonna della seconda matrice e calcola il prodotto scalare tra loro. Questo è il secondo elemento della matrice risultante. Questo è il calcolo (6*4 + 3*5).
3. Nella prima matrice, prendi la seconda riga della prima matrice e la prima colonna della seconda matrice e calcola il prodotto scalare tra loro. Questo è il terzo elemento della matrice risultante. Questo è il calcolo (4*8 + 7*3).
4. Infine prendi la seconda riga della prima matrice e la seconda colonna della seconda matrice e calcola il prodotto scalare tra loro. Questo è il quarto elemento della matrice risultante. Questo è il calcolo (4*4 + 7*5).
La moltiplicazione di matrici è stancante e, ad essere onesti, anche un po' confusa.
La buona notizia è che non è necessario capire come si fa la moltiplicazione di matrici, ma solo quando è possibile farla.
Il fattore più importante nella moltiplicazione di matrici è comprendere le dimensioni di ciascuna matrice e come apparirà la matrice risultante.
Se non impari nient'altro da questo capitolo, almeno impara bene la prossima sezione:
La dimensionalità della moltiplicazione di matrici.
Se una matrice ha 2 righe e 2 colonne, è una matrice 2 x 2 (leggi la "x" come "per").
Se una matrice ha 3 righe e 4 colonne, è una matrice 3 x 4.
Quindi la formula generale per descrivere la dimensionalità di una matrice è sempre le righe x colonne. Questa notazione è importante per capire quale combinazione di matrici possiamo e non possiamo moltiplicare.
Due matrici possono essere moltiplicate tra loro se la prima matrice ha dimensioni a x b e la seconda matrice ha dimensioni b x c, e quindi la matrice risultante avrà dimensioni a x c.
Cosa intendo con questo?
Il numero di colonne della prima matrice deve essere esattamente uguale al numero di righe della seconda matrice.
Questa che segue è una combinazione di matrici che non può funzionare, che proprio non si può moltiplicare:

Se proviamo a fare il prodotto a punti tra la prima fila della prima matrice e la prima colonna della seconda matrice, puoi vedere che 8 è da solo. A chi lo abbiniamo e con chi lo moltiplichiamo: con il vuoto? Vedi, non si può fare.
La regola generale che potresti aver afferrato è che le righe nella prima matrice devono avere la stessa dimensione (stesso numero di elementi) delle colonne nella seconda matrice. Non è possibile eseguire il prodotto due vettori se hanno dimensioni diverse.
Ma cosa accadrebbe se invertissimo queste due matrici?

Moltiplicare una matrice 3 x 2 con una matrice 2 x 2 produce una matrice valida 3 x 2.
Ora le dimensioni moltiplicatrici seguono correttamente il modello a x b e b x c,

producendo una matrice a x c risultante.
Guardiamo le dimensioni:

- a: Il numero di righe nella prima matrice, che è 3
- b: Il numero di colonne nella prima matrice, che è 2
- c: Il numero di colonne nella seconda matrice, che 2

Quindi, una matrice 3 × 2 moltiplicata per una matrice 2 × 2 produce una matrice 3 × 2.
Mi piace pensare a questa operazione come una sorta di schiacciamento e fusione dei due numeri centrali, in questo modo:

- 3 x 2 moltiplicato per 2 x 2
- Dimensioni della matrice risultante: 3 x 2

Un ultimo consiglio è che quando si moltiplicano le matrici, l'ordine conta. Ecco un esempio in cui moltiplichiamo due matrici 2 x 2 insieme (quindi le loro dimensioni saranno sempre valide), ma lo facciamo in due ordini diversi e otteniamo risultati molto diversi.

Esempio di come la moltiplicazione delle matrici NON è commutativa
Nel contesto dell'apprendimento automatico, l'ordine in cui moltiplicare le matrici può essere molto confuso, ma come vedrai più avanti, la parte più importante è solo ottenere le dimensioni da abbinare. Questa proprietà non commutativa della moltiplicazione della matrice è importante da tenere a mente, ma non dovrai affrontarla.
Ora che sai come funzionano le dimensioni della matrice, c'è un ultimo passo prima di imparare con successo tutta l'algebra lineare di cui hai bisogno per creare una rete neurale.

La trasposizione
La trasposizione è un concetto semplice si traspone una matrice semplicemente cambiando le sue righe e le sue colonne.
Nell'esempio seguente, trasponiamo una matrice 3 x 2 in una matrice 2 x 3. Il simbolo “T” significa solo che stiamo trasponendo la matrice.

Esempio di trasposizione di una matrice 3 x 2 in una matrice 2 x 3 semplicemente scambiando righe e colonne.
La prima colonna della matrice diventa la prima riga nella nuova matrice trasposta.
La seconda colonna della matrice diventa la seconda riga nella nuova matrice trasposta, e questo modello continua.
Perché è importante il recepimento? Perché ci permette di moltiplicare due matrici insieme che altrimenti non potremmo farlo.

Torniamo all'esempio precedente:

2 x 2 volte una matrice 3 x 2 è impossibile
Ma cosa succede se trasponiamo quella matrice 3 x 2 in una matrice 2 x 3?

La trasposizione ci permette di moltiplicare due matrici insieme senza cambiare l'ordine (che come avete imparato prima, l'ordine conta).
La trasposizione della matrice ci dà anche una grande flessibilità nella moltiplicazione della matrice secondo questa legge:
WTX = XW
chiamata legge della trasposizione, puoi provare a dimostrarla usando questo esempio che riporto qui sotto.

Prova a moltiplicare queste matrici per dimostrare la legge di cui sopra.
Riepilogo

- Un prodotto a punti tra due vettori restituisce un numero singolo.
- Per la moltiplicazione della matrice l' ordine è importante.
- Puoi moltiplicare solo due matrici se la prima matrice ha dimensioni di una a x b, e la seconda ha dimensioni b x c, risultando in una matrice con dimensioni a x c.
- La trasposizione di una matrice scambia le colonne e le righe. Fa una matrice con dimensioni a x b che si trasforma in una matrice con dimensioni b x a.

3)*Corso intensivo sulle derivate.

Il calcolo è la salsa segreta che ci permette di ridurre al minimo la funzione di costo di cui abbiamo parlato prima. Sì, hai letto bene: il costo di un modello di apprendimento automatico è solo una funzione matematica.

Entreremo nei dettagli in seguito, ma capiremo che la funzione di costo è enorme, prendendo migliaia di variabili come input. Per ridurre al minimo i costi, dobbiamo scoprire la sua pendenza in modo da poter capire in quale direzione andare.

Le derivate sono un modo per trovare la pendenza di qualsiasi funzione. Se si guarda il grafico della linea qui sotto, possiamo dire che ha una pendenza di 2.

Ma come possiamo trovare la pendenza di una funzione più complessa come una parabola? È qui che entrano in gioco le derivate.

img31ey.jpg

img31ey.jpg

Ma come si trova la pendenza di una parabola? Dov'è l'ascesa finita?

Cosa è una derivata

Torniamo all'esempio della parabola visto in precedenza: non è una retta, quindi non c'è alcuna salita costante. Potresti pensare che calcolare la pendenza di una parabola sia impossibile, ma facciamo un esercizio di astrazione.

Se ingrandisci abbastanza una sezione della parabola, ad esempio da x = 3.0 a x = 3.001, essa apparirà come una linea retta. A quel punto si può calcolarne la pendenza.

Questo concetto di amplificazione della in linea si applica a tutte le funzioni, se l'ingrandimento è sufficiente per far diventare la sezione esaminata una linea si può ottenere la sua pendenza.

Tuttavia, c'è un problema: la pendenza cambia a seconda di dove ti trovi nel grafico. La sezione esaminata non è statica come per una linea, la pendenza tra x -0,0 -0,01 è diversa da x -0,0 a 1.001.

Questo è ciò che è una derivata: l'equazione algebrica che rappresenta la pendenza del grafo in tutti i punti del grafico.

Per una parabola, la derivata sarebbe un'equazione algebrica che rappresenta la linea y=2x. Si può anche pensare alla derivata come l'equazione algebrica tangente (perpendicolare) al grafico in tutti i punti.

img32ey.jpg

img32ey.jpg

Se ancora non capisci, non preoccuparti. Tutto quello che devi sapere è come calcolare le derivate.

Calcolo delle derivate: regola della potenza.

Come abbiamo trovato la derivata di $y = x^2$?

Bene, usiamo la regola della potenza:

1. Porta giù l'esponente di x^2, cioè il 2, davanti alla x e moltiplicalo per la base, ottenendo $2x^2$

2. Sottrai 1 dall'esponente, ottenendo $2x^1$

3. Sorridi felice con il risultato finale: 2x

Proviamo ora a trovare la derivata di $y = 3x^2$:

1. Porta giù il 2, ottenendo $(3 * 2)x^2 = 6x^2$

2. Sottrai 1 dall'esponente, ottenendo $6x^1 = 6x$

Per una notazione più compatta, sostituiamo y con f(x), perché è meglio pensare in termini di funzioni nel machine learning.

Quindi, se f(x)=4x2 rappresentiamo la derivata di f(x) con la notazione f '(x) detta "f primo di x" ed è f '(x)=8x.

Ok, entriamo nei casi in cui la regola della potenza non è così ovvia.

Come faresti a trovare f '(x) per f(x)=2x? La regola della potenza si applica ancora.

1. L'esponente del termine x è 1, quindi fai scendere l'1, ottenendo (2*1)x1.

2. Sottrai 1 dall'esponente, ottenendo 2x0. Ricorda che qualsiasi numero elevato a zero vale 1, perciò ottieni 2*1=2.

Quindi, se f(x)=2x, la derivata f'(x)=2, che coincide con la costante che moltiplica x. Quindi, per ogni f(x)=cx, dove c è una costante, risulta f'(x)=c. Questo ha un senso intuitivo:

Una linea retta ha la stessa pendenza in ogni suo punto.

Rappresentiamo una linea con una pendenza pari a 2 come y=2x, e cos'altro è la derivata oltre alla pendenza di una funzione? La derivata qui è solo uguale alla pendenza, che è f'(x)=2

E il caso finale a cui pensare è la derivata di una costante. Non possiamo usare la regola della potenza per trovare la derivata di una costante come f(x) =5, ma andrò avanti velocemente alla risposta:

Per tutti i f(x) =c dove c è un numero reale costante, f'(x) =0. Quindi la derivata di qualsiasi costante è 0. Questo ha un senso intuitivo:

Un grafico di y =3, che mostra come una linea costante è solo una linea piatta, che ha una pendenza 0, che significa derivata = 0.

La linea f(x)=c dove c è una costante rappresenta solo una linea piatta. Prendiamo ad esempio f(x)=3. È una linea piatta, che significa nessun aumento e tutto corre, dando una pendenza 0. Quindi ha un senso che la derivata f'(x) sia uguale a 0.

Calcolo delle derivate: termini multipli

Se f(x)=x + 5, come troviamo f'(x)? Beh, la regola è semplice: basta trovare la derivata di ogni singolo termine.

La derivata di x è 1, e la derivata della costante 5 è 0, quindi otteniamo f'(x)=1.

Se f(x)=2x2 — 3x + 8, la derivata è solo 4x-3 + 0 – 4x-3.

Calcolo della derivata: Regola del prodotto

Supponiamo che stiamo moltiplicando due funzioni f(x) e g(x) insieme come h(x) =f(x)g(x) e vogliamo trovare la sua derivata.

Ecco la formula generale:

```
h'(x)=f'(x)g(x) + f(x)g'(x)
```

Prima di correre verso la scogliera più vicina e cercare di saltare giù da essa, ascoltami: lavoriamo con un esempio.

1. f(x) =3x, g(x) =2x2

2. Quindi le derivate sono f'(x) =3, g'(x) = 4x

3. Questo porta a h'(x) = 3(2x2) + 3x(4x) = 6x2 + 12x2 = 18x2

Dimostriamo che questo funziona moltiplicando f(x) e g(x) insieme e poi prendendo la derivata:

1. f(x) * g(x) = 6x3

2. La derivata di 6x3 è 18x2, quindi abbiamo dimostrato!

Se vuoi capire perché funziona e come i matematici lo hanno dimostrato, ci sono molti tutorial, quanto esposto basta per le nostre reti neurali.

Calcolo delle derivate: La sommazione.

Basandosi sulla stessa idea di come la derivata di una somma sia solo la derivata di ogni singola parte, tutte sommate, possiamo fare la stessa cosa con la sommazione.

Come troveremmo f’(x) con questo metodo?

style="margin-bottom: 5px;" />

Potresti semplicemente fare prima la somma e poi prendere la derivata, che ti risulterebbe f(x) =10x, che fa f’(x) =10.

Pensiamo un po’ più intelligentemente.

Poiché la sommazione riassume tutti i termini, possiamo solo usare la regola della somma derivata per prendere la derivata di tutte le singole parti di tali somme, e quindi riassumerle tutte.

Quindi il primo passo è quello di trovare la derivata di x, che è 1. Allora la riassumo e basta.

La derivata di x=1 e poi riassumiamo 1 dieci volte per ottenere una risposta finale = 10.

Calcolo delle derivate: regola della catena

La regola della catena è di gran lunga la parte più essenziale per imparare le derivate. Potresti non capire molto, e per approfondire l'intuizione dovrei spiegarti limiti, rette secanti, ecc., quindi impareremo solo come applicare la regola della catena.

La regola della catena riguarda la ricerca delle derivate quando le funzioni sono composte, come f(g(x)).

Come si calcola la derivata di questa funzione composta? Ecco cosa dice la regola della catena:

Se h(x) = f(g(x)), allora h’(x) = f’(g(x)) * g’(x)

Per prima cosa, procediamo alla vecchia maniera con $f(x) = 4x^2$ e $g(x) = 3x^3$.

1. $f(g(x)) = f(3x^3) = 4(3x^3)^2 = 4(9x?) = 36x?$.

2. La derivata di 36x? è — cerca freneticamente 36 per 6 — 216x?.

Ora procediamo con la regola della catena:

1. f’(g(x)) =f’(3x3) * g’(x) = f’(3x3) * 9x2

2. Per scoprire cos'è f'(3x3), dobbiamo prima trovare f'(x).

3. f'(x) =8x, così f'(3x3) 8*3x3 = 24x3.

4. Quindi f'(3x3) * 9x2 - 24x3 * 9x2 = 216x5.

Ottimo, l'abbiamo dimostrato! Ora è il momento di imparare la notazione che useremo per l'apprendimento automatico.

Possiamo riscrivere la regola della catena come segue:

Illustrazione della regola della catena

La sintassi d/dx significa solo prendere la derivata di un'equazione rispetto a x. È un altro modo di riscrivere f'(x). Ad esempio, d/dx di 2x è solo la derivata di 2x, che è uguale a 2.

Questa notazione è importante perché mostra una catena, qualcosa che non si può fare la notazione f primo.

Scomponiamolo:

- d/dx f(g(x)) significa che stiamo cercando di scoprire f'(g(x)), proprio come la regola della catena.
- df/dg è un modo succinto per scrivere df(x)/dg(x), il che significa che vogliamo scoprire la derivata di f(x) rispetto a g(x), che scriviamo come f'(g(x)).
- dg/dx è il nostro vecchio amico g'(x).

Quindi torniamo indietro e facciamo i calcoli con f(x) 4x2 e g(x) 3x3.

df/dg: f'(g(x)) e f'(x) =8x che è uguale a f'(3x3) = 8(3x3)=24x3

dg/dx: g'(x) =9x2

Quindi moltiplichiamo questi due termini insieme per arrivare alla risposta ancora una volta: 216x5.

Una caratteristica utile di questa notazione è l'illustrazione di una catena, soprattutto se si pensa alla moltiplicazione delle frazioni.

Questo potrebbe essere troppo confuso per pensarci al momento, quindi lo affronteremo di nuovo quando parleremo di derivati parziali.

Calcolando le derivate: derivate speciali

Ci sono alcune derivate speciali che dovresti saper calcolare, dato che verranno utilizzate nei calcoli di backpropagation. Tutte queste usano la regola della catena.

Se f(x) = ln(x), che è il logaritmo naturale di x, allora f'(x) = 1/x * 1 = 1/x. È semplicemente 1 fratto il termine interno, e poi, grazie alla regola della catena, si moltiplica per la derivata del termine interno.

Facciamo un altro esempio, ma tieni a mente la regola della catena: se $f(x) = \ln(2x^2)$, allora $f'(x) = (1/2x^2) * 4x = 2/x$. Abbiamo moltiplicato per 4x perché questa è la derivata di $2x^2$.

Ok, questo è davvero strano: se $f(x) = e^x$, allora $f'(x) = e^x$. È la stessa cosa.

Ma con la regola della catena, se $f(x) = e^{\{2x\}}$, allora $f'(x) = e^{\{2x\}} * 2 = 2e^{\{2x\}}$.
Moltiplichiamo per la derivata di 2x, che è semplicemente 2 grazie alla regola della catena.

4)*Corso intensivo sulle derivate parziali.

Congratulazioni, questo è l'ultimo pezzo di conoscenza matematica fondamentale di cui hai bisogno prima di poter costruire una rete neurale da zero.
Se abbiamo una funzione che vive in tre dimensioni come z = 2x + 3y, come possiamo calcolarne la pendenza? Domanda trabocchetto: non è possibile.
Ma possiamo simularla. Possiamo calcolare la derivata di una funzione multidimensionale rispetto a una variabile alla volta, trattando tutte le altre variabili come costanti.
Questa è chiamata derivata parziale, in cui trattiamo solo una variabile in un'equazione multivariabile come una variabile reale e tutte le altre come costanti.
Se una funzione f accetta due variabili x e y, come f(x, y), possiamo rappresentare la derivata parziale di f rispetto a x come $\partial f/\partial x$ (pronunciato "del f del x").

Come possiamo calcolare $\partial f/\partial x$ se f(x, y) = 2x + 3y?

1. Poiché stiamo cercando di calcolare la derivata parziale di f rispetto a x, trattiamo x come una variabile e y come una costante.
2. Qual è la derivata di 2x trattando x come variabile? È solo 2.
3. Qual è la derivata di 3y trattando y come costante? Beh, 3 volte una costante è ancora una costante, e la derivata di qualsiasi costante è uguale a 0.

4. Arriviamo a $\partial f/\partial x = 2 + 0 = 2$

Proviamo a trovare $\partial f/\partial y$ se $f(x, y) = 2x + 3y$.

1. Tratta y come variabile e x come costante.
2. $\partial f/\partial y = 0 + 3 = 3$.

Comprendere intuitivamente la discesa del gradiente

L'intuizione alla base di derivate parziali è quella di pensare a come una singola variabile in una funzione multidimensionale influenzi l'output di quella funzione. $\delta f/\delta x$ chiede, "se cambio solo x e mantengo tutte le altre variabili costanti, in che modo questo cambiamento in x influisce sull'output della mia funzione f?".
Questa è la domanda più importante nell'apprendimento automatico.

Ma se potessi calcolare la derivata parziale del costo del mio modello di apprendimento automatico rispetto a una singola variabile? Allora potrei modificare quella variabile con l'obiettivo di minimizzare al massimo il costo.
Lo ripeto: il calcolo infinitesimale è il segreto dell'apprendimento automatico. Le derivate parziali ci permettono di scoprire come una qualsiasi variabile influisce sul costo di un modello e di modificarlo di conseguenza per ridurlo o aumentarlo a nostro piacimento.

Se $\delta f/\delta x$ è positivo, come ad esempio $\delta f/\delta x = 2$, allora si traduce in, "se x aumenta di qualche piccolo valore, allora la funzione f aumenta di quel piccolo valore per 2".

Se $\delta f/\delta x$ è negativo, come ad esempio $\delta f/\delta x = -2$, allora si traduce in "se x aumenta di qualche piccolo valore, allora la funzione f diminuisce di quel piccolo valore di 2".

Diciamo che la funzione di costo del modello ha una variabile/parametro chiamato w. Se $\delta C/\delta w$ è positivo, significa che se w aumenta, aumenta anche il costo C. Quindi facciamo qualcosa di intelligente: diminuiamo il valore di w. Se w diminuisce, allora lo farà C.

Ti lascio lavorare per te stesso se dovremmo aumentare o diminuire w per ridurre al minimo il costo se $\delta C/\delta w$ è negativo.

Questo è il cuore del machine learning: la discesa del gradiente. Supponiamo che la nostra funzione di costo C abbia 1000 variabili, denominate w_1, w_2, ecc., il che la rende una funzione a ben 1000 dimensioni. Indichiamo C come C(w_1, w_2, …, w_n)
Il gradiente di C è indicato come ?C, che è semplicemente l'insieme di tutte le derivate parziali di C rispetto a ciascuna delle sue 1000 variabili:

Il gradiente di C, si pronuncia "nabla" C, è solo una raccolta dei suoi 1000 derivati parziali poiché ha 1000 variabili come ingresso.

Sembra familiare? È solo un vettore. La
derivata parziale del costo rispetto alla sua prima variabile, δC/δw_1 diventa la prima voce, e poi così via fino a δC/δw_1000. Il gradiente ci consente di padroneggiare quella che un tempo era una funzione cattiva del costo: ora sappiamo come la modifica di ogni variabile modifica la funzione di costo.

Puoi pensare a ogni derivata parziale come a una manopola, dove girarla a sinistra (diminuendolo) o a destra (aumentandolo) regola di conseguenza il costo. Con la forza dell'apprendimento automatico, ora hai mille manopole per modificarlo a tuo piacimento, permettendoti di cambiare con sicurezza il costo.

Ma nessun essere umano può gestire 1000 manopole, figuriamoci i 100 miliardi di una tipica rete neurale. Invece, lasciamo che sia la rete a farlo.
Se un modello di apprendimento automatico ha ?C, sa esattamente come ogni manopola modifica il costo e di quanto ruotarla.

Ad esempio, se ∂C/∂w_1 = 1 e ∂C/∂w_2 = 9, possiamo vedere che ∂C/∂w_2 ha un effetto maggiore sulla funzione di costo. Aumentando la variabile w_2 di 1 unità, il costo C aumenterà di 9!

Quindi la manopola ∂C/∂w_2 ci offre il massimo vantaggio in termini di variazione del costo.

Se questa spiegazione non vi ha convinto, vi consiglio di rileggerla. Le derivate parziali sono alla base dell'intera intuizione del funzionamento delle reti neurali.

Regola della catena con derivate parziali

Vi prometto che questo è l'ultimo pezzo di matematica di cui abbiamo bisogno prima di poter comprendere appieno le reti neurali.
La regola della catena con derivate parziali si basa sull'idea della regola della catena normale. È difficile da spiegare, ma facile da dimostrare.
Supponiamo di avere una funzione v =
3x + 4y, x = 3a² e y = 4b². Come posso ottenere ∂v/∂a o ∂v/∂b se ho a che fare con funzioni composte annidate l'una nell'altra?
Potresti provare a sostituire x in termini di a e y in termini di b per calcolare direttamente ∂v/∂a e ∂v/∂b, ma cosa succederebbe se v fosse più complicato, come 3x?+4y²?

Quindi, sostituendo x = 3a² e y = 4b² in v, si otterrebbe v = 9a¹? + 64b?, che è molto più complicato rispetto a lavorare con ogni equazione separatamente.

La regola della catena semplifica i calcoli con espressioni algebriche complesse affrontandoli a pezzetti. Costruiamo un grafico di calcolo:

La funzione v è composta dalle variabili x e y, e x è composta da a, e y è composta da b. Questo albero è il diagramma più utile da disegnare prima di calcolare le derivate parziali.
Se voglio trovare $\partial v/\partial a$, devo andare da v a x, ottenendo $\partial v/\partial x$, e poi da x ad a, ottenendo $\partial x/\partial a$. Quindi si moltiplicano tra loro per creare la regola della catena:

Un altro metodo sensato è l'idea della moltiplicazione tra frazioni. Sebbene non si tratti affatto di frazioni, si può in un certo senso vedere come il denominatore di $\partial v/\partial x$ e il numeratore di $\partial x/\partial a$ siano la stessa cosa, ∂x, e quindi "si annullino".
Questo è il metodo a catena, il mio metodo preferito per calcolare le derivate parziali. La catena e l'albero sono il modo più semplice per calcolare queste derivate essenziali.
Proviamone un altro: quanto fa $\partial v/\partial b$?
Per prima cosa usiamo la catena e l'albero per costruire come dovrebbe apparire il calcolo della derivata parziale:

viaggiando da v a y e poi b
1. Costruisci la catena: $\partial v/\partial b = (\partial v/\partial y) * (\partial y/\partial b)$

2. Calcola $\partial v/\partial y$: $v = 3x + 4y$, quindi $\partial v/\partial y$ tratta y come una variabile e tutto il resto come una costante, quindi $\partial v/\partial y = 0 + 4(1) = 4$

3. Calcola $\partial y/\partial b$: $y = 4b^2$, quindi $\partial y/\partial b$ tratta b come una variabile e tutto il resto come una costante, quindi $\partial v/\partial y = 8b$
4. Moltiplica la catena tra loro: $\partial v/\partial b = 4(8b) = 32b$

Una volta che ti sarai abituato alla notazione, ti renderai conto che il calcolo infinitesimale non è poi così difficile. Se capisci il metodo della catena e dell'albero e sei in grado di fare questi calcoli, sei in una posizione ideale per costruire la tua rete neurale.

5) Il modello del perceptron

Il percettrone (in inglese perceptron) è un modello di rete neurale artificiale, il primo di questo genere, introdotto nel 1943 da Warren McCulloch e Walter Pitts, il percettrone può essere considerato come il più semplice modello di rete neurale feed-forward, in quanto gli input alimentano direttamente l'unità di output attraverso connessioni pesate.

Una rete neurale è approssimativamente modellata su un neurone umano. Un neurone prende in ingresso un dato, fa qualcosa e restituisce un output. Come faccia tutto questo non lo so, non sono un biologo. Ma rappresentiamo le reti come una raccolta di un

gruppo di nodi. Ogni nodo ha un valore numerico. Colleghiamo i nodi ad altri nodi usando pesi, che funzionano come moltiplicatori dei valori. È più facile mostrarlo:

In questo primo strato, abbiamo due nodi. Il primo nodo ha un valore di 5 e il secondo nodo ha un valore di 3.

Il primo nodo ha un peso del valore 6 che lo collega al nodo nel secondo strato.

Il secondo nodo ha un peso del valore 9 che lo collega al nodo nel secondo strato.

Per ottenere il valore di un nodo a cui è collegato dai nodi nel livello precedente, si somma fondamentalmente il valore del nodo per il peso di connessione, per tutti i nodi che si collegano a quel nodo.

Lo so, è un po' difficile da spiegare – quindi ti mostrerò solo:

1. Il primo nodo ha valore 5, con peso 6. Moltiplicare insieme il valore del nodo e il peso e il prodotto è 30.

2. Il secondo nodo ha valore 3, con peso 9. Moltiplicare insieme il valore del nodo e del peso e il prodotto è 27.

3. Aggiungi questi due prodotti insieme, e ottieni 30+27=57.

Ti sembra familiare? È un prodotto a punti tra i valori di un nodo e i pesi che li collegano a un nodo nel livello successivo:

Ma al momento non è importante. Basta conoscere queste regole della rete:

1. Ogni nodo di un livello si connette a ogni singolo altro nodo del livello successivo tramite connessioni chiamate pesi.

2. Ogni peso ha un valore predeterminato.

3. Solo il primo livello di nodi inizia ad avere valori, solitamente chiamato livello di input.

4. Ogni altro livello di nodi viene calcolato in base al livello precedente e ai pesi che li collegano.

Passiamo a un esempio più ampio: una rete neurale a 2 strati

Poiché ogni nodo in un livello deve connettersi a ogni altro nodo nel livello successivo, otteniamo 2 volte 2 = 4 pesi. Espandiamo questo concetto ad altre dimensioni:

- 3 nodi nel primo livello, 5 nodi nel secondo livello: ogni nodo nel primo livello deve connettersi a ogni nodo nel secondo livello, e ci sono 5 nodi nel secondo livello, quindi ci sono 5 pesi per nodo. Moltiplicando 3 per i 3 nodi nel primo livello, si ottengono 15 pesi.
- i nodi nel primo livello, j nodi nel secondo livello: si ottiene i volte j = ij numero di pesi.

Il primo nodo nel secondo livello ha ottenuto il valore 57 perché era connesso al peso 6 e al peso 9.

1. Il peso 6 e si connette a un nodo con 5, dandogli il prodotto 30.

2. Il peso 9 e si connette a un nodo con valore 3, dandogli il prodotto 27.

3. Sommando questi due prodotti si ottiene 57.

Il secondo nodo nel secondo strato ha ottenuto il valore 43 perché era connesso al peso 2 e al peso 11.

1. Il peso 2 e si connette a un nodo con valore 5, dandogli il prodotto 10.

2. Il peso 11 si connette a un nodo con valore 3, dandogli il prodotto 33.

3. Sommando questi due prodotti si ottiene 43.

Pregiudizio

L'ultima cosa di cui abbiamo bisogno prima di passare alla notazione è il concetto di pregiudizio. Un pregiudizio è un numero aggiunto a un nodo alla fine del calcolo del prodotto scalare dei pesi e viene applicato a ogni nodo in ogni strato tranne lo strato di input.

Esempio di una rete a 2 livelli con pregiudizi

Nell'esempio precedente, assegniamo a ciascun nodo del secondo livello il proprio valore di pregiudizio e lo aggiungiamo dopo il calcolo dei pesi. Un pregiudizio è importante perché se ciascuno dei nostri pesi fosse 0, i valori dei neuroni risultanti per il secondo

livello sarebbero sempre 0. La famosa malattia dello 0 si propagherebbe a ogni livello, finché l'intera rete non calcolerebbe semplicemente 0.

Un pregiudizio evita questo problema aggiungendo un valore alla fine di ogni calcolo dei neuroni e non è influenzato dalla moltiplicazione per 0.

Ecco quindi le regole dei pregiudizi:

1. Ogni nodo ha il proprio valore di pregiudizio assegnato casualmente, ad eccezione dei nodi del livello di input.

2. I pregiudizi vengono aggiunti al valore di un nodo dopo la moltiplicazione dei pesi.

Questa prima regola è importante e intuitiva. Immagina se nella frazione di secondo in cui vedi una volpe e il tuo cervello cercasse di classificare quella macchia marroncina come una volpe piuttosto che come un amalgama di colori, il tuo cervello decidesse: "Accidenti amico, fammi aggiungere una tiara e una giacca a quella cosa prima di elaborarla".

Vediamo il mondo come input grezzo, non come input distorto.

Intuizione delle reti neurali

Pesi, nodi, pregiudizi... ma cosa sono? Sono d'accordo che ci sia un po' confusione, ma ricordate quelle manopole di cui ho parlato prima nella sezione sulle derivate parziali? Pesi e pregiudizi sono le manopole che possiamo controllare per ottenere l'output desiderato. Non abbiamo controlli sul primo strato, poiché è l'input grezzo (pensate al vostro cervello che classifica una volpe: non potete decidere subito come classificarla), e non abbiamo controllo su alcun nodo della rete, poiché questi sono sempre determinati dai valori dei nodi nello strato precedente.

L'unica cosa che possiamo controllare sono i pesi e i pregiudizi, le nostre fidate manopole e quadranti che solo il nostro computer sa come usare e regolare.

Quindi la nostra rete neurale calcolerà una funzione di costo C che include tutti quei pesi e i pregiudizi, e potrà imparare come aggiornarli di conseguenza per minimizzare il costo attraverso il gradiente ?C.

Riepilogo

Le reti neurali sono costituite da 4 componenti: nodi, pesi, pregiudizi e costi. Ecco i passaggi per gestire una rete neurale:

1. Si forniscono i dati di input alla rete come livello di input, quindi la rete utilizza tali valori provenienti dal livello di input e i pesi che lo collegano al secondo livello per calcolare i valori per quest'ultimo.

2. Questo processo si propaga attraverso tutti i livelli della rete, finché ogni nodo della rete non ha un valore.

3. L'ultimo livello produrrà la previsione della rete, che verrà poi confrontata con i dati etichettati per fornire una misura di costo per la rete.

4. In base al costo, possiamo calcolare il gradiente rispetto al costo ?C (nabla C) e aggiornare di conseguenza pesi e pregiudizio.

5. Ripetere i passaggi da 1 a 4 fino a ridurre al minimo il costo.

Ecco le regole delle reti neurali da tenere a mente:

- Un nodo in un livello è connesso a ogni singolo nodo del livello successivo tramite connessioni chiamate pesi.
- Inizialmente, i pesi vengono inizializzati in modo casuale, ma li modifichiamo in base al gradiente ?C.
- Ogni nodo della rete, ad eccezione dei nodi del livello di input, ha un pregiudizio, che viene aggiunto al nodo dopo la moltiplicazione peso-nodo. I valori iniziali dei pregiudizi non sono importanti: basta assegnarli in modo casuale.

Possiamo ora passare alla notazione base delle reti neurali:

6) Notazione della rete neurale

Il segreto per comprendere le reti neurali è avere la notazione corretta. Se si hanno troppi indici e nodi che si muovono in giro, combinandosi a formare il mostro di Lochness delle equazioni, anche l'esperto di machine learning più capace tremerà dalla paura. La semplicità è fondamentale. Tutto avrà senso nella sezione 7, dove illustrerò la vera intuizione dietro le reti neurali.

Prima, però, dobbiamo imparare la notazione.

Livelli di una rete

Pensate all'immagine standard di un cervello: pensate a neuroni che si connettono ad altri neuroni per chilometri e chilometri, non a un gigantesco muro di neuroni collegati uno accanto all'altro.

I livelli delle reti neurali si basano sulla stessa premessa. Abbiamo tre tipi di livelli:

- Livello di input: questo è il primo livello di una rete neurale, ma non è propriamente un livello. Consideriamo il livello di input come l'input grezzo del modello, non come parte integrante della rete neurale stessa.
- Livelli nascosti: tutti i livelli tra il livello di input e il livello di output sono chiamati livelli nascosti. Come suggerisce il nome, non sappiamo esattamente cosa facciano dietro le quinte. Possiamo solo interpretare il significato del livello di input, input grezzo, e il significato del livello di output: l'output della rete.
- Livello di output: L'ultimo livello è il livello di output, che dovrebbe generare la previsione della rete.

Possiamo progettare una rete per provare a risolvere qualsiasi problema possibile. Ad esempio, cosa succederebbe se volessimo prevedere se una persona è a rischio di malattie cardiovascolari basandoci esclusivamente sul suo peso e altezza? Ecco i numeri:

- Livello di input: Accettiamo due input, il peso e l'altezza di una persona, quindi il nostro livello di input avrà due nodi.
- Livelli nascosti: Tutti i nodi e i livelli che vogliamo. Il cielo è il limite.
- Livello di output: Ci sono solo due possibilità, la persona è a rischio di malattie cardiovascolari o non a rischio, il che significa che possiamo codificare queste possibilità con un singolo output di probabilità. 0 per non a rischio e 1 per a rischio. Il livello di output avrà quindi un solo nodo, poiché dobbiamo generare un solo numero.

Ecco un esempio di rete che ho creato e che useremo per sviluppare la nostra intuizione per il resto del capitolo:

Il livello di input ha 2 nodi, il livello nascosto uno ne ha 3, il livello nascosto due ne ha 3 e il livello di output ne ha 1.

Quando si lavora con i livelli, è essenziale numerare ogni livello e sapere quanti nodi ha ogni livello.

Contiamo i livelli a partire dal primo livello nascosto, escludendo il livello di input (è input grezzo, quindi non conta). Tuttavia, a volte considereremo anche il livello di input come livello 0, poiché semplifica i calcoli e la notazione. È importante ricordare che la convenzione è quella di etichettare sempre il primo livello nascosto come primo livello della rete.

Nella rete sopra, abbiamo due livelli nascosti e un livello di output, quindi abbiamo 3 livelli in totale.

Rappresentiamo il numero totale di livelli nella rete utilizzando la variabile L,e qui L = 3.

Per contare i nodi, rappresentiamo il numero di nodi in ciascun livello con $n^{[L]}$, dove L è il numero del livello. Per questo, è possibile includere facoltativamente il livello di input, che rappresenteremo con 0.

Scriviamo tutte le notazioni relative alle dimensioni dei livelli per $n^{[L]}$ per tutti i numeri di livello.

$n^{[0]}=2, n^{[1]}=3, n^{[2]}=3, n^{[3]}=1$

$n^{[0]}$ rappresenta il numero di nodi nel livello di input, $n^{[1]}$ rappresenta il numero di nodi nel primo livello nascosto, $n^{[2]}$ rappresenta il numero di nodi nel secondo livello nascosto e $n^{[3]}$ rappresenta il numero di nodi nel terzo livello nascosto.

Domanda veloce: quanto vale $n^{[L]}$?

L è il numero totale di livelli nella nostra rete (escluso il livello di input), che è 3, quindi L = 3. Quindi $n^{[L]} = n^{[3]} = 1$ perché abbiamo 1 nodo nel livello di output.

I pesi

Come si indica il peso? Te lo spiego un poco meglio: quali sono le dimensioni di tutti i pesi in una rete?

Concentriamoci su una piccola porzione della nostra rete, solo il livello di input e il primo livello nascosto.

$L=1\ n^{[0]}=2\ n^{[1]}=3\ n^{[L-1]}=2\ n^{[L]}=3$

Il numero di nodi nel livello di input (livello 0) è $n^{[0]} = 2$, e il numero di nodi nel primo livello nascosto (livello 1) è $n^{[1]} = 3$.

Vogliamo trovare un modo sintetico per rappresentare l'insieme dei pesi che collegano il livello corrente (livello 1) al livello precedente (livello 0), ma come possiamo farlo? Esiste uno schema?

Analizziamo il nostro ragionamento:

- Ogni nodo nel livello 0 si connette a ogni nodo nel livello 1.
- Ci sono tre nodi nel livello 1, quindi ogni nodo nel livello 0 ha 3 connessioni.
- Ci sono 2 nodi nel livello 0, quindi abbiamo 3*2 = 6 connessioni, il che significa che ci sono 6 pesi in totale.

Abbiamo scoperto quanti pesi ci sono tra il livello 0 e il livello 1, ma possiamo estendere questo calcolo a qualsiasi livello.

Per ogni livello L, ci saranno sempre $n^{[L]} * n^{[L-1]}$ pesi che collegano quei due livelli. Qui L = 1, quindi $n^{[L]} = 3$ (numero di nodi nel livello 1) e $n^{[L-1]} = 2$ (numero di nodi nel livello 0). Otteniamo 3*2 = 6 pesi, proprio come prima.

Perché questo è importante? Beh, se ricordate dall'ultima volta, trovare il valore di un nodo nel livello successivo richiedeva molti calcoli:

È facile perdersi nella foresta di prodotti scalari, quindi proviamo un metodo più semplice: la moltiplicazione di matrici. Ecco il nostro obiettivo:

- Rappresentare ogni livello della rete come una matrice (riutilizzata da un vettore).
- Rappresentare ogni insieme di pesi tra due livelli come una matrice.

Per prima cosa, costruiamo l'intuizione:

1. Raccogliamo tutti i pesi rossi e li posizioniamo in un vettore: [5, 6, 7]

2. Raccogliamo tutti i pesi azzurri e li posizioniamo in un vettore: [8, 9, 10]

3. Notate come possiamo rappresentare i nodi nel livello 0 come una matrice 2 x 1 (2 righe, 1 colonna), che è praticamente la stessa cosa di un vettore.

4. Notate come possiamo rappresentare i nodi nel livello 1 come una matrice 3 x 1 (3 righe, 1 colonna), che è praticamente la stessa cosa di un vettore.

I valori dei nodi non sono forse solo un elenco di numeri? Perché non usare un vettore? Anche se potremmo farlo, usare le matrici semplifica la matematica per la moltiplicazione di matrici. Un vettore e una matrice con una singola colonna o una singola riga sono essenzialmente equivalenti, ma rappresentare i livelli come matrici sarà molto più utile in futuro, una volta che avremo chiarito tutti i calcoli.

Dopotutto, perché gestire 16 vettori diversi quando si possono gestire 2 matrici?

Quindi vogliamo creare un'equazione di moltiplicazione di matrici che moltiplichi una matrice 2 x 1 per la matrice dei pesi e che dia come risultato una matrice [3 x 1].

Proviamo:

Un primo tentativo

Siamo ingenui, le nostre dimensioni sono sbagliate! Non si può moltiplicare una matrice 2 x 1 per una matrice 2 x 3 perché le dimensioni centrali non coincidono.

Riproviamo:

Con le dimensioni corrispondenti, otteniamo ciò che vogliamo.

Ok, funziona. Notate come abbiamo ottenuto esattamente le stesse risposte un minuto fa quando abbiamo eseguito manualmente il prodotto scalare, ma ora abbiamo una rappresentazione concisa di tutti i valori dei nodi nel primo livello.

Sebbene questa sia una soluzione perfetta (e molto migliore dei prodotti scalari), faremo un ulteriore passo avanti per rendere la costruzione di reti neurali ancora più semplice.

Trasponiamo la matrice dei pesi e rendiamola la prima matrice nella moltiplicazione di matrici.

Ma aspetta, l'ordine non è importante? Sì, ma in questo caso particolare produciamo comunque esattamente gli stessi calcoli e la stessa soluzione, quindi va tutto bene.

Una matrice trasposta 3x2 moltiplicata per una matrice 2x1 ci dà una matrice 3x1 come prima.

Ora che abbiamo rappresentato la nostra matrice dei pesi come una matrice 3x2, possiamo estrapolare questo processo per qualsiasi strato.

- Lo strato 0 ha 2 nodi e, poiché il primo strato ha 3 nodi, ogni nodo nello strato 0 ha 3 connessioni con pesi.
- Trattiamo l'insieme delle connessioni di pesi che appartengono a un nodo nello strato 0 come un vettore di dimensione 3, poiché ci sono 3 connessioni.
- Ci sono due nodi nello strato 0, quindi abbiamo 2 insiemi di pesi.

- Per formare la matrice dei pesi, impiliamo i vettori uno accanto all'altro verticalmente, in modo che ogni vettore sia una colonna a sé stante. Questo ci dà una matrice 3x2.

Ora possiamo descrivere le dimensioni della matrice dei pesi per qualsiasi strato L che collega lo strato precedente L-1.

$W^{[L]}$ ha dimensioni $n^{[L]}$ x $n^{[L-1]}$ se $n^{[L]}$ è il numero di nodi nel livello L e $n^{[L-1]}$ è il numero di numeri nel livello precedente L-1.

Il livello 1 ha 3 nodi e il livello 0 ne ha 2, quindi la dimensione sarà 3 x 2. Semplice, vero?

Usando questa formula, possiamo trovare le dimensioni di tutti i pesi nella rete

$n^{[0]} = 2$, $n^{[1]} = 3$, $n^{[2]} = 3$, $n^{[3]} = 1$

- $W^{[1]}$: 3 x 2 matrice
- $W^{[2]}$: 3 x 3 matrice
- $W^{[3]}$: 1 x 3 matrice

Pregiudizi

I pregiudizi sono ancora più semplici dei pesi. Poiché ogni neurone in un livello, ad eccezione del livello di input, avrà un pregiudizio, la matrice dei pregiudizi per un livello avrà esattamente le stesse dimensioni della matrice dello stesso livello.

Torniamo al livello 0 e ai primi livelli:

L=1 $n^{[0]}$=2 $n^{[1]}$=3 $n^{[L-1]}$=2 $n^{[L]}$=3

I nodi nel livello di input non avranno alcun pregiudizio, ma ogni nodo nel primo livello avrà pregiudizi. Ci sono tre nodi nel primo strato, quindi ci sono tre pregiudizi.

Proprio come abbiamo rappresentato il primo livello come una matrice 3 x 1, anche i pregiudizio nel primo strato saranno una matrice 3 x 1.

In generale, seguiamo questa regola per trovare le dimensioni dei pregiudizi in un livello:

Per qualsiasi strato L,la matrice di pregiudizio $b^{[L]}$ ha le dimensioni $n^{[L]}$ x 1.

Ora abbiamo tutti i blocchi di cui abbiamo bisogno per capire come le reti neurali calcolano un output nel processo di feed-forward:

7) Propagazione in avanti (Feed Forward)

Il modo migliore per capire la propagazione in avanti è semplicemente farla. Qualche cosa l'abbiamo fatta, ma ho tralasciato parecchie cose.
Torniamo a questo esempio, però stavolta useremo la notazione tipica delle reti neurali.

Esempio di una rete a 2 livelli con pregiudizi

Per ora, rappresentiamo la matrice dei nodi in uno strato come $z^{[L]}$Per qualsiasi strato l.
Scriviamo quello che sappiamo:
$z^{[0]}$: la matrice 2 x 1 degli elementi [3, 2], che rappresenta il livello 0.
$z^{[1]}$: la matrice 2 x 1 che rappresenta il primo strato.
$W^{[1]}$: la matrice 2 x 2 che rappresenta i pesi che collegano il primo strato e lo strato 0.
$b^{[1]}$: la matrice 2 x 1 degli elementi [1, 4], che rappresentano i pregiudizi per il primo livello.
Ora useremo la notazione per descrivere i calcoli invece dei numeri:

La moltiplicazione tra matrici funziona, allora noi moltiplichiamo la matrice del peso per la matrice del nodo del livello precedente, e poi aggiungiamo la matrice del pregiudizio del livello corrente per ottenere il risultato finale.
Solo così possiamo ricontrollare la nostra matematica, ecco i calcoli:

Una rappresentazione dell'equazione della matrice della rete di 2 strati di cui sopra

Arriviamo alla stessa risposta, quindi sappiamo che la nostra formula è corretta.
Descriviamola:
Per ottenere i valori per i nodi nel livello successivo, è necessario moltiplicare la matrice del peso per i valori del nodo dal livello precedente e quindi aggiungere al risultato i valori del pregiudizio.

Le funzioni di attivazione

Ho omesso un passaggio del processo di feed-forward, ma lo affrontiamo ora.

Una volta ottenuto un valore per il neurone con la moltiplicazione della matrice dei pesi e l'addizione del pregiudizio, dobbiamo ridurre tale valore a qualcosa che le nostre reti possano gestire.
Le funzioni di attivazione eseguono questa modifica.
Ci sono due attributi importanti che una funzione di attivazione deve avere:
1. Deve essere non lineare, ovvero NON PUÒ essere nella forma g(z) = mz + b.
2. Deve comprimere l'input in un intervallo predeterminato, come da 0 a 1.
La funzione di attivazione che useremo è la funzione sigmoide, che si presenta così:

Come si presenta la funzione sigmoide
Useremo questa equazione perché modella più fedelmente il funzionamento reale di un neurone: sono accesi o spenti (on-off), ma hanno anche una piccola finestra di transizione che è una sorta di situazione di metà acceso e metà spento.
Una nota per il futuro: la convenzione prevede di indicare la funzione di attivazione come g(z).
La funzione sigmoide varia tra 0 e 1. Spostandosi verso destra, la funzione sigmoide restituisce una linea piatta a 1, e spostandosi verso sinistra, la funzione sigmoide restituisce linee piatte a 0.
Ora che sappiamo cos'è una funzione di attivazione, possiamo tornare alla nostra rete di prima:

Il calcolo di $z^{[1]}$ è solo un passaggio intermedio, in cui il primo nodo nel livello 1 ha valore 27 e il secondo nodo nel livello 1 ha valore 29.
Ora passiamo questi valori di z alla nostra funzione di attivazione sigmoide $g(z) = 1/(1 + e(-z))$ per ottenere i valori effettivi per i nodi in un livello, $a^{[L]}$.
Quando passiamo 27 in g(z), otteniamo g(27) = 0,99. Per 29 in g(z), otteniamo g(29) = 0,99.

Ora possiamo effettivamente rappresentare il primo livello come è veramente: $a^{[1]}$- [0.99, 0.99].
Mettiamo questi calcoli in una formula:

$z^{[1]}$ è il nostro risultato intermedio
$a^{[1]} = g(z^{[1]})$
Passiamo i valori di $z^{[L]}$ alla nostra funzione di attivazione sigmoide per ottenere i valori finali dei nodi in un livello.

Quindi il processo di feed forward prevede i seguenti passaggi:
Moltiplica la matrice dei pesi per le attivazioni del livello precedente, quindi aggiunge la matrice del pregiudizio del livello corrente.
Questo fornisce i valori dei nodi pre attivati per un livello L, $z^{[L]}$. Passiamo il livello pre attivato $z^{[L]}$ alla funzione di attivazione sigmoide g(z) per ottenere $a^{[L]}$. Questo rappresenta i valori finali dei nodi in un livello.
Prima di generalizzare completamente per qualsiasi livello L, dobbiamo parlare di input e output.

Input e output.

Tornando ai principi del machine learning, abbiamo bisogno di dati affinché una macchina impari.
Tornando all'esempio delle malattie cardiovascolari, abbiamo fornito alla nostra rete due input: altezza e peso. Cosa succederebbe se avessimo 10.000 righe in un database, ciascuna rappresentante l'altezza e il peso di una persona? Avremmo una matrice di dati di input di 10.000 x 2. Chiameremmo ogni persona un campione di training e, poiché abbiamo record di 10.000 persone, abbiamo 10.000 campioni di training.

Indichiamo la matrice di dati di input come X.

Abbiamo anche un database di 10.000 righe, corrispondenti alle stesse persone, ciascuna con un numero che può essere 0 o 1: 0 se non sono a rischio di malattie cardiovascolari e 1 se lo sono. Questo è ciò che vogliamo per i nostri dati di output.

Indichiamo la matrice dei dati di output come Y.

Poiché i dati di output erano un vettore unidimensionale, è meglio renderli una matrice con una colonna, poiché ciò semplifica i calcoli.
Questo è il machine learning in poche parole: forniamo gli input e gli output in modo che la rete neurale possa apprendere i pattern e poi generalizzare a nuovi input a cui nessuno conosce la risposta.
Perché è importante? Perché utilizzeremo X come livello di input per la nostra rete, fornendo alla rete tutti i dati di training.
Ma ne parleremo più avanti, quando parleremo della vettorizzazione. Per ora, scegliamo una persona come unico campione di training: Giovanni, che pesa 70 kg ed è alto 180 cm.
Rappresentiamo Giovanni, il campione di training, come $X^{[i]}$ per l'i-esimo campione di training, che rappresenteremo come $X^{[i]} = [150, 70]$.
Poiché useremo questo campione di training come input per la nostra rete, e sappiamo di poter chiamare il livello di input anche livello 0-esimo, non dovremmo designare l'input [150, 70] come $a^{[0]}$?

Ecco come apparirà la nostra rete:

Facciamo i calcoli per il livello 1 e assicuriamoci che le nostre dimensioni corrispondano:
• $a^{[0]}$ È il nostro livello di input con dimensioni 2 x 1 poiché il livello di ingresso ha 2 nodi.
• $A^{[1]}$ È l'attivazione del primo livello con dimensioni 3 x 1 poiché il livello 1 ha 3 nodi.

• $W^{[1]}$ È il nostro strato di matrice di peso con dimensioni $n^{[1]}$ x $n^{[0]}$3 x 2.

• $b^{[1]}$ È la matrice di pregiudizio per il livello 1 con dimensioni 3 x 1 dal momento che il primo strato ha 3 nodi.

Ecco le equazioni, che dovrebbero essere ormai molto familiari.

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g(z^{[1]})$

Il bello di come abbiamo impostato la nostra notazione è che non dobbiamo effettivamente controllare i nostri calcoli. Finché le dimensioni della matrice sono corrette e la moltiplicazione è corretta, possiamo essere sicuri al 99% che i nostri calcoli stiano funzionando.

Un'ultima volta, passiamo in rassegna le dimensioni:

$\mathbf{z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}}$

3x1 3x2 2x1 3x1

a[1] = g(z[1])

3x1 3x1

Sì, funziona tutto.

Probabilmente hai già notato il modello feed-forward, ma lo generalizzerò in modo che tu possa fare il resto dei calcoli della matrice da solo.

Per qualsiasi livello L,ecco il processo di feed-forward per una rete neurale:

$\mathbf{z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]}}$

$\mathbf{a^{[L]} = g(z^{[L]})}$

Come esercizio, prova a rappresentare $a^{[2]}$ e $a^{[3]}$ come espressioni algebriche usando questo processo di feed-forward.

Una volta raggiunto l'ultimo livello L=3, otterrai l'output finale della rete: $a^{[L]}$. Nota inoltre che tutti i nodi della rete avranno un valore compreso tra 0 e 1 a causa della funzione di attivazione sigmoide, che ci consente di creare test di probabilità.

Nell'apprendimento automatico, notiamo l'output di un modello di apprendimento automatico in un modo speciale: ŷ (pronunciato "y-hat"). ŷ rappresenta la previsione della rete e confrontiamo ŷ con y (i valori reali su cui stiamo testando il modello) per creare una misura di errore per il modello che misuri quanto scarsamente la rete predice i dati desiderati.

Torniamo a Giovanni: se la rete prevede che per Giovanni, nel campione di training [150, 70], abbia una probabilità dello 0,72 di essere a rischio di malattie cardiovascolari, ma si scopre che la vera diagnosi di Giovanni è che non è a rischio di malattie cardiovascolari. Quindi, confrontando il nostro valore predetto ŷ = 0,72 con y = 0, il costo per il nostro modello è (previsto-atteso), che è pari a 0,72.

Ottenere un costo pari a 0 significa che il nostro valore predetto ŷ dovrebbe essere esattamente uguale a y, il che significa che il costo ŷ-y = 0, che è ciò che desideriamo.

Prima di scrivere il codice per il processo di feedforward, analizziamolo:

1. Inizializzare in modo casuale i pesi e i pregiudizi per la rete.

2. Eseguire il processo di feedforward per ciascun campione di training.
3. Ottenere ŷ alla fine del processo di feedforward per ciascun campione di training.
4. Definisci una misura del costo per la rete su tutti i campioni di training.
Sembra un sacco di lavoro. Lo è se lo fai in questo modo, ciclando su tutti gli esempi di training, ti garantisco che finirai per essere più confuso di quanto lo fossi prima ancora di iniziare a leggere questo articolo.
L'ultimo passaggio prima di poter diventare maghi delle reti neurali è vettorializzare l'intero processo su tutti gli esempi di training.

8) Vettorializzare, Vettorializzare, Vettorializzare

Questa sarà una sezione breve ma importante. Ora hai tutte le informazioni per eseguire il feedforward da solo, ma non ti ho ancora spiegato come calcolare effettivamente il costo per la rete.

I calcoli dei costi sono facili solo se prima possiamo vettorizzare su tutti i campioni di allenamento della rete.

Riprendiamo i 10.000 record di dati di altezza e peso. La matrice dei dati di input di training che abbiamo chiamato X aveva dimensioni 10.000 x 2. Questo è ciò che significa vettorizzare i nostri dati di training: invece di eseguire un ciclo ed eseguire il processo di feedforward per tutti i campioni di training, eseguiamo il feedforward su tutti i dati di training contemporaneamente.

Per semplificare i calcoli futuri, trasponiamo X in una matrice 2 x 10.000, o più comunemente chiamata n x m, dove n è il numero di caratteristiche e m è il numero di campioni di training.

Cosa intendiamo per caratteristica?

Possiamo considerarlo come il numero di variabili che inseriamo nella rete. La nostra rete considera l'altezza e il peso di una persona per prevedere se sia o meno a rischio di malattie cardiovascolari. Altezza e peso sono le due variabili che usiamo come input per la rete, quindi le chiamiamo le nostre due caratteristiche.

Aspetta, due caratteristiche? Non è lo stesso numero di nodi che abbiamo nel livello di input?

Esatto. Caratteristiche, livello di input, X: tutti intercambiabili. Quindi è un po' complicato dire che X è una matrice n x m, quando $n^{[0]}$ x m è molto più facile da capire.

Attenzione: tutti i nostri valori intermedi e i livelli di attivazione verranno vettorializzati anche su tutti i campioni di training e modificheremo la notazione come segue:

- X: la matrice $n^{[0]}$ x m dei dati di training. La stessa cosa del livello di input.
- $A^{[0]}$: la stessa cosa di X, solo con una notazione diversa per la formula feedforward. È solo il livello di input vettorizzato su tutti i campioni di training per avere una dimensione $n^{[0]}$ x m.

In generale, rappresenteremo il livello vettorizzato delle attivazioni su tutti i campioni di training per un livello L come $A^{[L]}$. Questi livelli di attivazione vettorizzati avranno sempre m colonne (numero di campioni di training) e il loro numero di righe sarà uguale al numero di nodi in quel livello, $n^{[L]}$.

Nota a margine: sebbene tecnicamente abbiamo sempre a che fare con matrici, notiamo una matrice con una lettera maiuscola solo se ha più di una colonna, nel senso che non può essere trasformata in un vettore. Quindi ora le nostre attivazioni.

Non vettorizziamo i pesi e i pregiudizi della nostra rete perché vogliamo una sola copia dei pesi e dei pregiudizi da generalizzare per tutti i campioni di training. Quindi siamo a posto su questo fronte.

Riprendiamo la matematica, tenendo presente quanto segue:

- $n^{[0]}$: 2
- $n^{[1]}$: 3
- $n^{[1]}$ x $n^{[0]}$

$Z^{]1]} = W^{[1]} A^{[0]} + b^{[1]}$ $n^{[1]}$ x m $n^{[0]}$ x m $n^{[1]}$ x 1

1. Moltiplichiamo la matrice dei pesi di dimensioni 3 x 2 per il livello di input vettorizzato di dimensioni 2 x m per ottenere una matrice di 3 x m.

2. La matrice dei pregiudizi è 3 x 1, ma possiamo estenderla a 3 x m semplicemente copiandola più volte, un processo chiamato broadcasting. Quindi possiamo semplicemente sommare due matrici di 3 x m per ottenere $Z^{[1]}$ come matrice di 3 x m.

E per l'output di attivazione finale?

$A^{[1]} = g(Z^{[1]})$

Beh, sappiamo che entrambi $Z^{[1]}$e così via $A^{[1]}$ hanno le stesse dimensioni, quindi è abbastanza facile.

Gran parte delle reti neurali si riduce a ottenere le dimensioni giuste della matrice. Con la vettorializzazione puoi fare affidamento sulla moltiplicazione della matrice piuttosto che su loop complicati.

Espandiamo questo per qualsiasi strato L:

$Z^{[L]} = W^{[L]} A^{[L-1]} + b^{[L]}$

$A^{[L]} = g(Z^{[L]})$

- $Z^{[L]}$Valori pre-attivati per il livello L, vettorializzato in tutti i campioni di allenamento. Ha dimensioni $n^{[L]}$ x m.
- $A^{[L]}$Valori dei nodi per il livello L,vettorializzato in tutti i campioni di allenamento. Ha dimensioni $n^{[L]}$ x m.
- $A^{[L-1]}$Valori dei nodi per il livello L-1, vettorializzato in tutti i campioni di allenamento. Ha dimensioni $n^{[L-1]}$ x m.

Vediamo se riesci da solo a calcolare le dimensioni per $A^{[2]}$e così via $A^{[3]}$.