Università di Bologna

Attività progettuale di Sistemi Intelligenti M

Reti neurali artificiali: dal MLP agli ultimi modelli di CNN

Autore:
Ali Alessio Salman

21 agosto 2017

Indice

Capitolo 1

Reti Neurali Artificiali: le basi

1.1 Breve introduzione

Una rete neurale artificiale – chiamata normalmente solo rete neurale (in inglese *Neural Network*) – è un modello di calcolo adattivo, ispirato ai principi di funzionamento del sistema nervoso degli organismi evoluti che secondo l'approccio connessionista[1] possiede una complessità non descrivibile con i metodi simbolici. La caratteristica fondamentale di una rete neurale è che essa è capace di acquisire conoscenza modificando la propria struttura in base alle informazioni esterne (i dati in ingresso) e interne (tramite le connessioni) durante il processo di apprendimento. Le informazioni sono immagazzinate nei parametri della rete, in particolare, nei pesi associati alle connessioni. Sono strutture non lineari in grado di simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non sarebbero in grado di fare.

L'unità base di questa rete è il neurone artificiale introdotto per la prima volta da McCulloch e Pitts nel 1943 (fig. 1.1).

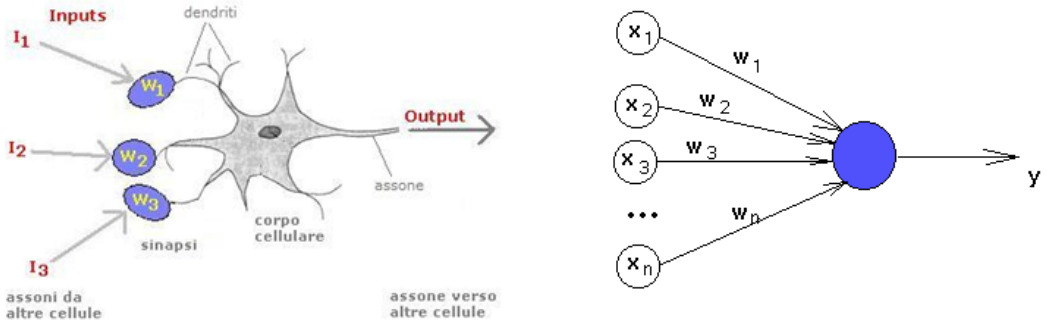

Figura 1.1: Modello di calcolo di un neurone (a sinistra) e schema del neurone artificiale (a destra)

Si tratta di un'unità di calcolo a N ingressi e 1 uscita. Come si può vedere dall'immagine a sinistra gli ingressi rappresentano le terminazioni sinaptiche, quindi sono le uscite di altrettanti neuroni artificiali. A ogni ingresso corrisponde un peso sinaptico w, che stabilisce quanto quel collegamento sinaptico influisca sull'uscita del neurone. Si determina quindi il potenziale del neurone facendo una somma degli ingressi, pesata secondo i pesi w.
A questa viene applicata una funziona di trasferimento non lineare:

$$f(x) = H(\sum_i (w_i x_i)) \tag{1.1}$$

ove H è la funzione gradino di Heaviside [2]. Vi sono, come vedremo, diverse altre funzioni non lineari tipicamente utilizzate come funzioni di attivazioni dei neuroni. Nel '58 Rosenblatt propone il modello di *Percettrone* rifinendo il modello di neurone a soglia, aggiungendo un termine di *bias* e un algoritmo di apprendimento basato sulla minimizzazione dell'errore, cosiddetto *error back-propagation*[3].

$$f(x) = H(\sum_i (w_i x_i) + b), \quad ove \quad b = bias \tag{1.2}$$

$$w_i(t+1) = w_i(t) + \eta \delta x_i(t) \tag{1.3}$$

dove η è una costante di apprendimento strettamente positiva che regola la velocità di apprendimento, detta *learning rate* e δ è la discrepanza tra l'output desiderato e l'effettivo output della rete.

Il percettrone però era in grado di imparare solo funzioni linearmente separabili. Una maniera per oltrepassare questo limite è di combinare insieme le risposte di più percettroni, secondo architetture multistrato.

1.2 Multi-layer Perceptron

Il Multi-layer Perceptron (*MLP*) o percettrone multi-strato è un tipo di rete feed-forward che mappa un set di input ad un set di output. È la naturale estensione del percettrone singolo e permette di distinguere dati non linearmente separabili.

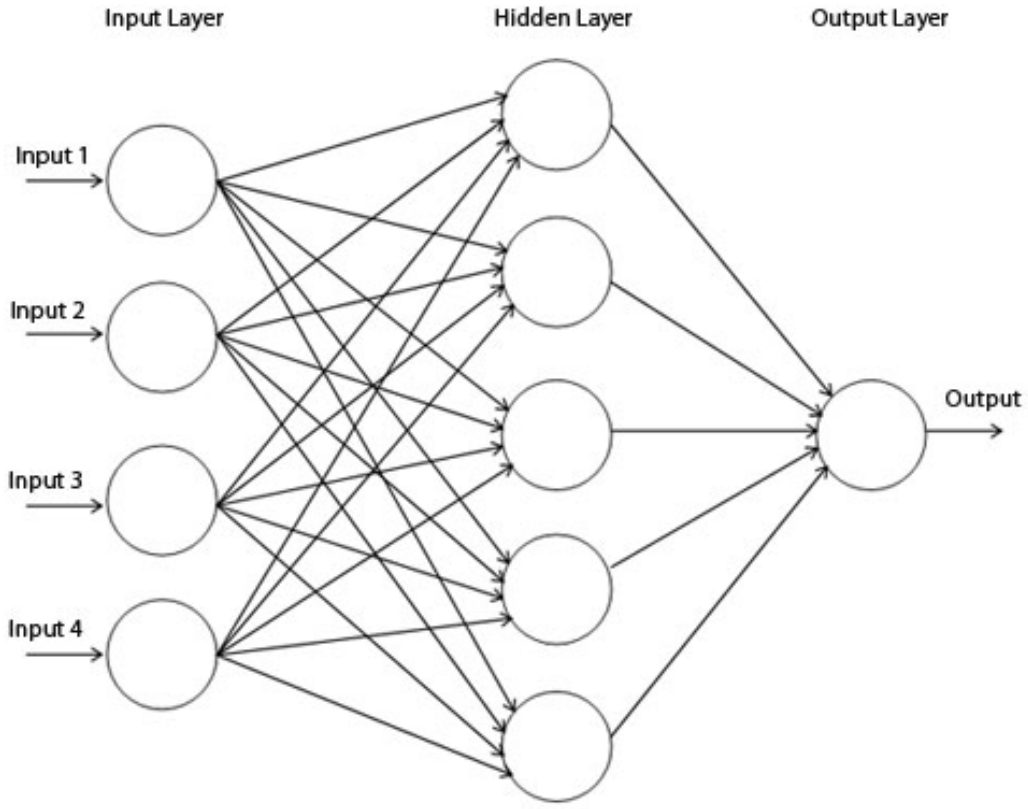

Figura 1.2: Struttura di un percettrone multistrato con un solo strato nascosto

Il *mlp* possiede le seguenti caratteristiche:

- ogni neurone è un percettrone come quello descritto nella sezione 1.1. Ogni unità possiede quindi una propria funzione d'attivazione non lineare.
- a ogni connessione tra due neuroni corrisponde un peso sinaptico w
- è formato da 3 o più strati. In 1.2 è mostrato un MLP con uno strato di input; un solo strato nascosto (o *hidden layer*; ed uno di output.)
- l'uscita di ogni neurone dello strato precedente è l'ingresso per ogni neurone dello strato successivo. È quindi una rete *completamente connessa*. Tuttavia, si possono disconnettere selettivamente settando il peso sinaptico w a 0.
- la dimensione dell'input e la dimensione dell'output dipendono dal numero di neuroni di questi due strati. Il numero di neuroni dello strato nascosto è invece indipendente, anche se influenza di molto le capacità di apprendimento della rete.

Se ogni neurone utilizzasse una funzione lineare allora si potrebbe ridurre l'intera rete ad una composizione di funzioni lineari. Per questo - come detto prima - ogni neurone possiede una funzione di attivazione non lineare.

1.2.1 Strati Nascosti

I cosiddetti *hidden layers* sono una parte molto interessante della rete. Per il teorema di approssimazione universale[4], [4] [1] una rete con un singolo strato nascosto e un numero finiti di neuroni, può essere addestrata per approssimare una qualsiasi funzione continua su uno spazio compatto di $\mathbb{R}^n$. In altre parole, un singolo strato nascosto è abbastanza potente da imparare un ampio numero di funzioni. Precisamente, una rete a 3 strati è in grado di separare regioni convesse con un numero di lati $\leqslant$ numero neuroni nascosti.

Reti con un numero di strati nascosti maggiore di 3 vengono chiamate reti neurali profonde o *deep neural network*; esse sono in grado di separare regioni qualsiasi, quindi di approssimare praticamente qualsiasi funzione. Il primo e l'ultimo strato devono avere un numero di neuroni pari alla dimensione dello spazio di ingresso e quello di uscita. Queste sono le terminazioni della *"black box"* che rappresenta la funzione che vogliamo approssimare.

L'aggiunta di ulteriori strati non cambia *formalmente* il numero di funzioni che si possono approssimare; tuttavia vedremo che nella pratica un numero elevato di strati migliori di gran lunga le performance della rete su determinati task, essendo gli hidden layers gli strati dove la rete memorizza la propria rappresentazione astratta dei dati in ingresso. Nel capitolo 4 vedremo un'architettura all'avanguardia con addirittura 152 strati.

1.3 Caso di studio: prevedere il profitto di un ristorante

Prendendo spunto dalla traccia d'esame di Sistemi Intelligenti M del 2 Aprile 2009:
"Loris è figlio della titolare di una famoso spaccio di piadine nel Riminese e sta tornando in Italia dopo aver frequentato con successo un prestigioso Master in Business Administration ad Harvard, a cui si è iscritto inseguendo il sogno di esportare in tutto il mondo la piadina romagnola. Nel lungo viaggio in prima classe, medita su come presentare alla mamma, che sa essere un tantino restia alle innovazioni, il progetto di aprire un ristorantino a New York City."

Loris ha esportato con successo la piadina a NY, si veda [5] ma col passare degli anni ha notato alcuni problemi e vuole utilizzare di nuovo le sue brillanti capacità analitiche per migliorare il profitto del suo ambizioso ristorante.
I problemi sono 2:

1. il ristorante è conosciuto ormai - si sa che tutti vogliono mangiare italiano - ma il numero dei coperti era rimasto a 22, come quelli iniziali;

2. gli orari di apertura sono troppo lunghi e vi sono alcune zone morte dove il costo di mantenere aperto il ristorante è maggiore rispetto al ricavo dei pochi clienti che si siedono a mangiare durante quelle ore;

Secondo la National Restaurant Association[6] [7] il profitto medio lordo annuo di un ristorante negli Stati Uniti varia dal 2 al 6%. Così Loris ha collezionato alcuni dati riguardo gli ultimi anni e - attratto da tutta quest'entusiasmo attorno alle reti neurali - decide di provare ad utilizzarle per trovare il trade-off ottimale di coperti e di orari di apertura settimanali per massimizzare il profitto del suo ristorante.

1.4 Implementazione da zero di un MLP

Per il suddetto caso di studio si è implementato da zero un percettrone multistrato a 3 strati come quello illustrato in sezione 1.2. Vediamo qui di seguito le varie parti da implementare passo passo per costruire un MLP, addestrarlo e testare che l'addestramento sia stato eseguito in maniera corretta. Il progetto è realizzato in Lua, per sfruttare il framework per il *Machine Learning* Torch3

1.4.1 Dataset e Architettura

Nei vari anni, Loris ha cambiato le due variabili in gioco annotando di volta in volta i risultati. Siccome i coperti erano troppo pochi e le file d'attesa erano troppo lunghe, il ristorante perdeva alcuni clienti. Quindi Loris ha portato i coperti a 25 e diminuendo le ore settimanali a 38, ed i profitti sono aumentati. Tuttavia, non era raro che ancora qualche cliente dovesse aspettare in piedi per troppo tempo (si sa la vita a NY è frenetica), finendo poi per scegliere un ristorante adiacente. Inoltre, aveva diminuito le ore di troppo; nel weekend i clienti arrivavano fino a tardi, quindi rimanere aperti un'ora in più sarebbe stato lungimirante. Così, dopo l'allargamento della sala principale, ha aggiunto altri coperti ed aumentato le ore settimanali a 40, segnando un record personale di 4.4% di profitti annui.

Quindi, i dati in ingresso ed in uscita, in X e Y rispettivamente sono:

$$X = \begin{pmatrix} 22 & 42 \\ 25 & 38 \\ 30 & 40 \end{pmatrix} Y = \begin{pmatrix} 2.8 \\ 3.4 \\ 4.4 \end{pmatrix}$$

Osservando le dimensioni dei dati si nota che la rete deve avere 2 input e dare in uscita 1 output, che chiameremo $\hat{y}$, in contrapposizione a y che è l'uscita desiderata. Per quanto detto nella sezione 1.2, il MLP deve avere 2 neuroni nello strato di ingresso ed 1 solo in uscita. Per quanto riguarda lo strato nascosto, la rete avrà 3 neuroni. Questi parametri vengono chiamati *hyperparameters*. In figura 1.3 è mostrata l'architettura generale della nostra rete.

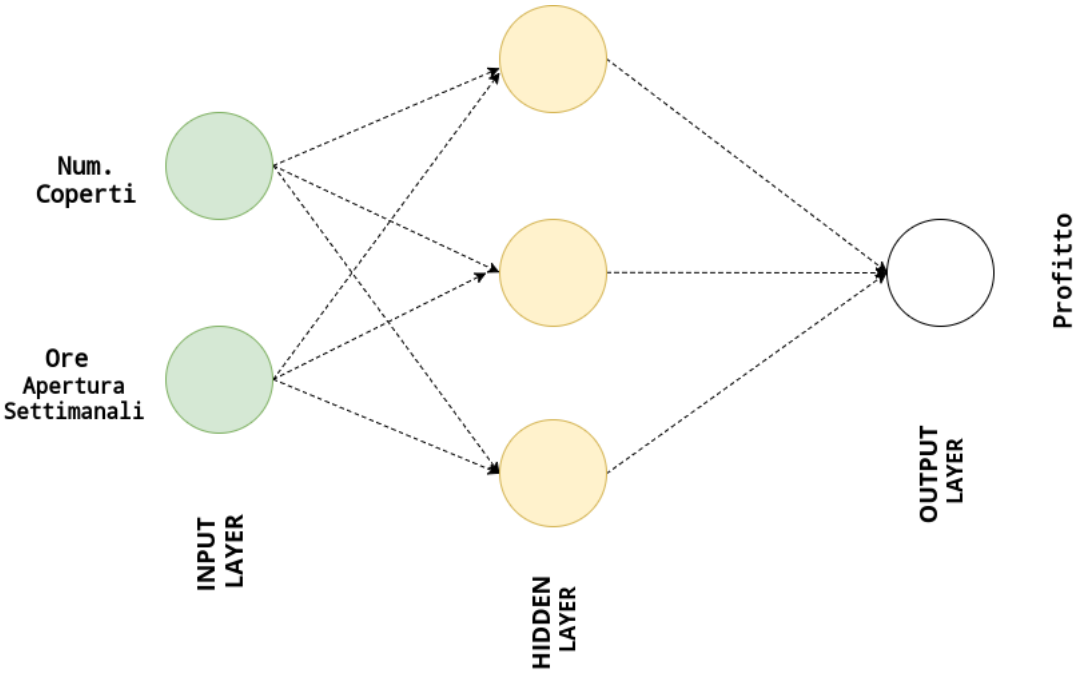

Figura 1.3: Struttura di un percettrone multistrato con un solo strato nascosto

1.4.2 Modello Neurone & Forward Propagation

1.4.3 Backpropagation

Come si fa ad addestrare una rete multistrato con diversi neuroni per strato? Tramite l'algoritmo di *backpropagation of errors* di ...

1.4.4 Verifica numerica del gradiente

1.4.5 Addestramento

1.5 Ottimizzazione: diverse tecniche

1.6 Overfitting: come rilevarlo e risolverlo

1.7 Risultati

Capitolo 2

Reti Neurali Convoluzionali

2.1 Breve introduzione

La reti neurali convoluzionali, alle quali ci riferiremo con l'abbrevazione *CNN* - dall'inglese *Convolutional Neural Network*, sono un'evoluzione delle normali reti artificiali caratterizzati da una particolare architettura che le ha rese negli anni molto efficaci su diversi compiti.

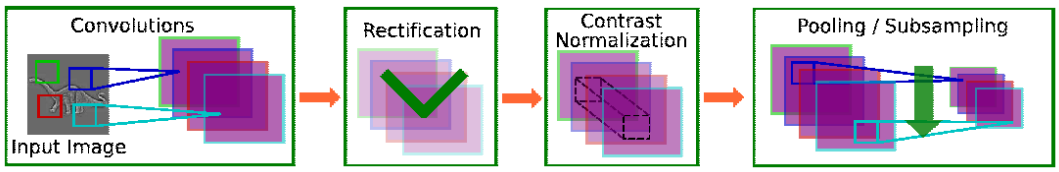

Figura 2.1: I diversi strati tipici di una CNN

2.2 Architettura

2.2.1 Strato di Convoluzione

2.2.2 Strato di Pooling

2.2.3 Strato completamente connesso (FC)

2.3 Applicazioni e risultati

Capitolo 3

CNN: implementazione e addestramento

3.1 Il framework Torch

3.2 Modello di CNN basato su LeNet

3.2.1 Strato di Convoluzione

3.2.2 Strato di Pooling

3.2.3 Strato completamente connesso (FC)

3.3 Dataset d'esempio: CIFAR

3.4 Addestramento su CIFAR

Vedremo nel capitolo seguente un confronto dei risultati di una CNN con un'architettura allo stato dell'arte addestrata sullo stesso dataset.

Capitolo 4

Addestrare un modello allo stato dell'arte

4.1 ResNet

4.2 Implementazione di Resnet di Facebook

4.3 Addestramento su CIFAR

4.4 LeNet vs Resnet: confronto

Capitolo 5

Caso d'uso attuale: fine-tuning su dataset arbitrario

Un caso di studio attuale è quello di prendere una rete allo stato dell'arte, già addestrata per mesi su un dataset enorme come quello di *Imagenet* per utilizzarla su un dataset arbitrario a nostra scelta, per scopi industriali o personali.

5.1 Dataset

5.2 Fine-Tuning

5.3 ModelZoo

5.3.1 Fine-tuning su Resnet

Facebook ha reso pubbliche le sue implementazioni di Resnet su github a tutti, cosicché possano essere utilizzate per qualsivoglia esperimento. Ci sono vari modelli, più o meno potenti a seconda degli strati della rete (da 18 a 140).

Capitolo 6

Conclusioni

Bibliografia

[1] Wikipedia. (2016). Connessionismo Wikipedia, L'enciclopedia libera, indirizzo: http://it.wikipedia.org/w/index.php?title=Connessionismo&oldid=82972690 (visitato il 17/08/2017).

[2] ——, (2016). Funzione gradino di Heaviside, indirizzo: https://it.wikipedia.org/wiki/Funzione_gradino_di_Heaviside (visitato il 20/08/2017).

[3] ——, (2016). Percettrone, indirizzo: http://it.wikipedia.org/w/index.php?title=Percettrone&oldid=88722444 (visitato il 20/08/2017).

[4] ——, (2016). Universal Approximation Theorem, indirizzo: https://en.wikipedia.org/w/index.php?title=Universal_approximation_theorem&oldid=796183757 (visitato il 20/08/2017).

[5] (2016). Gradisca New York, indirizzo: www.gradiscanyc.com (visitato il 20/08/2017).

[6] A. Central. (2016). The Average Profit Margin for a Restaurant, indirizzo: http://yourbusiness.azcentral.com/average-profit-margin-restaurant-13113.html (visitato il 20/08/2017).

[7] R. R. Group. (2016). The Restaurant Financial Red Flags, indirizzo: http://rrgconsulting.com/ten_restaurant_financial_red_flags.htm (visitato il 20/08/2017).