Deep Learning

Daniele Avolio

September 26, 2023

Contents

1 Introduzione

Appunti

2 Deep Learning 101

In questo corso affronteremo diverse tematiche, il che può sembrare assurdo se ci si pensa.

- Classificazione (binaria, cioè sì o no)
- Multi-class classification (non più binaria)
- Regressione (il guessing viene fatto su un valore numerico)
- Gestione di immagini e riconoscimento
- Serie numeriche (predizioni di mercato e trend)
- Classificazione di testi

2.1 Architetture e strumenti nel deep learning

- Autoencoder
  - Tutti i possibili tipi
  - Qui si fa anche **Clustering** e **Anomaly detection**
- Architetture generative
  - Tutti i possibili tipi
- XAI: Explainable AI

2.2 Libri utili

- "Deep Learning in Python"
- "Tensorflow tutorial"

2.3 Strumenti che useremo

- Tensorflow
  - High-level più di altri
- Keras
  - High-level API basato su Tensorflow
  - Ci saranno cose che non possiamo fare con Keras perché è troppo ad alto livello

2.4 Schema generale di un problema di deep learning

Abbiamo delle coppie $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ dove x_i è un vettore di features e y_i è un valore numerico (regressione) o una classe (classificazione).

$$y_i = f(x_i)$$

Non conosciamo la funzione f, quindi dobbiamo impararla.

$$y = \alpha x + \beta$$

Con una rete neurale puoi approssimare praticamente qualsiasi funzione.

Una rete neurale permette di collegare un input di dati a una funzione di output.

Abbiamo diversi tipi di reti neurali a seconda del tipo di problema che vogliamo risolvere. È importante essere in grado di selezionare l'architettura giusta per risolvere il problema.

Definiamo alcuni concetti che useremo:

- N: rete neurale
- w: valori dei pesi della rete neurale
- f: funzione di output della rete neurale

$$f \in N(w)$$

2.5 Perché si usa il termine "Tensore"?

Un *tensore* non è altro che una matrice.

- 0D tensor: scalar
- 1D tensor: vector
- 2D tensor: matrix
- 3D tensor: tensor

2.6 AI vs DL

- AI: è un ampio insieme di tecniche per risolvere problemi che richiedono "intelligenza".
  - Esempio: Stockfish, un programma che gioca a scacchi.
- DL: È un sottoinsieme di AI che si concentra sull'*astrazione*.
  - L'astrazione consiste nel fornire una funzione che traduce dati di input in dati di output senza conoscere la funzione stessa.
  - È un approccio induttivo: fornisci un input e ti aspetti un output, senza conoscere la funzione che li collega.
  - Questo è completamente diverso dall'AI basata sulla logica.

3 Introduzione alle Reti Neurali

3.1 Il modello di McCulloch-Pitts

Un modello pensato dai due tizi qui presenti,

- Warren McCulloch
- Walter Pitts

Era formato da:

- Un insieme di neuroni
- Un insieme di connessioni tra i neuroni
- Un insieme di pesi associati alle connessioni
- Una funzione di attivazione
- Una funzione di output

Quindi immaginiamo $x_1, x_2, \ldots, x_n$ che vengono dati come input, e quello che viene fuori è un valore di $y \in \{0, 1\}$. Questo è un modello di classificazione binaria.

In soldoni, in deep learning si usa un vettore di numeri per tirare fuori un altro vettore di numeri.

Nella maggior parte del tempo, però, non lavoriamo solamente con i dati.

- immagini
- Testo
- Audio
- ...

Non ci sono concetti di foto, video, immagini. L'unico concetto che esiste è quello dei numeri.

3.2 Modello di Rosenblatt

Qui il modello è leggermente diverso. Gli elementi in questo modello sono i seguenti:

- Valori di input: x_i
- Funzione di attivazione: ϕ
- Pesi degli archi: w_i
- Bias: b

$$h(x|w,b) = h(\sum_{i=1}^{l} w_i \cdot x_i - b) = h(\sum_{i=1}^{l} w_i \cdot x_i) = sign(w^T x) \tag{1}$$

Nota: Quando il **bias** non viene specificato, allora si assume che sia 0.

I pesi w_i sono collegati archi che vanno da x_i al prossimo neurone. Sia il valore di input che il peso sono **numeri reali**. Non sono lo stesso valore, hanno solamente il formato di *reale* che è uguale tra loro. Ciò che viene fatto è solamente la somma della prodotto tra ogni peso w_i e x_i **meno** il bias.

La funzione di attivazione: Dipende. Ogni funzione che ha *2 stati* va bene per noi. Una funzione di attivazione può essere una qualsiasi che in un punto ha valore 1 e in un altro ha valore -1.

3.3 Esempio con 2 neuroni

.

Immaginiamo di avere un piano cartesiano con una retta che interseca in 2 punti.

$$sign(w_1 \cdot x_1 + w_2 \cdot x_2) \tag{2}$$

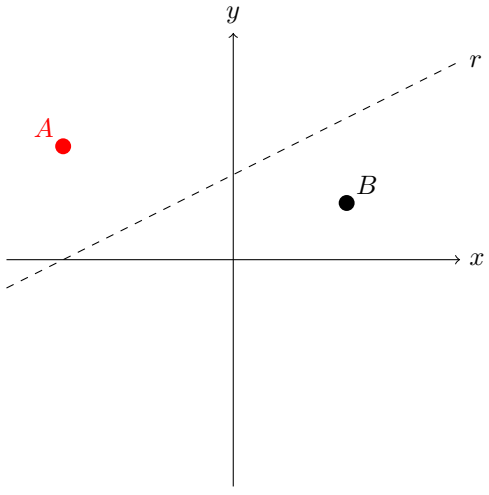
Il parametro della funzione non è altro che **l'equazione di una retta.**

In particolare, se consideriamo la reta che separa gli spazi del piano, vediamo che la retta è **capace di separare 2 punti nel piano.**

Cosa abbiamo:

- Un neurone
- 2 valori in input
- 2 pesi

Con la funzione di attivazione **sign** si avrà come output una retta che separa



dei punti.

Nota: Quando è utile avere un valore output che non è una separazione di punti in un piano? Nel caso della **regressione**.

Oltre la funzione sign:

- Funzione di attivazione lineare
- Funzione di attivazione sigmoid
- Funzione di attivazione Tanh

3.4 Rappresentazione le funzioni logiche

:

3.4.1 AND

Immaginiamo di avere:

- $x_1, x_2 \in \{0, 1\}$
- Bias= -30
- Funzione di attivazione: Logistica o Sigmoid

Come facciamo a rappresentare un AND?

$$h(x) = g(-30 + 20x_1 + 20x_2) \tag{3}$$

x_1	x_2	$h(x)$
0	0	1
0	1	0
1	0	0
1	1	1

Lo stesso ragionamento vale per:

- OR
- NOT
- (NOT x_1) AND (NOT x_2)

3.4.2 Il problema dello XOR

Il percettrone non può imparare regioni che non sono linearmente separabili.

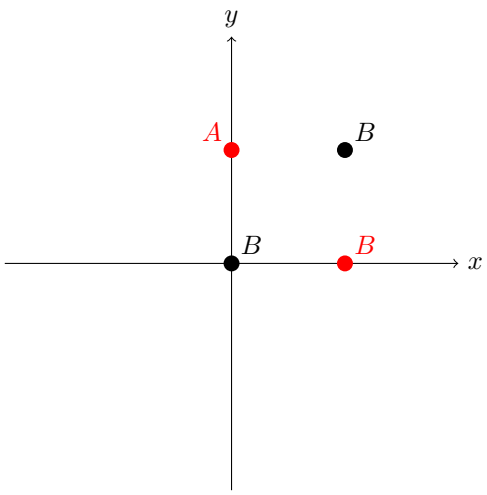


Figure 1: XOR non possibile con 1 percettrone

Come vediamo qui non possoamo tracciare una retta per dividere i due punti. In questo caso ci serve una funzione **non lineare**.

La soluzione a questo problema è **aggiungere LAYER** alla rete neurale. Aggiungere un layer significa aggiungere un neurone successivamente ad un altro.

Maggiore è il numero di layer, maggiore diventa la potenza espressiva della rete. Praticamente possiamo catturare qualsiasi cosa aggiungendo layer alla rete. Questo è **il vero potere del deep learning**.