Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

Machine Learning: predire le scelte del consumatore

Tesi di Laurea in Statistica Numerica

Relatore:
Chiar.ma Prof.ssa
ELENA LOLI
PICCOLOMINI

Presentata da:
SILVESTRI MARCO

I Sessione
Anno Accademico 2017/2018

Alla mia famiglia.

Parole chiave

Machine Learning

TensorFlow

Reti Neurali

Classificare scelte consumatore

"I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted. "

Alan Mathison Turing

Scritta in LaTeX

Abstract

La capacità di predire il comportamento dei consumatori è fondamentale per lo studio di un settore. Io ho utilizzato le competenze acquisite nel mio corso di studi per sfruttare il machine learning sotto lo sguardo economico e del marketing. Nell'ambito dell'apprendimento automatico esiste un'ampia gamma di modelli predittivi che sono applicabili al marketing e ognuno presenta particolari vantaggi e svantaggi. Il mio obiettivo è quello innanzitutto di fornire una visione completa del machine learning, del suo utilizzo e dei suoi metodi principali. Affronterò le reti neurali, utilizzate nella mia applicazione, per dare una completa visione dell'utilizzo reale di questi metodi. Presenterò un modello di applicazione che permetterà al lettore di andare a capire tutti questi concetti per poi essere in grado di applicarli semplificando il lavoro di ogni azienda operante nel settore.

Contents

List of Figures

Introduzione

Il consumatore, secondo l'articolo 3 del codice del consumo, è la persona fisica che agisce per scopi estranei rispetto all'attività imprenditoriale commerciale, artigianale o professionale eventualmente svolta. Esso rappresenta il focus e l'obiettivo di questa tesi: la capacità di offrire un prodotto adeguato ad un consumatore sarà influenzata dall'intervento delle tecniche di predizione delle sue scelte. Affronteremo questa correlazione ponendo particolare attenzione al ruolo che il machine learning ha in tutto questo. La raccolta dei dati opportuni e il corretto utilizzo di questi, permette di proporre uno studio molto dettagliato influenzando la vita di tutti i giorni delle aziende operanti nei settori che citeremo. Iniziamo introducendo l'obiettivo fondamentale su cui viene sviluppata l'applicazione mostrata in seguito: il Machine Learning. Questa forma di statistica ci permetterà di andare a classificare i nostri consumatori in insiemi differenziati per interessi, riuscendo così a proporre prodotti ritenuti interessanti dai nostri clienti.

Nel primo capitolo di questa tesi sono stati introdotti i concetti fondamentali del Machine Learning, Deep Learning e i metodi principali di elaborazione dei dati. Questo viene fatto ponendo molta attenzione alle reti neurali.

Nel secondo capitolo, viene analizzata la raccolta dei dati e delle infor-

mazioni. Sono mostrati i modi migliori esistenti per raccogliere i dati e costruire il database. Questi devono essere raccolti opportunamente per poi essere rielaborati affinchè il database su cui la rete neurale si istruisce sia corretto.

Nel terzo capitolo viene mostrata l'applicazione TensorFlow realizzata: in questa parte è fondamentale l'utilizzo dei concetti assimilati in precedenza sulle reti neurali. Il funzionamento di base dell'applicazione prevede prima la creazione di un database partendo da un file in formato ".json". Un file Python leggerà questo formato e creerà un database ".db". Su questa struttura dati andrò ad interfacciare il file contenenti le istruzioni di creazione della mia rete neurale. Questa verrà istruita e creata indicando il numero di layer e la funzione di attivazione scelta. Il centro della mia applicazione sarà un ulteriore file ".python" che gestirà l'interazione dall'esterno con un utente e le relative risposte alle domande che vengono poste. Strutturando l'applicazione in questo modo tutte le volte che viene interrogata la mia applicazione non viene istruita nuovamente la rete.

Nelle conclusioni mostriamo i risultati ottenuti e le tecniche adottate che si sono rivelate le migliori. Proponiamo inoltre una futura estensione di questa applicazione.

Attraverso tutto questo sistema si riesce a gestire l'offerta di prodotti per il consumatore. Questo è molto utile nella vita quotidiana delle aziende.

Chapter 1

Machine Learning

In questo capitolo viene affrontato il Machine Learning sotto molti aspetti. Partendo dalla definizione dello stesso si arriverà ad illustrare in modo dettagliato i principali metodi di elaborazione dei dati. Un ruolo fondamentale ha il Deep Learning: ci permetterà di capire e di elaborare molti risultati ottenuti in vari settori.

1.1 Analisi del Machine Learning

Il Machine Learning (ML) è una forma di statistica applicata mirata ad utilizzare i computer per stimare statisticamente una funzione complessa. Nel 1997 Mitchell fornì la seguente definizione di Machine Learning: "Un algoritmo apprende dall'esperienza E riguardanti una classe di problemi T con una misura pari a P, se la sua performance sui problemi T, misurata tramite P, aumenta con l'esperienza E". Possiamo quindi affermare che il ML è un insieme di tecniche che permettono alla macchine di "imparare" dai dati ed

in seguito prendere decisioni o fare una predizione su di essi. Un sistema di Machine Learning può essere applicato ad una base di "Conoscenza" proveniente da sorgenti multiple per risolvere molti e differenti compiti:

- classificazione facciale
- riconoscimento del parlato
- riconoscimento di oggetti
- ...

A differenza degli algoritmi euristici, ossia quegli algoritmi che seguono un insieme specifico di istruzioni per risolvere un dato problema, il Machine Learning abilita un computer ad apprendere (ad esempio riconoscere "configurazioni percettive" da solo e fare predizioni su di esse).

Il Machine Learning può essere adattato a tre differenti tipi di compiti:

- Classificazione
- Clustering
- Predizione

Molti degli algoritmi di Machine Learning possono essere divisi nelle due categorie di Apprendimento Supervisionato e Apprendimento Non Supervisionato a seconda di un fattore principale. Questo fattore è il fatto che l'insieme di addestramento sia supervisionato o non supervisionato.

1.2 Deep Learning: sottoinsieme del Machine Learning

Il Deep Learning è invece una sotto-area del Machine Learning che fa uso delle "Reti Neurali Profonde" (Deep Neural Network). Esse sono dotate di molti strati e di nuovi algoritmi per il pre-processamento dei dati per la regolarizzazione del modello:

- word embeddings
- dropout
- data-augmentation
- ...

Le Reti Neurali sono un modello dell'attività neuronale del cervello. Il Deep Learning trae ispirazione dalle Neuroscienze, utilizzando appunto queste reti neurali. A differenza del cervello biologico, dove qualsiasi neurone può connettersi a qualsiasi altro neurone sotto alcuni vincoli fisici, le Reti Neurali Artificiali (ANN) hanno un numero finito di strati e connessioni. Hanno infine anche una direzione prestabilita della propagazione dell'informazione.

Il principale problema è il loro costo computazionale.

Tra il 2006 e il 2012, il gruppo di ricerca guidato da Geoffrey Hinton dell'Università di Toronto è stato in grado finalmente di parallelizzare gli algoritmi per le ANN su architetture parallele. Il principale risultato è stato un notevole incremento del numero di strati, neuroni e parametri del modello in generale (anche oltre i 10 milioni di parametri) permettendo alle macchine

di computare una quantità massiccia di dati addestrandosi su di essi. Pertanto, il primo requisito per l'addestramento di un modello di Deep learning è avere a disposizione train-set molto grandi. Questo rende il Deep Learning molto adatto ad affrontare l'era dei Big Data (questi sono grandi moli di dati, provenienti da sorgenti eterogenee e difficili da gestire ed analizzare utilizzando strumenti tradizionali).

Possiamo quindi affermare che il Deep Learning è uno degli approcci all'apprendimento automatico che ha preso spunto dalla struttura del cervello, ovvero dall'interconnessione dei vari neuroni. Altri approcci includono la programmazione logica induttiva, il clustering e le reti bayesiane. Queste ultime sono basate su modelli DAG (grafo aciclico diretto) costituiti da un insieme di variabili e dalle loro dipendenze condizionali.

1.3 Utilizzi del Deep Learning

Il Deep learning ha influenzato le applicazioni industriali come mai era successo prima al Machine Learning. Infatti esso è in grado di trattare un enorme quantità di dati (milioni di immagini per esempio) e riconoscere alcune caratteristiche discriminative. Le ricerche che il Deep Learning è in grado di affrontare sono basate su:

- testo
- l'individuazione di frodi o spam
- il riconoscimento delle scritte
- la ricerca delle immagini

- il riconoscimento del parlato
- i recommendation system
- la Street View detection
- la traduzione di lingue
- e molti altri compiti ...

In Google, le reti deep hanno già rimpiazzato decine di "sistemi a regole". Oggi il Deep Learning per la Computer Vision già mostra di avere capacità super-umane, e che variano dal riconoscimento di figure comuni come cani e gatti fino all'individuazione di noduli cancerosi in immagini tomografiche polmonari.

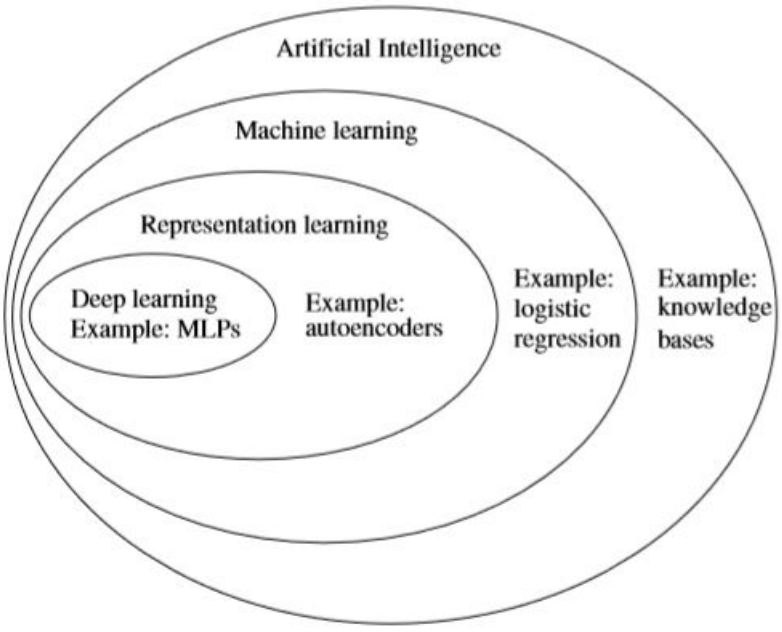


Figure 1.1: Grafico con differenze tra ML e DL

1.4 Metodi principali di elaborazione dei dati

Andiamo ad analizzare i metodi principali con cui riusciamo ad elaborare i dati. Cominciando dalla regressione semplice, andremo a parlare di algoritmi complessi, come l'algoritmo "K-means".

1.4.1 Regressione Semplice

Analizziamo la prima tecnica di studio della correlazione di due variabili. Questo tipo di regressione è associato alla correlazione di tipo lineare.

La Regressione Lineare è una inferenza statistica che studia la relazione tra due differenti variabili. Una variabile, che possiamo assumere sia "X", è detta "variabile indipendente". Mentre l'altra, che assumiamo possa essere "Y", è detta "variabile dipendente aleatoria". Quest'ultima variabile è di tipo casuale, ed è quindi determinata da una certa casualità. Mentre invece la relazione tra le due variabili non è casuale ma è legata in modo preciso. E' importante occuparsi di come la mia variabile aleatoria y abbia una relazione (causata quindi da casualità) di tipo f(x). Avrò quindi un legame descrittto sotto forma di funzione, e lo possiamo esprimere come

$$y = f(x) \tag{1.1}$$

Possiamo anche esprimerla sotto forma di retta:

$$y_i = \beta_0 + \beta_1 * x_i + \epsilon_i \tag{1.2}$$

in cui abbiamo quei coefficenti e quell'errore. Quindi la mia variabile aleatoria

conterrà sempre un errore poichè è approssimata con una funzione.

E se dovessimo rispondere alla domanda "ma a cosa serve questo modello?" la risposta quale sarebbe? Possiamo rispondere con una semplice ma chiara e corretta frase: "serve per poter predire dei valori della x in cui la y non è misurata". Quindi grazie a questa tecnica potrò avere sia delle predizioni (quindi stime della variabile dipendente aleatoria nei punti della X che non conosco) sia il valore della variabile aleatoria tra due punti di X che si conoscono.

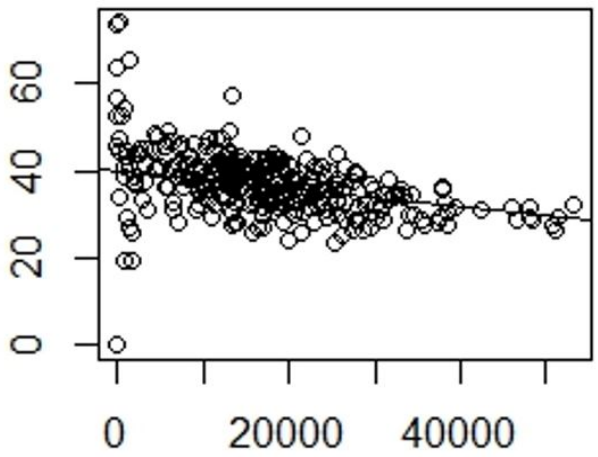


Figure 1.2: Esempio di regressione lineare in Rstudio

Approssimazione con Polinomi di grado superiore

Possiamo andare a fare una regressione approssimando il modello con un polinomio di grado maggiore di 1. La differenza principale si trova nella stima che andiamo a fare. L'equazione del modello sarà composta da un polinomio con grado maggiore di 2. Mostriamo ora l'equazione nel caso di

polinomio uguale a 2:

$$y_i = \beta_0 + \beta_1 * x_i + \beta_2 * x_i^2 + \epsilon_i \tag{1.3}$$

Abbiamo così definito la regressione quadratica, che avrà di conseguenza un coefficente in più e un termine di grado superiore. Possiamo definire anche l'equazione della regressione cubica:

$$y_i = \beta_0 + \beta_1 * x_i + \beta_2 * x_i^2 + \beta_3 * x_i^3 + \epsilon_i \tag{1.4}$$

in cui ho aggiunto un ulteriore coefficente e un termine di grado superiore. E' difficile che si salga ulteriormente di grado: più si sale e più parametri dobbiamo andare a stimare. Dopo il quarto grado non si riesce più a migliorare l'approssimazione: le funzioni diventano molto oscillanti e quindi è difficile che rappresentino bene il mio set di dati.

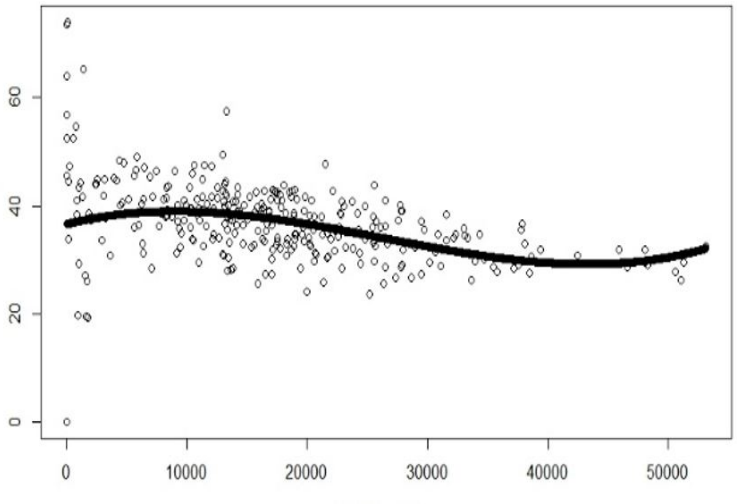


Figure 1.3: Esempio di regressione lineare con polinomi di grado superiore in Rstudio

1.4.2 Regressione Multivariata

Un altro metodo molto utilizzato è la regressione multivariata. Nella regressione lineare semplice, abbiamo immaginato che una certa variabile Y dipendesse dall'andamento di un'altra variabile X, in maniera lineare con andamento crescente o decrescente. Il modello visto prima però non teneva conto dell'esistenza di più influenze, e quindi non analizzava la correlazione che c'era tra di queste.

A differenza della regressione semplice, quella multivariata è creata in modo differente. Nella sua forma più semplice, un modello lineare specifica la relazione (lineare) tra una variabile dipendente (risposta) y, ed un insieme di predittori x, cosicché:

$$y = b0 + b_1X_1 + b_2X_2 + b_3X_3 + ...b_nX_n \tag{1.5}$$

In questa equazione b0 è il coefficiente di regressione per l'intercetta ed i valori bi sono i coefficienti di regressione (per le variabili da 1 a n) calcolati dai dati. Ovviamente bisogna aggiungere all'equazione sopra citata anche l'errore che si crea con questo tipo di regressione. Possiamo quindi esprimerla in questo modo:

$$y = b0 + b_1X_1 + b_2X_2 + b_3X_3 + ...b_nX_n + \epsilon \tag{1.6}$$

Il funzionamento principalmente non differisce molto da quello della regressione lineare semplice. Esistono metodi per stimare i parametri e l'errore. Infatti l'obiettivo di questi metodi è proprio quello di ridurre i costi, per permettere una stima sempre migliore. Devo infatti trovare il metodo migliore

per minimizzare gli errori per la mia funzione, in modo da ottenere così i pesi ottimali per ogni variabile. Questo è anche il fine ultimo del Machine Learning, ovvero quello di ridurre al minimo l'errore di una predizione, calcolato a partire da dati reali (in poche parole, un algoritmo che apprende con l'esperienza). Per diminuire il valore del costo di questa funzione, citiamo un algoritmo che si chiama "Discesa del Gradiente". Andiamo ora ad introdurre brevemente il suo funzionamento essendo una tecnica molto adatta a minimizzare gli errori. Si inizia assegnando un valore casuale ai pesi. Questi si continueranno a ricalcolare, secondo una formula, fino alla loro convergenza. In questo modo riusciamo a dare una stima migliore dei nostri pesi, e questa sarà sempre più realistica e corretta.

1.4.3 Alberi e Foreste

Gli alberi di classificazione rappresentano il primo insieme di metodi rivolti a migliorare la probabilità di un risultato. Altri metodi utilizzabili in questo modo sono le Reti di Bayes che vedremo dopo. Gli alberi di decisione (o alberi di classificazione) costituiscono il modo più semplice di classificare degli "oggetti" in un numero finito di classi. Mentre le foreste sono un metodo che impiega molte centinaia di alberi di classificazione: se sono casuali (per esempio) si scambiano in ciascun albero il contenuto.

Gli errori all'interno degli alberi

Molto importante è ridurre l'errore di questo algoritmo di Apprendimento Supervisionato quanto più possibile, al fine di arrivare ad un modello ac-

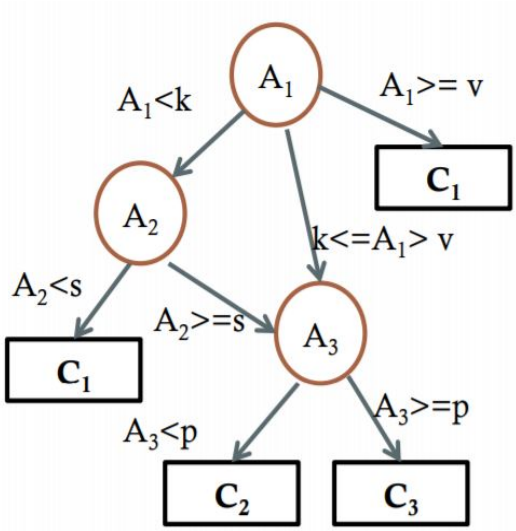


Figure 1.4: Un albero decisionale

cettabile. Possiamo utilizzare una metrica chiamata "Area sotto la Curva" (detta anche ROC). Questo tipo di metrica, usata per la prima volta nella seconda guerra mondiale con lo scopo di individuare i nemici tramite l'utilizzo di radar, altro non è che un grafico con l'insieme delle coppie: veri positivi in funzione dei falsi positivi. Al variare del valore della soglia che il classificatore utilizza per stabilire l'appartenenza (o meno) di un punto alla classe positiva tengo traccia dei veri positivi in funzione dei falsi. Quindi indicando con "p" e "n" rispettivamente l'appartenenza alla classe positiva e negativa, possiamo definire veri positivi e falsi positivi confrontando gli output del classificatore con quelli del dataset.

Potremmo ulteriormente costruire un grafico che rappresenti la relazione tra sensitività (cioè la percentuale del numero dei veri positivi, posti sull'asse y) e la specificità (la percentuale dei falsi positivi, posti sull'asse x). Costruendo questo grafico, avremo una curva, la cui area sottostante rappresenta il

valore della metrica (da cui il nome).

Importante è ricordare uno dei metodi ritenuto molto efficace per misurare l'effettiva capacità di un modello nel classificare un dato in input. Questo si chiama "convalida incrociata". Il principio è molto semplice: si prende una percentuale di dati a disposizione e si fa il training del modello con quello, poi si utilizzano i restanti per fare i test e misurare l'errore. Questa operazione si ripete un certo numero di volte, dove la differenza è che ad ogni nuovo passaggio si sceglie un set diverso di dati di training e dati di test.

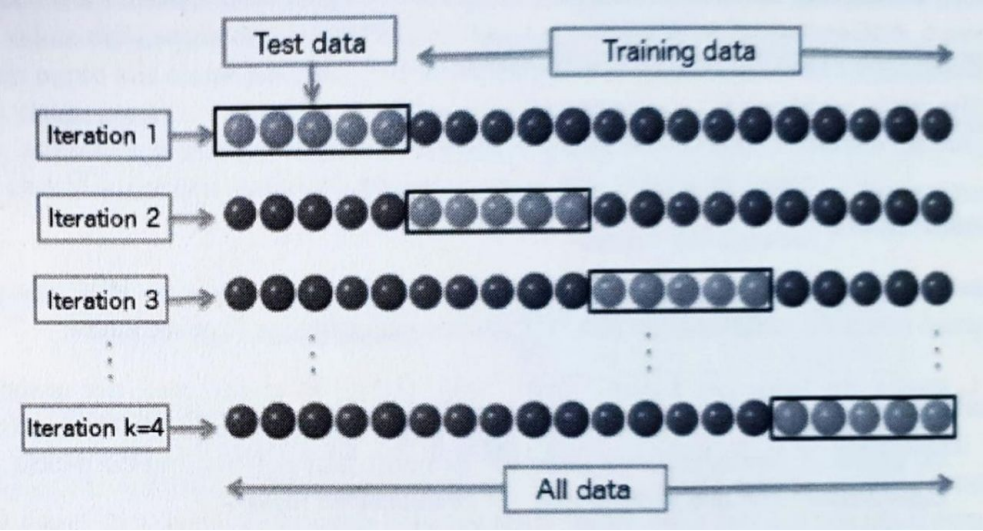


Figure 1.5: Esempio di regressione lineare con polinomi di grado superiore in Rstudio

Ultimo aspetto da affrontare è il Bias e la Varianza. Gli alberi decisionali hanno bias basso e varianza alta. Per comprendere il significato di questa frase bisogna sapere che il bias è nullo quando riusciamo a classificare correttamente ogni dato di input rispetto al proprio valore vero. Quindi il fatto che gli alberi decisionali abbiano per definizione un bias molto basso è una cosa positiva. La varianza alta indica che il modello in realtà sta cercando di

costruire una curva che passi da tutti i punti che rappresentano i dati che gli stiamo passando, affinchè impari da essi. Questo è un problema noto come overfitting, ed è una situazione da cui bisogna stare alla larga elaborando un modello di Machine Learning. Il contrario del fenomeno di overfitting è l'underfitting, dove invece il modello non impara assolutamente nulla dai dati.

1.4.4 Reti di Bayes

Iniziamo analizzando la probabilità condizionata, concetto fondamentale per affrontare le Reti di Bayes.

La probabilità è un numero da 0 a 1 (estremi inclusi) che indica il grado di certezza che un evento si verifichi (lanciando una moneta, l'evento è che esca ad esempio testa). Quando questa è 1, l'evento si verificherà sicuramente, quando è 0, non si verificherà mai. La probabilità condizionata è la probabilità che un evento si verifichi in base in base alla conoscenza che abbiamo riguardo altri fattori. In termini matematici, per esprimere la probabilità condizionata di un evento B se A è vero (p = 1) si indica con:

$$p(B|A) \tag{1.7}$$

Prima di procedere, è bene spiegare anche la probabilità congiunta. Questa indica la probabilità che due eventi A e B si verifichino insieme. Esempio: se ho una moneta, la probabilità che esca testa è 1 su 2:

$$\frac{1}{2} = 0.5 \tag{1.8}$$

Se ho due monete, la probabilità che esca testa per entrambe è 1 su 4:

$$\frac{1}{2} * \frac{1}{2} = \frac{1}{4} = 0.25 \tag{1.9}$$

In questo caso posso dire che:

$$p(A, B) = p(A) * p(B) \tag{1.10}$$

La formula che ho appena riportato funziona solo se A e B sono eventi indipendenti (il risultato del lancio di una moneta non influisce sul risultato del lancio dell'altra), che in termini matematici si esprime con la probabilità condizionata appena vista, ovvero equivale a dire:

$$p(B|A) = p(B) = 0.5 \tag{1.11}$$

Questo significa che la probabilità che si verifichi B sapendo che A si è verificato, è uguale alla probabilità che si verifichi B indipendentemente dal verificarsi di A. Pensando invece all'esempio della pioggia (se venissi a sapere che oggi piove, penserei che sia più probabile che piova anche domani):

$$p(B|A) > p(B) \tag{1.12}$$

In generale, la formula della probabilità congiunta sarà:

$$p(A, B) = p(A) * p(B|A) \tag{1.13}$$

Teorema di Bayes

Con le poche definizione appena date, possiamo dimostrare il Teorema di Bayes. Conoscendo la probabilità condizionata p(A—B) non abbiamo al momento un metodo veloce per calcolare p(B—A). La probabilità condizionata, infatti, non possiede la proprietà commutativa. A differenza di questa, come si può facilmente comprendere, la probabilità congiunta dispone di questa caratteristica, ovvero:

$$p(A, B) = p(B, A) \tag{1.14}$$

per eventi A e B qualsiasi. Dalla formula enunciata prima della probabilità condizionata, riscriviamo questa uguaglianza come:

$$p(A) * p(B|A) = p(B) * p(A|B) \tag{1.15}$$

A questo punto calcolare p(B—A) è semplice, basta dividere entrambi i membri per p(A):

$$p(B—A) = \frac{p(B) \ * \ p(A—B)}{p(A)} \tag{1.16}$$

Applicando questo teorema al nostro problema, abbiamo che:

$$argmax_j P(c_j|A) = argmax_j \frac{P(A|c_j) * P(c_j)}{P(A)} \tag{1.17}$$

poi semplificando il problema abbiamo che:

$$argmax_j P(c_j|A) = argmax_j P(A|c_j) * P(c_j) \tag{1.18}$$

Ma dato che il record A è composto di N Attributi:

$$A(x_1, x_2, \ldots x_N) \tag{1.19}$$

allora:

$$argmax_j P(A|c_j) * P(c_j) = argmax_j P(x_1, x_2, \ldots, x_N|c_j) * P(c_j) \tag{1.20}$$

Assumiamo inoltre che gli N attributi siano tutti indipendenti:

$$argmax_j P(x_1, x_2, ..., x_N|c_j)P(c_j) = argmax_j P(c_j) * \prod_{i=1}^{N} P(x_i|c_j) \tag{1.21}$$

Possiamo ora concludere che l'algoritmo di NB sceglie la classe cj che massimizza la quantità sopra citata a destra dell'equazione. Ma come stimo P(cj) e P(xi—cj)? Una possibile soluzione è quella di approssimare le probabilità come frequenze relative, rispetto ai valori del training set:

$$P(cj) = \frac{Numero\ di\ istanze\ classificate\ c_j}{Numero\ di\ istanze\ totali} \tag{1.22}$$

1.4.5 Classificatore K-Nearest Neighbour

Introduciamo ora il classificatore "K-Nearest Neighbour". Le istanze sono rappresentate come punti di uno spazio N dimensionale. Si calcola la distanza Euclidea tra i punti e questo permette di classificare nelle varie classi. Cioè, la classe di un punto viene approssimata alla classe dei K punti più vicini.

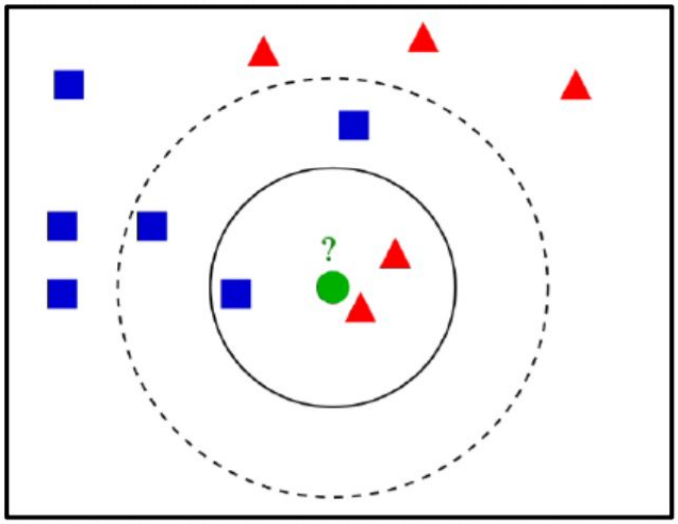


Figure 1.6: Grafico su K-Nearest Neighbour

1.4.6 Reti Neurali

Le reti neurali rappresentano un sistema di apprendimento supervised, che trae spunto dal corrispettivo biologico.

I circuiti neurali artificiali sono la base di sofisticate forme di intelligenza artificiale (sempre più evolute) in grado di apprendere sfruttando meccanismi simili (almeno in parte) a quelli dell'intelligenza umana. Come citato in precedenza, il prototipo delle reti neurali artificiali sono quelle biologiche.

Le reti neurali del cervello umano sono la sede della nostra capacità di comprendere l'ambiente e i suoi mutamenti, e di fornire quindi risposte adattive calibrate sulle esigenze che si presentano. Sono costituite da insiemi di cellule nervose fittamente interconnesse fra loro. Al loro interno troviamo:

1. i somi neuronali, ossia i corpi dei neuroni. Ricevono e processano le informazioni; in caso il potenziale di azione in ingresso superi un certo valore, generano a loro volta degli impulsi in grado di propagarsi nella

rete;

2. i neurotrasmettitori, composti biologici di diverse categorie (ammine, peptidi, aminoacidi), sintetizzati nei somi e responsabili della modulazione degli impulsi nervosi;

3. gli assoni o neuriti: la via di comunicazione in uscita da un neurone. Ogni cellula nervosa ne possiede di norma soltanto uno;

4. i dendriti: la principale via di comunicazione in ingresso. Sono inoltre multipli per ogni neurone, e formano il cosiddetto albero dendritico;

5. le sinapsi, o giunzioni sinaptiche: i siti funzionali ad alta specializzazione nei quali avviene il passaggio delle informazioni fra neuroni. Ognuno di questi ne possiede migliaia. A seconda dell'azione esercitata dai neurotrasmettitori, le sinapsi hanno una funzione eccitatoria (facilitando la trasmissione dell'impulso nervoso) oppure inibitoria (tendente a smorzarlo). Le connessioni hanno luogo quando il neurotrasmettitore viene rilasciato nello spazio intersinaptico, raggiungendo così i recettori delle membrane post-sinaptiche (ossia del neurone successivo), e, alterandone la permeabilità, trasmettendo l'impulso nervoso.

Un singolo neurone può ricevere simultaneamente segnali da diverse sinapsi. Una delle sue capacità più importante è quella di misurare il potenziale elettrico di tali segnali in modo globale. Facendo in questo modo si stabilisce quindi se è stata raggiunta la soglia di attivazione per generare a sua volta un impulso nervoso. Quindi al superare di una soglia, viene generato un impulso. Tale proprietà è implementata anche nelle reti artificiali.

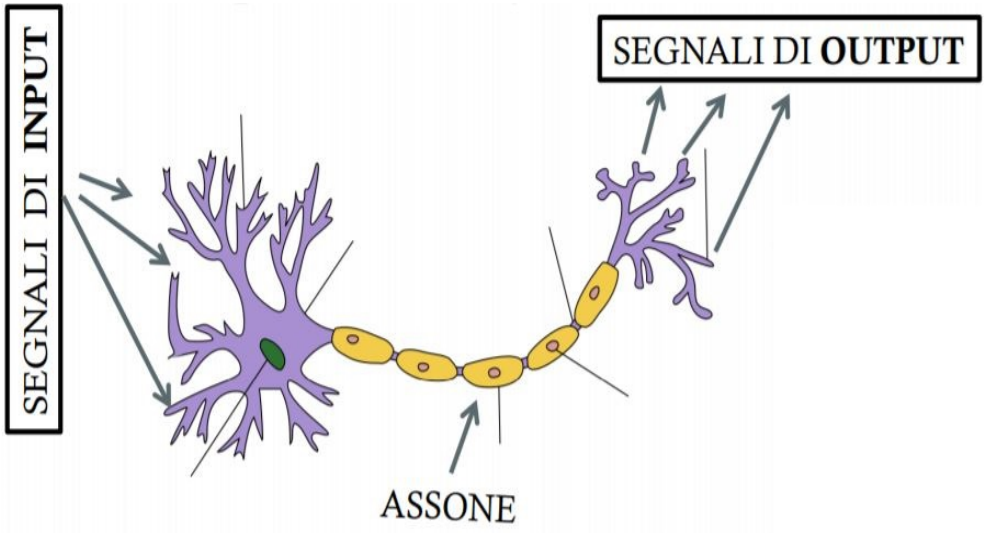


Figure 1.7: Composizione di cellule nervose

La configurazione sinaptica all'interno di ogni rete neurale biologica è dinamica. Si tratta di un fattore determinante per la loro efficienza. Il numero di sinapsi può incrementare o diminuire a seconda degli stimoli che riceve la rete, e la definiamo dinamica per questo motivo. Più sono numerosi, maggiori connessioni sinaptiche vengono create (e viceversa). In questo modo, la risposta adattiva fornita dai circuiti neurali è più calibrata, e anche questa è una peculiarità implementata nelle reti neurali artificiali.

Breve storia delle reti neurali artificiali

Nel 1943 viene proposto il primo modello teorico di un rudimentale neurone artificiale. A proporlo è una coppia di scienziati, McCulloch e Pitts. Da qua nasce il neurone "matematico" di McCulloch-Pitts.

I due scienziati descrivono un apparato in grado di ricevere n dati binari in ingresso in ognuno dei suoi elementi, a cui segue un singolo dato in uscita

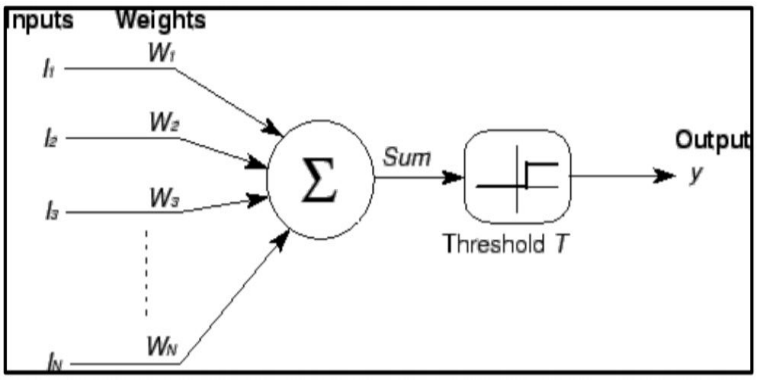


Figure 1.8: Il neurone matematico di McCulloch-Pitts

per ciascuno. Tale macchina è in grado di lavorare su funzioni booleane elementari, e solo su quelle.

Il canadese Donald Olding Hebb fu uno studioso originale della psicologia del '900, precursore di molte teorie e scoperte successive, e uno dei primi scienziati ad approfondire il legame tra il sistema nervoso e il comportamento umano. Nel 1949, lo psicologo Hebb ipotizza la possibilità di istruire le macchine con un apprendimento che emuli quello alla base dell'intelligenza umana.

Nel 1958 viene proposta da Rosenblatt la prima rete neurale: Perceptron. Queste sono le basi dell'apprendimento automatico.

Perceptron di Rosenblatt possiede uno strato di nodi (neuroni artificiali) di input e un nodo di output. I pesi sinaptici (un peso indica la forza di una connessione fra due nodi) sono dinamici, permettendo alla macchina di apprendere (in un modo sommariamente simile, anche se molto più elementare, a quello delle reti neurali biologiche). Inoltre, il modello è feedforward: gli impulsi si propagano in un'unica direzione: in avanti. E' giusto quindi ricordare che il suo campo di applicazione è molto limitato. E in cosa consiste questa

limitazione? Nel riconoscere forme, classificandole in due gruppi separati, e nel calcolare semplici funzioni.

Ovviamente procedendo si migliorerà questo primo modello. Infatti il passo successivo è il Perceptron multistrato (MLP). Al suo interno, fra i nodi di input e quello di output si trova uno strato hidden, dove avviene l'elaborazione delle informazioni provenienti dallo strato di input, che poi vengono inviate al nodo di output. È una rete feedforward non lineare: le connessioni in ingresso e in uscita da ogni singolo nodo sono multiple. A merito di tale architettura, il MLP può computare qualsiasi funzione.

Werbos, nel 1974, descrive nella sua tesi di dottorato come impostare l'apprendimento di un MLP.

Il suo lavoro viene poi ripreso e perfezionato da Rumelhart, Hinton e Williams, che nel 1986 elaborano il celebre Error Back-Propagation. Con l'algoritmo di retropropagazione dell'errore entriamo nel presente, essendo tuttora utilizzato. L'EBP permette di perfezionare in stadi successivi l'apprendimento automatico di una rete neurale. Si implementa modificando i pesi delle connessioni fra nodi che non producono l'output ottimale, finché non si ottiene quest'ultimo. Non meno importante, in tal senso, risulta il precedente lavoro di Hebb, relativo alle connessioni reciproche fra neuroni. Hebb postula che il loro peso deve incrementare unicamente in caso di convergenza fra i due valori pre e post-sinaptico. Con i MLP e l'EPB il machine learning trova ora (nel 1990) alcuni campi di applicazione pratica.

Intanto nel 1982 parallelemente vengono implementate anche reti neurali con architetture feedback dette reti Hopfield. Hopfield è il fisico che, proprio nel 1982, propose il modello. In tali architetture, le informazioni fra nodi

viaggiano in qualunque direzione: in avanti, all'indietro e fra nodi di una stessa fila. Il campo delle applicazioni si amplia ulteriormente, le limitazioni passate vengono superate e migliorate. Sempre nel 1982 Kohonen progetta un tipo di rete neurale dall'architettura sia feedforward che feedback. Sua caratteristica peculiare è la capacità di modificare la configurazione (mappa) dei propri nodi in base al peso che assumono man mano che vengono forniti gli input. I nodi con pesi simili si avvicinano, quelli con pesi molto diversi si allontanano. La rete di Kohonen è anche conosciuta come rete SOM (Self-Organizing Maps).

Un'altra rete neurale proposta fu quella di Elman nel 1990. Anche questo è un modello di rete ricorrente (bidirezionale), ma con la variante che alla classica struttura MLP viene aggiunto un gruppo di nodi aventi lo scopo di conservare le informazioni della precedente configurazione di valori della rete. Grazie a tale modifica, la rete di Elman si rivela vantaggiosa nel calcolo delle sequenze temporali.

L'evoluzione delle reti neurali prosegue con la recente tecnologia adottata da IBM, tramite la quale è stata sviluppata una rete neurale basata su materiali a cambiamento di fase. L'hardware di tale architettura utilizza leghe come il germanio tellururo di ammonio, che presentano l'interessante proprietà di assumere due diversi stati: cristallino (configurazione spaziale omogenea) e amorfo (configurazione spaziale poco definita). Nei neuroni a cambiamento di fase, gli impulsi elettrici sono in grado di provocare una cristallizzazione del materiale, innescandone infine il firing (attivazione). Ebbene, questo è analogo a quello che avviene nelle cellule nervose. Per ora il neurone artificiale di IBM permette di scrivervi informazioni ma non le memorizza sta-

bilmente, però è certo che il suo funzionamento è quanto di più simile esista all'emulazione di un cervello umano.

Algoritmo di Perceptron nelle reti neurali

Vediamo l'algoritmo di Perceptron che ho citato sopra. Dato un insieme di valori in input:

$$(x1, x2, \ldots, xm) \tag{1.23}$$

e un insieme di coefficienti (pesi):

$$(w1, w2, \ldots, wm) \tag{1.24}$$

Considero un "sommatore" colui che combina valori di input e pesi sulle varie sinapsi:

$$z = w1 * x1 + w2 * x2 + \ldots wm * xm \tag{1.25}$$

Mi serve inoltre una funzione di attivazione che decide l'output:

$$\phi = \begin{cases} 1 & if\ z > \theta \\ -1 & if\ z \leq \theta \end{cases} \tag{1.26}$$

Quindi, l'insieme dei valori in input sono le features dei dati da classificare. La funzione di output restituisce due possibili valori (classificatore binario):

- -1 se i dati sono appartenenti alla Classe 0
- 1 se i dati sono appartenenti alla Classe 1

Come settare il valore dei pesi? Innanzitutto inizializzo i pesi a 0 o a numeri

casuali piccoli. Per ogni campione X(i) del training set:

$$X^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, ...x_m^{(i)}) \; di \; classe \; Z \tag{1.27}$$

Calcolo il valore previsto di output Zp e lo confronto con il valore reale Z. Aggiorno i pesi di conseguenza:

$$w_j = w_j + v * (Z - Z_P) * x_j^{(i)} \tag{1.28}$$

L'algoritmo termina quando:

- ho raggiunto un numero massimo di iterazioni
- oppure l'errore di aggiornamento al passo j (E(j)) è diventato inferiore ad una soglia prefissata:

$$E(j) = \sum_{i=1}^{m} v(Z - Z^P)x_j^{(i)} \tag{1.29}$$

L'algoritmo di Perceptron è in grado di classificare correttamente le istanze nel caso di separabilità lineare. Nel caso di non separabilità lineare sono interessanti le reti neurali multi-livello.

Pregi delle reti neurali

Le reti neurali, come abbiamo già detto, lavorano in parallelo e quindi sono in grado di trattare molti dati. E' un sistema dotato di una buona immunità al rumore; se alcune unità del sistema dovessero funzionare male, la rete nel suo complesso avrebbe delle riduzioni di prestazioni ma difficilmente andrebbe

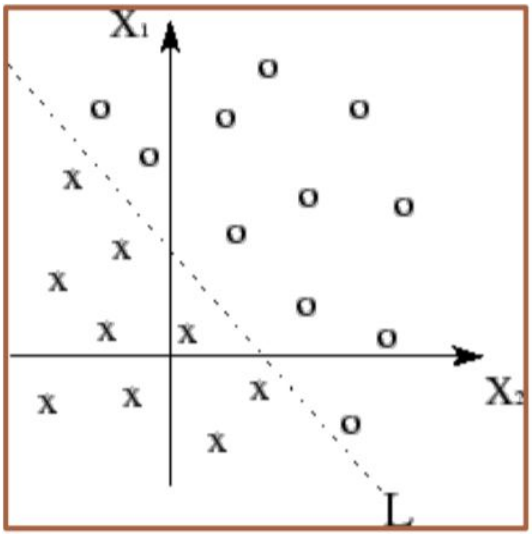


Figure 1.9: La separabilità lineare: interessante quando analizziamo l'algoritmo di Perceptron

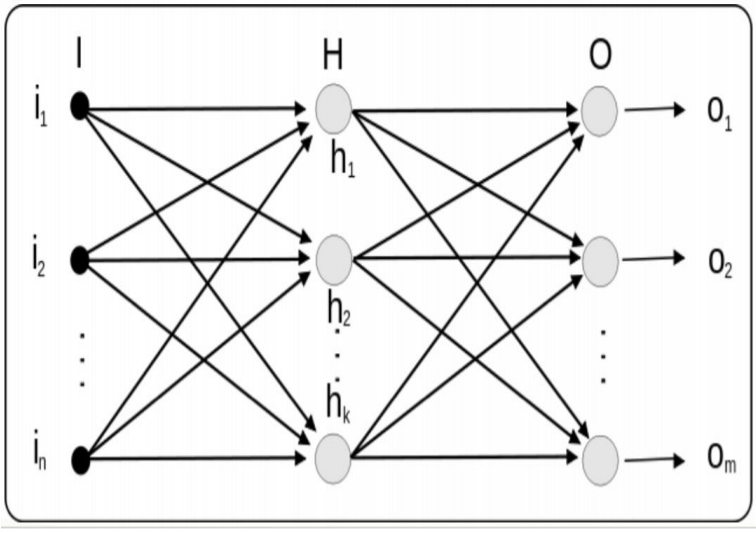


Figure 1.10: Le reti neurali multi-livello

incontro ad un blocco del sistema. I dati raccolti devono saper essere elaborati nel modo corretto: non bisogna andare incontro a conclusioni affrettate senza prima aver elaborato bene i risultati.

Difetti delle reti neurali

Come la definizione inglese delle reti neurali ("blackbox") ci suggerisce, i modelli ottenuti non sono spiegabili in linguaggio simbolico umano: i risultati vanno accettati "così come sono". Come per qualsiasi algoritmo di modellazione, anche le reti neurali sono efficienti solo se le variabili predittive sono scelte con cura. Una limitazione di queste reti è il fatto di non essere in grado di trattare in modo efficiente variabili di tipo categorico (per esempio, il nome della città) con molti valori diversi. Necessitano di una fase di addestramento del sistema (fondamentale) che fissi i pesi dei singoli neuroni. Per questo motivo bisogna stare molto attenti quando si utilizzano modelli come questi. Questa fase può richiedere molto tempo, se il numero dei record e delle variabili analizzate è molto grande. Non esistono teoremi o modelli che permettano di definire la rete ottima.

Chapter 2

Raccolta di dati e informazioni

Ora affrontiamo un problema sempre molto importante quando si fanno analisi di questo tipo: la raccolta dei dati. E' molto importante raccogliere i dati nel modo più corretto possibile. Citiamo ora alcuni dei metodi di raccolta dei dati più utilizzati.

Questa parte ci servirà sia per raccogliere dati su come rispondere a eventuali interessi, sia nel raccogliere gli interessi maggiori nei consumatori. In pratica, questa parte è fondamentale nella costruzione delle frasi che compongono il database della mia applicazione descritta nel capitolo seguente.

2.1 Metodi Trade-Off

Trattiamo ora dei metodi che permettono di raccogliere dati statistici dai consumatori.

2.1.1 Analisi Congiunta

Iniziamo con un'analisi molto complessa: per capirla fingiamo che dobbiamo raccogliere delle considerazioni dei consumatori su un prodotto. Nel nostro caso ci interessa come un consumatore ci risponde riguardo ad un proprio interesse e per fare questo possiamo proporgli un prodotto. Qua di seguito ci viene mostrata la teoria corretta da seguire.

L'analisi congiunta, nel suo formato tradizionale Full Profile, considera un prodotto alla volta. Pertanto, ha senso considerare l'analisi congiunta quando non esiste un vero contesto concorrenziale per il prodotto. Alternativamente, se sul sito web avete a disposizione alcune offerte davvero speciali, le persone potrebbero decidere senza eseguire ulteriori confronti. L'analisi congiunta opera al meglio rappresentando la migliore alternativa per questo tipo di applicazione. Inoltre può essere molto efficace nell'ottimizzazione di ogni tipo di messaggio, determinando la migliore combinazione di elementi da includere. E' bene considerare l'analisi congiunta anche quando il vostro prodotto viene scelto solo poco frequentemente. Poiché la modellazione a scelte discrete impiega delle scelte, può non riuscire a individuare le motivazioni di una decisione se il brand in questione viene scelto solo raramente. Potreste anche pensare di restringere l'ampiezza della concorrenza in modo da ottenere una migliore risposta (se il vostro brand viene scelto raramente da un più ampio ventaglio).

Quando (e perchè) usare l'analisi congiunta?

L'analisi congiunta può fornire conoscenze preziose in situazioni in cui il prodotto avrebbe una quota minima e, potrebbe soccombere completamente rispetto alle altre scelte in caso di scelta. In pratica, se un prodotto viene scelto raramente, non vi sarebbero informazioni sufficienti dalle poche volte che verrebbe selezionato per determinare che cosa determina tale scelta. L'analisi congiunta opera particolarmente bene nell'ottimizzazione dei messaggi, in particolare delle pubblicità stampate e dei siti web. Inoltre può essere particolarmente efficace nel determinare gli specifici livelli di servizio che devono essere offerti nelle relazioni complesse con i clienti, per esempio fra un'utility e i clienti commerciali, o fra una casa farmaceutica e i medici e le cliniche servite. Dipende tutto da dove e come vogliamo raccogliere informazioni per la nostra applicazione.

Alternativa all'analisi congiunta: Choice-Based Conjoint(CBC)

Il suo nome è accurato: prende qualcosa dall'analisi congiunta e qualcosa dalla modellazione a scelte discrete (che affronteremo dopo). In pratica offre una scelta fra tre varianti di un prodotto. Tutte le scelte non hanno sempre gli stessi attributi. Il fatto che le scelte condividano gli stessi attributi è proprio la parte di "analisi congiunta" dell'approccio CBC. Il programma CBC offre a pagamento un'opzione "avanzata" che consente di avvicinarsi agli schemi sperimentali che si impiegherebbero con la modellazione standard a scelte discrete. Ma non vi è alcun requisito nel suo uso. E in effetti abbiamo incontrato un buon numero di studi in cui nessuna delle persone coinvolte

aveva la minima idea che si potessero definire degli attributi così specifici per le scelte. Se confrontiamo il metodo CBC con l'analisi congiunta full-profile, a vantaggio del CBC vi è il fatto che impone una scelta. Questo assetto è più simile a ciò che le persone svolgono ogni giorno rispetto alle valutazioni richieste dall'analisi congiunta tradizionale. Per contro, CBC chiede alle persone di considerare molte più informazioni rispetto all'analisi congiunta. In pratica, ogni schermata mostrata a una persona con CBC potrebbe contenere tre o quattro profili di prodotti, uno in fianco all'altro. L'analisi congiunta full-profile presenta un solo profilo. Quindi nel nostro caso è utile quando vogliamo mettergli a disposizione più prodotti legati a interessi diversi o uguali, il consumatore sceglierà quello che preferisce.

2.1.2 Q-Sort e Case 5

Q-Sort e Case 5 sono i più "antichi" di tutti i metodi che trattiamo in queste pagine. I lavori preliminari si devono a Thurstone, figura molto influente nella psicologia. Negli anni Venti ha sviluppato un metodo per trasformare le valutazioni su vari elementi in dati posti lungo una scala relativa. Dal momento che eravamo agli arbori del trattamento dei dati, egli chiamò la sua procedura con il nome di "legge delle valutazioni comparative". La procedura per porre in ordine gli elementi si chiama Q-Sort. In realtà Q-Sort ha seguito una direzione tutta sua, che a volte la fa somigliare più a un sistema fideistico ben predisposto piuttosto che a un vero metodo scientifico. Usiamo solo la prima parte di Q-Sort: una valutazione parziale e guidata di un lungo elenco di elementi. Con valutazione parziale intendiamo il fatto che vengono valutati

solo gli elementi più o meno desiderabili, mentre gli altri rimangono relegati nella parte centrale. Poniamo noi domande come (per esempio) "inserisci i tre tuoi interessi preferiti" e poi chiediamo di ordinarli per preferenza. Possiamo fare inoltre ulteriori domande su questi stessi dati. Ora siamo in grado di affermare che questo è un metodo molto razionale per affrontare il problema. Le persone vi diranno con facilità che cosa preferiscono e che cosa detestano, anche osservando un lungo elenco.

L'analisi con il metodo detto Case 5 di Thurstone, produce una precisa gerarchia dei livelli di importanza, molto simile a quella prodotta da MaxDiff che vedremo dopo. Case 5 parte con una matrice win-loss, che registra quante volte ciascun elemento è stato valutato in maniera superiore rispetto a ciascun altro. Ne emerge una tabella dei livelli relativi di importanza, come in MaxDiff.

2.1.3 MaxDiff

Maximum difference scaling è sia il nome di un software sia il nome di una nota procedura statistica che è piuttosto differente. In questo caso ci occupiamo del software. Questo programma permette di confrontare degli elementi, chiedendo alle persone di compiere una scelta. Suddivide poi, in vari insiemi, un elenco composto da un numero di elementi compreso tra 8 e 34. Gli insiemi possono contenere da 2 a 6 elementi da confrontare contemporaneamente. Si tratta di un "derivato" della procedura Case 5, di cui abbiamo appena parlato. MaxDiff propone infatti un confronto tra elementi e chiede quale sia il migliore o il peggiore tra tre elementi: i test hanno dimostrato

che confrontando appunto tre elementi per volta si procede più rapidamente.

Questo metodo utilizza un tipo particolare di struttura sperimentale, che equilibra la frequenza con cui ciascun elemento dell'elenco compare nel confronto con ciascun altro elemento. Aumentando il numero degli elementi, aumentano anche le dimensioni della struttura. Confrontando tre elementi per volta, i partecipanti possono svolgere tre scelte per quattro elementi sottoposti a test. Le risposte vanno a formare la scala di importanza dei vari attributi, i quali sono differenziati molto più chiaramente rispetto a quanto avviene nelle classiche scale di valutazione. Questo metodo è utile per valutare gli elementi che potrebbero essere offerti in alternativa gli uni rispetto agli altri. MaxDiff, Q-Sort e Case 5 non hanno lo scopo di mostrare in quale modo interi prodotti o servizi sarebbero accettati sul mercato. E infatti non è quello che ci serve a noi. Sono infatti metodi molto utili perchè forniscono elenchi ben differenziati e utili per la soluzione dei problemi.

2.2 Analisi HB

L'analisi HB (Hierarchical Bayesan) ha dato origine al mondo della modellazione a scelte discrete. Si tratta di un metodo di calcolo estremamente complesso, che darà molto da lavorare al computer. Tale analisi consente di raddoppiare o anche triplicare la quantità di informazioni che potete trarre da ciascun partecipante allo studio. Questa teoria, sviluppata negli anni Novanta, è un vero e proprio metodo di Machine Learning che estende quanto possiamo misurare con gli studi trade-off, potendo contare su un immenso numero di calcoli e alcuni concetti davvero complicati. In pratica, campiona

gli altri rispondenti e memorizza i valori forniti da coloro che hanno dato le informazioni necessarie. In genere la ripetizione viene eseguita migliaia di volte (anche fino a venti o più) per ciascun livello di attributo e per ciascun rispondente, conservando una media aggiornata delle stime. Quindi grazie a questo metodo riusciamo (tramite l'ausilio del Machine Learning) a migliorare i dati raccolti sottoponendoli ad un numero limitato di campioni.

2.2.1 Modellazione a scelte discrete

La modellazione a scelte discrete riesce a determinare quanto le variazioni nelle caratteristiche dei prodotti o dei servizi influenzeranno le scelte dei consumatori. Se predisposto e analizzato correttamente, ha una sorprendente capacità di predire come i nuovi prodotti o servizi (o una loro modifica) si comporteranno nel mercato vero e proprio. Questo metodo è quello che ci interessa meno: noi non dobbiamo proporre nessun nuovo prodotto o servizio, ma raccogliere gli interessi e le preferenze degli utenti nel modo migliore possibile. Ho preferito counque inserirlo per completezza. Se in futuro si vuole anche pubblicizzare un prodotto con la rete neurale creata, si può già fare un'analisi con modellazione a scelte discrete e elaborare la strategia migliore decidendo a quale gruppo di consumatori offrire questo prodotto (ovviamente quest'ultimo passagio si fa con la rete neurale che ho creato io).

Chapter 3

Applicazione TensorFlow

Ho creato un'applicazione utilizzando la Libreria software open source TensorFlow. Essa fornisce moduli testati ed ottimizzati utili nella realizzazione di algoritmi per diversi tipi di compiti percettivi e di comprensione del linguaggio di Tensorflow. Con questo software, si riesce a progettare una rete neurale in modo chiaro e immediato. Per il momento, il sistema funziona inserendo un proprio interesse. Il programma andrà a lavorare sulla rete creata precedentemente e classificherà la risposta per rispondere correttamente offrendo un prodotto adeguato.

3.1 Obiettivo dell'applicazione

Molto spesso, nella vita di tutti i giorni, ci accorgiamo che manca qualcosa che ci potrebbe essere molto utile. Spesso utilizziamo lo smartphone o il computer per cercare e trovare prodotti interessanti che soddisfino le nostre esigenze, ma non sempre rimaniamo soddisfatti dalle nostre ricerche. Questa

applicazione permette di fare il contrario, sarà lei che in base agli interessi del consumatore sarà in grado di offrire il prodotto ritenuto più adatto a lui. L'obiettivo di questa applicazione è infatti quello di realizzare un programma che abbia la capacità di classificare i consumatori in cluster. Questi gruppi permetterebbero di proporre prodotti mirati a consumatori direttamente coinvolti nel settore. Il prodotto si rileverebbe molto interessante se suggerito al cliente corretto nel momento corretto.

Questo problema è stato affrontato sotto l'ottica del Machine Learning. L'applicazione creata realizzerà tutto questo, grazie al sistema di frasi creato. L'utente dovrà inserire parole, e utilizzando la rete neurale Tensorflow è capace di classificarlo in un insieme e proporgli un prodotto che potrebbe essere interessante per lui.

3.2 Utilizzo di TensorFlow e di Python

TensorFlow si basa sull'utilizzo di Python, linguaggio di programmazione ad alto livello orientato agli oggetti. La versione di Python utilizzata è stata la "3.5.0" mentre TensorFlow l'ho installato utilizzando il comando "pip". Il sistema Operativo utilizzato è "Windows 10".

Nel codice scritto, ho principalmente utilizzato TFLearn: una libreria di Deep Learning con API di livello superiore per TensorFlow. Le funzionalità di TFLearn includono:

- Facilità d'uso e di comprensione (API di alto livello per l'implementazione di reti neurali profonde) e presenza di tutorial ed esempi.

- Prototipazione rapida tramite strati di rete neurale incorporati altamente modulari, regolarizzatori, ottimizzatori, metriche ...
- Piena trasparenza su TensorFlow. Tutte le funzioni sono costruite su tensori e possono essere utilizzate indipendentemente da TFLearn.
- Potenti funzioni di supporto per allenare qualsiasi grafico TensorFlow, con supporto di più ingressi, uscite e ottimizzatori.
- Visualizzazione grafica facile e bella, con dettagli su pesi, gradienti, attivazioni e altro ...
- Posizionamento senza sforzo del dispositivo per l'utilizzo di più CPU / GPU.

TFLearn permette di settare il numero di layer attraverso la riga di codice:

```
rete = fully_connected(rete, 8)
```

e ripeto questa riga quanti layer voglio creare. In quel metodo "8" è il numero di unità presenti nel layer e la "rete" deve essere costruita a partire da quest'altra riga di codice:

```
rete = input_data(shape=[None, len(X[0])])
```

Andremo dopo ad analizzare gli input dati X e Y. Infine si andrà a realizzare e ad utilizzare una funzione di attivazione, per poi eseguire la regressione (e quindi la realizzazione della rete):

```
rete = fully_connected(rete, len(y[0]), activation='softmax')
rete = regression(rete)
```

Qua ho utilizzato la funzione "softmax", ma ne esistono varie: "Linear", "Sigmoid", "Softmax", "Softplus", "ReLU" e molte altre. Dopo giustificherò le mie scelte con opportuna spiegazione. Come ultima cosa andrò a fare il training della mia rete con la seguente istruzione:

```
model = DNN(rete, tensorboard_dir='logs')
model.fit(X, y, n_epoch=1000, batch_size=8, show_metric=True)
```

3.3 Struttura dell'applicazione

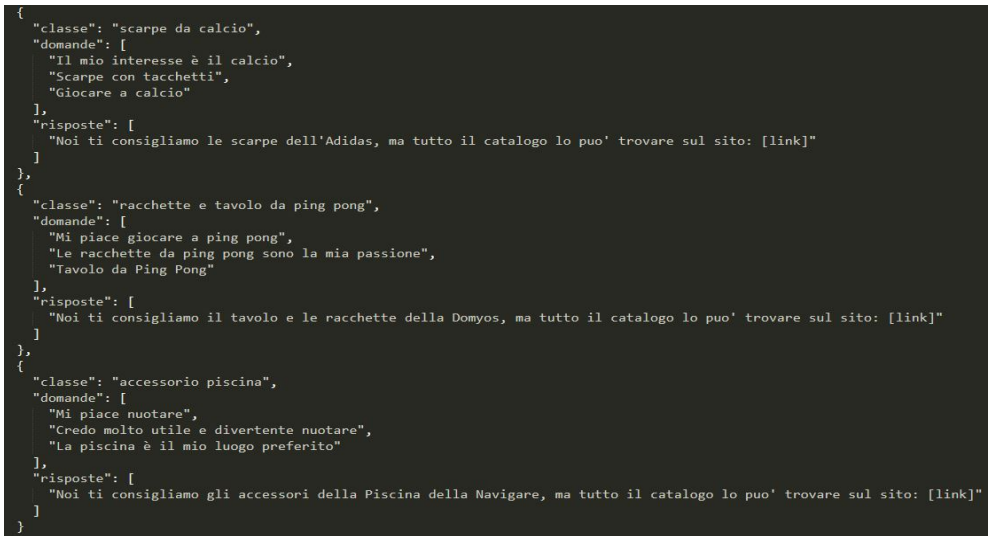
L'applicazione creata è composta da tre file:

- Il primo lo utilizzo per creare un database;
- Il secondo per creare la rete ed istruirla;
- Il terzo per andare a classificare eventuali frasi dell'utente per poi proporgli eventuali prodotti.

Ho realizzato inoltre un quarto file che mi è servito per i test di cui parlerò dopo. Il primo file realizza un database (in formato ".db") partendo da un file ".json" strutturato appunto nel formato "json".

Il secondo file è quello che crea e istruisce la rete neurale. Questo è strutturato in questo ordine procedurale:

- Il primo metodo progettato elabora il corpo della base di dati. Metodo che prevede la lettura dal Database creato prima e la relativa dichiarazione degli attributi nel programma.



```
{
  "classe": "scarpe da calcio",
  "domande": [
    "Il mio interesse è il calcio",
    "Scarpe con tacchetti",
    "Giocare a calcio"
  ],
  "risposte": [
    "Noi ti consigliamo le scarpe dell'Adidas, ma tutto il catalogo lo puo' trovare sul sito: [link]"
  ]
},
{
  "classe": "racchette e tavolo da ping pong",
  "domande": [
    "Mi piace giocare a ping pong",
    "Le racchette da ping pong sono la mia passione",
    "Tavolo da Ping Pong"
  ],
  "risposte": [
    "Noi ti consigliamo il tavolo e le racchette della Domyos, ma tutto il catalogo lo puo' trovare sul sito: [link]"
  ]
},
{
  "classe": "accessorio piscina",
  "domande": [
    "Mi piace nuotare",
    "Credo molto utile e divertente nuotare",
    "La piscina è il mio luogo preferito"
  ],
  "risposte": [
    "Noi ti consigliamo gli accessori della Piscina della Navigare, ma tutto il catalogo lo puo' trovare sul sito: [link]"
  ]
}
```

Figure 3.1: Mostro un esempio di domande del database che ho realizzato.

- Il secondo metodo progettato è il training del set di dati, che ritorna una tupla di due valori: l'array degli input e l'array degli output. I due array hanno lunghezza fissa. Questi sono gli array X e Y che utilizziamo per andare ad istruire la rete: creiamo il training set per la Rete Neurale trasformando le frasi da un linguaggio naturale a uno numerico.

- Il terzo metodo è quello descritto nel paragrafo precedente. Definisco e istruisco una ANN (Artificial Neural Network), di tipo DNN (Deep Neural Network), composta da un livello di input, due hidden layer e uno di output. I parametri sono sempre i soliti: X (array bidimensionale con i dati di input) e Y (array bidimensionale con i dati di output) appena generati. Andrò poi a salvare la rete, in maniera tale che si potrà utilizzare nuovamente da un'altro file senza essere nuovamente istruita.

- Ora scrivo due metodi che mi serviranno dopo: il primo prende in

input una frase, e la trasforma in una lista di temi, mentre il secondo trasforma questa lista in valori numerici (in modo da avere un input scritto in un formato compatibile ad essere elaborato dalla rete che abbiamo definito).

- Infine spiego il metodo che filtrerà ogni classe generando una probabilità per ognuna. Ho fissato una soglia di errore sotto cui non può andare. Dopo aver categorizzato ogni classe con una probabilità di appartenenza, andrò a scegliere quella più probabile.

Il terzo file che ho realizzato prevede l'utilizzo della rete sopra descritta. Questo file prevede nuovamente l'interazione con il database per avere la lista di risposte salvate in una categoria quando questa viene nominata dall'algoritmo di classificazione. Quindi, ordinando la struttura del file possiamo dire che tramite la lettura della rete già creata prima siamo ora pronti a fare varie domande. Questo è il file con cui l'utente interagisce, ed è composto dal metodo di generazione della rete, i due metodi citati prima (che generano la lista di temi) e infine il metodo che legge la risposta alla domanda che è stata appena classificata.

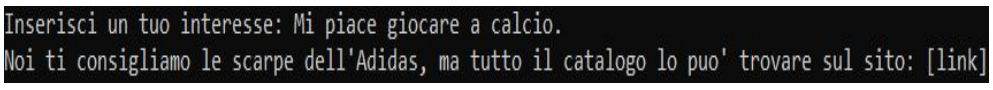


Figure 3.2: Mostro un esempio di utilizzo del programma che ho realizzato. L'utente utilizzatore ha inserito la frase "Mi piace giocare a calcio".

Ho realizzato anche un quarto ed ultimo file, in cui ho fatto i test che descrivo nel paragrafo successivo.

3.4 Test e Analisi dei risultati ottenuti

Ho creato la rete neurale indicando il numero di layer, la funzione di attivazione e utilizzando un database che ho creato io. Ho inoltre anche impostato la soglia di errore permessa. Innanzitutto ho fatto queste scelte misurando in tempo la creazione del modello e il numero di falsi positivi creati ogni turno. Essendo un database di dimensioni molto ridotte (circa cinquanta domande), e applicando il metodo della Convalida Semplice (introdotto nel paragrafo riguardanti gli alberi) ho seguito la tecnica solita per analizzare il mio modello. Di solito, trattando algoritmi di Apprendimento Supervisionato, non ha molto senso fare il training di questi modelli usando tutti i dati a disposizione per poi testarne l'efficienza provando a fare una previsione usando gli stessi dati. Per cui una via comunemente utilizzata è prendere una percentuale di dati a disposizione (come il 70%) e fare il training del modello con quelli, usando il restante 30% per fare i test e misurarne l'errore. In questo modo, avremo a disposizione dei dati nuovi per valutare effettivamente le prestazioni del modello, dato che il classificatore non li ha mai visti in fase di training.

Io ho fatto questo, e già mantenendo due livelli di profondità della rete, con rispettivamente 8 unità per livello, ho il 100% di predizione corretta dei dati. Questo è dovuto ai numeri ridotti del database: 50 domande divise in 10 categorie con relative risposte. Quindi non ci sono casi di falsi positivi (e questo è dovuto alla presenza carente di parole nel database). Più parole ci sono e più la rete neurale riesce a creare collegamenti e a funzionare nel modo migliore.

```
domanda= "Adoro cucinare nel tempo libero"
risposta = elabora_risposta(domanda)
if (risposta == "Noi ti consigliamo gli accessori di cucina della Kasanova, ma tutto il catalogo lo puo' trovare sul sito: [link]") :
    print("ok")
else :
    print("sbagliato")

domanda= "Quando cucino mi sento felice"
risposta = elabora_risposta(domanda)
if (risposta == "Noi ti consigliamo gli accessori di cucina della Kasanova, ma tutto il catalogo lo puo' trovare sul sito: [link]") :
    print("ok")
else :
    print("sbagliato")

domanda= "Mi interesso molto alla cucina"
risposta = elabora_risposta(domanda)
if (risposta == "Noi ti consigliamo gli accessori di cucina della Kasanova, ma tutto il catalogo lo puo' trovare sul sito: [link]") :
    print("ok")
else :
    print("sbagliato")

domanda= "Adoro passare le notti in cucina a cuinare"
risposta = elabora_risposta(domanda)
if (risposta == "Noi ti consigliamo gli accessori di cucina della Kasanova, ma tutto il catalogo lo puo' trovare sul sito: [link]") :
    print("ok")
else :
    print("sbagliato")

domanda= "Le torte che cucino rendono la mia vita più felice"
risposta = elabora_risposta(domanda)
if (risposta == "Noi ti consigliamo gli accessori di cucina della Kasanova, ma tutto il catalogo lo puo' trovare sul sito: [link]") :
    print("ok")
else :
    print("sbagliato")
```

Figure 3.3: Mostro parte delle domande che ho realizzato per testare il modello di rete creato.

Quindi la mia unica analisi è dovuta ad un fattore di tempo: ho infatti notato come la creazione di layers faceva variare proporzionalmente il tempo di esecuzione dell'algoritmo.

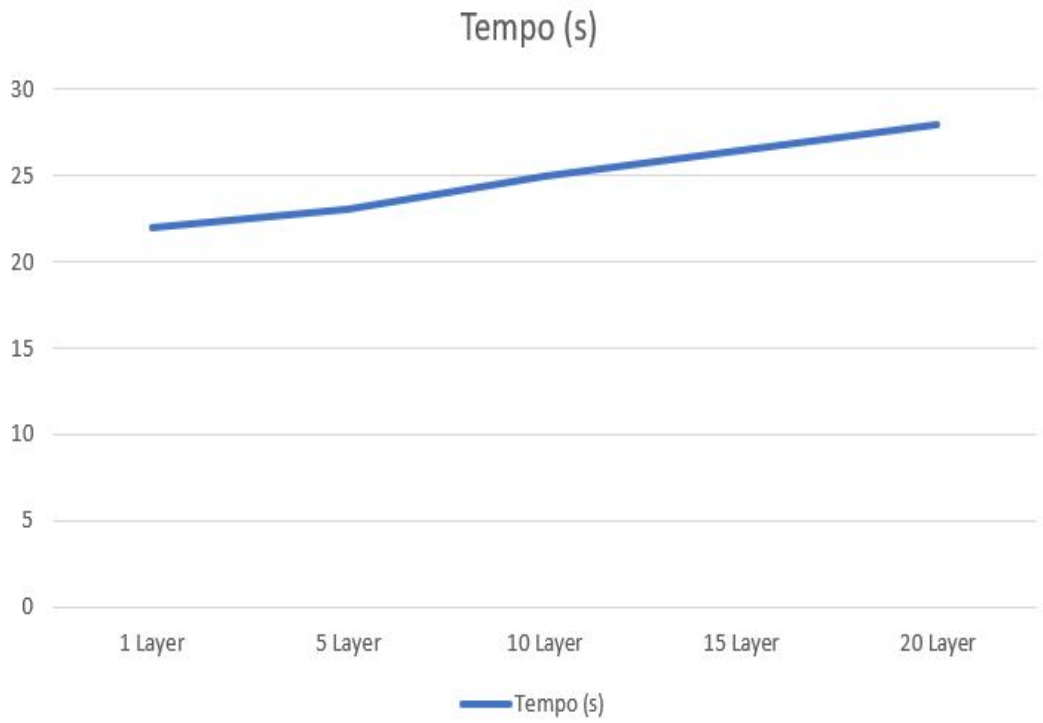


Figure 3.4: Funzione che mostra l'aumentare del tempo di esecuzione se cresce il numero di Layer creati nella rete neurale.

Questo dimostra come (per questo esercizio) basta mantenere un numero ridotto di layers. Bisogna anche ricordare come questo ha anche influito sulla scelta della funzione di attivazione: ho dimostrato che con numeri così ridotti, al variare della funzione di attivazione, non si riesce ad avere un riscontro nè sul tempo di esecuzione nè sui falsi positivi.

Chapter 4

Conclusioni

Considerando i risultati ottenuti, siamo in grado di offrire un prodotto adeguato agli interessi del consumatore, e questo grazie alla rete neaurale semplice che è stata creata. Abbiamo dimostrato che per la risoluzione del nostro problema, con una rete neurale con due livelli di profondità e 8 unità per livello ho il 100% di predizione corretta dei dati. Riesco quindi a classificare l'utente correttamente proponendogli il prodotto adeguato alle sue esigenze.

La parte più interessante di questa applicazione sono le future estensioni che potrebbero realizzarsi. Gli utilizzi di questa applicazione creata sono molti, e le idee avute sono davvero molte. Un utilizzo potrebbe essere quello di sistemi e applicazioni che la utilizzerebbero per sapere che pubblicità proporre ad ogni consumatore che si interfaccia con il sistema appena citato. Un altro utilizzo potrebbe essere quello di creare un'applicazione (magari per smartphone) che permette di realizzare un manager: in base a ciò che l'utente inserisce, l'app riesce a offrire i prodotti che gli potrebbero interessare. Ma non solo: potrebbe fornire un servizio di "raccolta" di molte altre

applicazioni, permettendo di avere in una sola schermata le informazioni più rilevanti dello smartphone. Questo permetterebbe la simulazione di un "assistente manager" capace di proporre (simulando un umano) prodotti e servizi adatti al cliente e a raccoglierli in una app che permette la gestione di tutto il dispositivo.

Abbiamo quindi visto come influire sulle scelte del consumatore: attraverso l'utilizzo di questa applicazione vado ad influenzare la vita del consumatore stesso. Questo lo faccio proponendogli prodotti mirati al suo interesse, sollecitandone l'acquisto.

Bibliography

[1] Alessandro Cucci, *A tu per tu col Machine Learning*, thedotcompany (2017).

[2] Steven Struhl, *AI Marketing*, Apogeo (2017).

[3] Elena Loli Piccolomini e Antonio Messina, *Statistica e calcolo con R*, McGraw-Hill Education (2015).

[4] Prof. Andrea Asperti, *slide e appunti del corso di Machine Learning*, UNIBO.

[5] Prof. Marco Di Felice, *slide e appunti del corso di Base di Dati (appendice su Machine Learning)*, UNIBO.

[6] Prof. Eduardo Rossi, *slide e appunti del corso di Econometria*, UNIPV.

[7] Prof. Davide Maltoni, *slide e appunti corso di Machine Learning*, UNIBO, `http://bias.csr.unibo.it/maltoni/ml/DispensePDF/9_ML_DeepLearning.pdf`,`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.188.7660&rep=rep1&type=pdf`.

[8] Prof. Antonino Furnari, *appunti su Tensorflow*, Università di Catania, http://www.dmi.unict.it/~furnari/teaching/SMM1617/lab4/9.

[9] Documentazione API tflearn, http://tflearn.org.

[10] Tutorial Tensorflow, https://www.tensorflow.org/tutorials/.

[11] Intelligenza Artificiale, http://www.intelligenzaartificiale.it/reti-neurali/.

[12] Deep Learning, http://www.deeplearningitalia.com/una-panoramica-introduttiva-su-deep-learning-e-machine-learning/.

[13] Deep Neural Network, https://medium.com/@williamkoehrsen/deep-neural-network-classifier-32c12ff46b6c.

[14] Training delle reti neurali, https://medium.com/@alessandropadrin/come-velocizzare-il-training-delle-reti-neurali-parte-ii-1ae149520ddb.

[15] Alberi Decisionali, https://www.tesionline.it/appunto/223/52/Definizione_di_alberi_decisionali.