Costruzione della rete

Costruzione della rete

Premessa

Procediamo alla creazione di una architettura di rete neurale con i seguenti assunti:

1. Conosciamo l'obbiettivo da raggiungere.
2. Possediamo i dati: sintetici o meno.
3. I dati sono stati convertiti in tensori.
4. I dati sono stati opportunamente pre-processati e normalizzati.

A fronte di questo, il passo successivo è scegliere un *framework* e scrivere del codice. Il nostro framework sarà **PyTorch**.

Costruzione della rete

Premessa

Nel DL, per architettura si intende un modello di **ML** basato su:

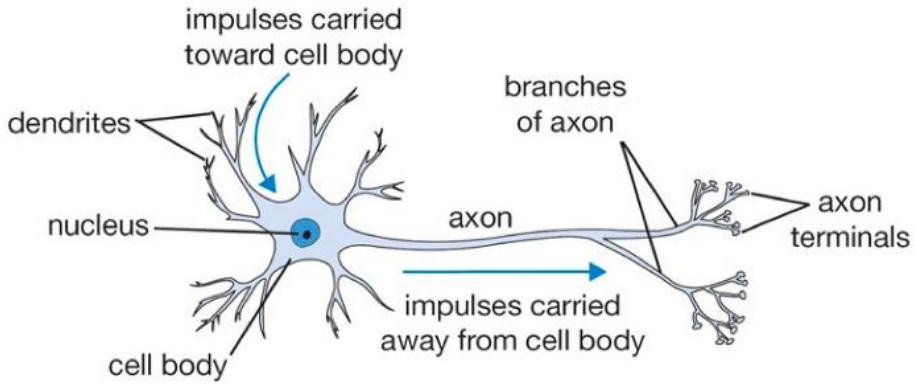
ANN (artificial neural network)

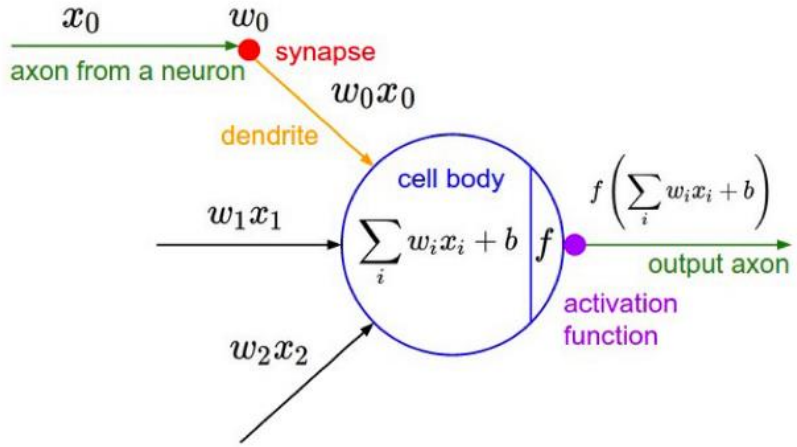
Ognuna di queste parole e ognuno di questi acronimi non è scelto a caso. Comprenderne il significato significa comprendere gli strumenti con cui si andrà a lavorare.

Costruzione della rete

Premessa

L'ispirazione dietro queste reti è biologica e, come nel nostro cervello, il componente fondamentale con cui si costruiranno le architetture è il neurone. Da qui la parola ***neural***.





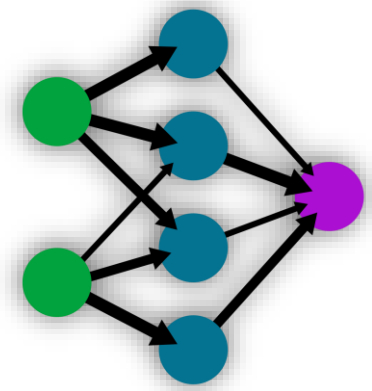
Ciò che noi, però, abbiamo a disposizione è il software perciò la biologia non può far altro che essere rappresentata da delle righe di codice. Da qui la parola ***artificial***.

Costruzione della rete

Premessa

Il neurone, come singolo componente, ha però una capacità limitata di descrivere informazioni o definire stimoli a fronte di input. La vera forza rappresentativa si ha quando **più neuroni vanno ad interconnettersi** fra loro e comunicare a vicenda i propri input e output.

Da qui la parola ***network***.



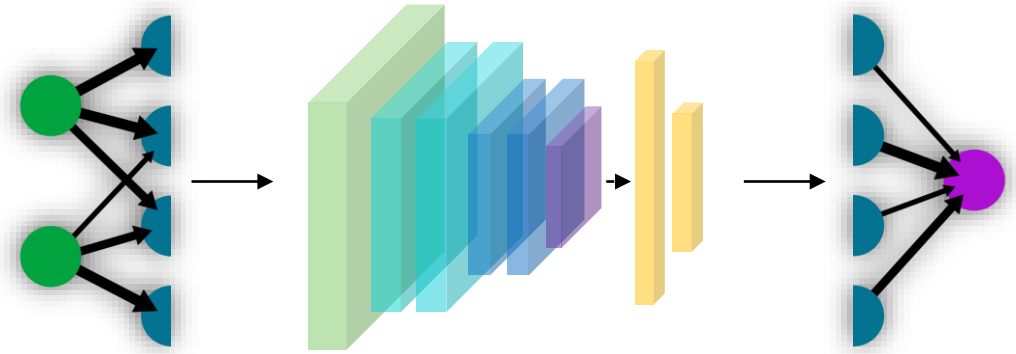
I modi in cui i neuroni si interconnettono fra loro, il lavoro che svolgono, il modo in cui scelgono se emettere o meno segnali sulla base dell'input va poi a definirne gruppi chiamati, in gergo, ***layer***.

Costruzione della rete

Premessa

Distinguiamo perciò gruppi di neuroni totalmente connessi fra loro, gruppi di neuroni posti a formare disposizioni particolari…

Fra questi gruppi, vi andiamo ad aggiungere altrettante operazioni, anch'esse rappresentate da un proprio *layer*, che modificano il modo in cui i segnali viaggiano nella rete, la quantità, il tipo…



Riuscire ad addestrare una rete attraversando molteplici layer e trattando con un numero di neuroni (con i loro parametri) sempre maggiore è ciò che da significato alla parola ***deep learning***.

Costruzione della rete

La rete in PyTorch

In PyTorch, una rete neurale ha queste caratteristiche:

1. È una classe Python.
2. La classe deriva le proprie caratteristiche da *nn.Module*.
3. Implementa un metodo *__init__*.
4. Implementa un metodo *forward*.

Ma non solo…

- *Possiede una funzione che definisce il raggiungimento dell'obbiettivo di addestramento: la funzione di costo o loss.*
- *Definisce il modo in cui la modificare i propri valori per far sì di ottimizzare la loss; definisce quindi un optimizer.*

Costruzione della rete

È una classe

Trattandosi di codice python, l'oggetto / l'entità astratta che va a definire una rete neurale è semplicemente descritto da una classe.

Per questo motivo, la più semplice definizione di ***modello di rete neurale*** lo si trova direttamente nella documentazione:

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Costruzione della rete

Deriva da nn.Module

Derivando da nn.Module, una rete eredità utili vantaggi fra cui…

…la gestione del grafo computazionale, del calcolo dei gradienti, dei parametri di addestramento, di eventuali sotto-moduli e della possibilità di sfruttare le potenzialità di dispositivi come GPU e TPU…

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Costruzione della rete

Implementa *__init__*

Come nella cucina di un ristorante si prepara la linea prima di un servizio (strumenti, ingredienti, utensili…), così una rete in PyTorch implementa il metodo *__init__*.

Qui, vi definisce all'interno i layer che rappresenteranno l'architettura del modello e che l'input dovrà attraversare.

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Costruzione della rete

Implementa *forward*

Dopo aver preparato la linea, gli chef affrontano il servizio e all'arrivo delle comande seguono i passi che porteranno alla preparazione dei piatti.

Implementare il metodo *forward* indica alla rete come i layer si combinano fra loro, l'ordine in cui saranno eseguiti e, più semplicemente, la strada che l'input dovrà attraversare per trasformarsi negli output desiderati.

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Proviamo?