UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI ECONOMIA "GIORGIO FUÀ"

Corso di Laurea Magistrale o Specialistica in
Data Science per l'Economia e le Imprese

Reti Neurali Artificiali nella Diagnosi di Diabete

Artificial Neural Networks in the Diagnosis of Diabetes

Relatore: Chiar.mo
Prof. Polinesi Gloria

Tesi di Laurea di:
Balducci Nicholas

Anno Accademico 2022 – 2023

Ringraziamenti

Il mio primo ringraziamento va ai miei genitori, in particolare a mia madre senza la quale non sarei arrivato all'importante traguardo ottenuto oggi perché mi ha sempre spronato a studiare e a volere di più dalla vita.

Grazie a Roberta per le lunghe chiacchierate notturne, per la sua capacità di rimettermi in piedi, di tirarmi su il morale e darmi allegria, a Lucia, per la costante vicinanza, per le serate di studio indimenticabili e per la sua capacità di vedere il bicchiere sempre mezzo pieno.

A mia nonna Luisa, per il supporto e la pazienza.

Infine, ringrazio la mia professoressa e relatrice Polinesi Gloria per il notevole supporto teorico e disponibilità avuta nei miei confronti.

Indice

Capitolo 1

INTRODUZIONE

Le reti neurali artificiali (ANN) sono potenti modelli di classificazione e di predizione[1] in grado di utilizzare decision boundaries altamente complessi (linee o piani che suddividono le osservazioni tra le due parti opposte a seconda delle loro caratteristiche non necessariamente lineari). Hanno ottenuto un'ampia accettazione in diverse applicazioni come nell' OCR (riconoscimento ottico dei caratteri), nell'elaborazione delle immagini e del linguaggio, nelle diagnosi mediche, nella bioinformatica e più in generale nel data mining. Inizialmente, le ANN sono state create per replicare il funzionamento del cervello umano e le strutture biologiche che compongono quest'ultimo allo scopo di replicarne l'intelligenza. L'unità fondamentale del sistema nervoso è la cellula nervosa o neurone, che svolge il ruolo comunicativo del sistema e ha tre proprietà fisiologiche fondamentali necessarie per questa funzione:

[1] I primi rispondono a problemi di discriminazione cioè vengono utilizzati per etichettare soggetti in base alle loro caratteristiche, ad esempio, un problema comune nel quale tali modelli sono applicati è l'identificazione di clienti solvibili e insolvibili, mentre, il modello predittivo, fornisce come risultato una predizione (forecast) basandosi su eventi già conclusi (dati storici).

- **Eccitabilità (irritabilità):** tutte le cellule sono eccitabili, cioè rispondono a modifiche ambientali (stimoli);
- **Conducibilità:** i neuroni rispondono agli stimoli generando segnali elettrici che sono velocemente trasmessi ad altre cellule localizzate a distanza;
- **Secrezione:** quando il segnale elettrico raggiunge la terminazione nervosa, il neurone secerne un neurotrasmettitore chimico che attraversa la fessura sinaptica e stimola la cellula che vi si trova a ridosso.

Il centro di controllo del neurone è il neurosoma detto anche soma o corpo cellulare che dà origine ad alcuni sottili processi che si ramificano in un gran numero di dendriti chiamati così per la loro somiglianza con i rami spogli di un albero d'inverno. I dentriti rappresentano il sito principale per la ricezione dei segnali da altri neuroni. Ad un polo del soma troviamo una protuberanza chiamata monticolo assonico luogo nella quale il neurone genera il potenziale d'azione e dal quale origina l'assone (fibra nervosa), quest'ultimo è specializzato nella conduzione del segnale nervoso in punti anche molto lontani dal soma. L'assone termina con un esteso complesso di fini ramificazioni che terminano con un bottone sinaptico un piccolo rigonfiamento che forma una congiunzione (sinapsi) con un'altra cellula. Le sinapsi rendono possibile l'integrazione neurale (elaborazione dell'informazione), ognuna di esse rappresenta un dispositivo per prendere decisioni, che determina se una seconda cellula risponderà ad un segnale che proviene dalla prima; senza tali strutture biologiche il sistema nervoso sarebbe incapace di prendere qualsiasi decisione

(Kenneth, 2012). Figura 1.1 mostra le similitudini tra un neurone biologico ed uno artificiale.

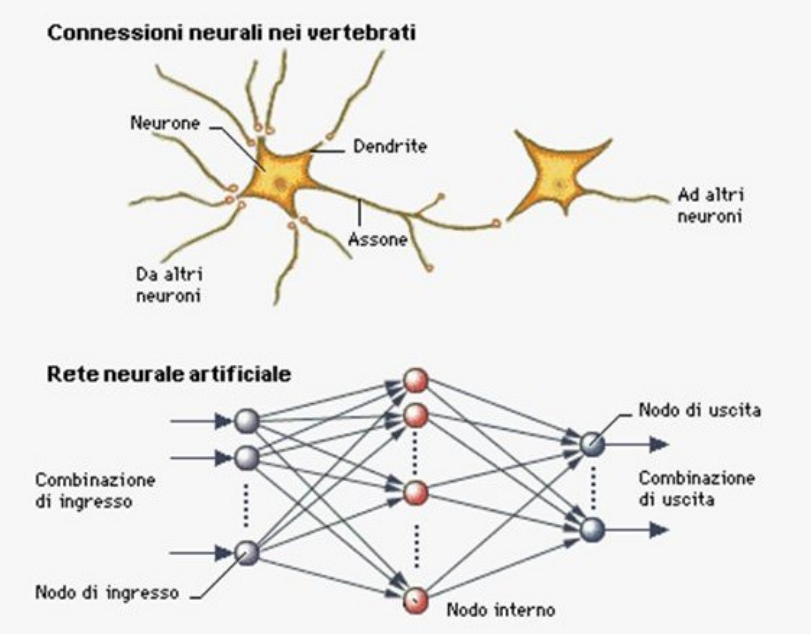

Figura 1.1: Cellula nervosa (in alto) e rete neurale artificiale "Two-layer network" (in basso).

Le reti neurali sono impiegate in vari ambiti[2].

- **Retail.** Al fine di instaurare relazioni proficue e durature con i clienti e migliorare le strategie di leadership. Uno degli esempi più classici negli acquisti online sono i consigli di Amazon: quando si effettua un acquisto, viene visualizzato un elenco di altri articoli simili, acquistati da altri acquirenti;

- **Valutazione dei rischi.** Le compagnie di assicurazione, negli ultimi anni, hanno visto ridurre le proprie perdite grazie all'analisi predittiva,

[2]https://www.it-impresa.it/blog/cosa-sono-le-reti-neurali.

che le ha aiutate ad analizzare e stimare le perdite future, a pianificare campagne di marketing e a fornire informazioni più puntuali;

- **Servizi finanziari.** L'analisi predittiva aiuta a ottimizzare la strategia aziendale, a partire dalla generazione dei ricavi e delle vendite fino alla gestione delle risorse;

- **Energia.** Al fine di ridurre il pericolo di guasti alle apparecchiature delle centrali elettriche, contribuendo così a ridurre i costi di manutenzione e a migliorare la disponibilità di energia;

- **Social media.** Rappresentano ormai uno strumento nuovo per produrre dati e informazioni utili alle aziende, con la possibilità, attraverso il monitoraggio costante dei commenti, di ottenere feedback immediati da parte di clienti, attivi e potenziali;

- **Medicina.** Per l'elaborazione dei risultati dei test, le immagini e altri dati dei pazienti inviando poi le informazioni al medico curante quando viene rilevata la malattia;

- **Arte.** Le reti neurali sono in grado di creare dipinti, libri e musica, imitando autori famosi (con il giusto quantitativo di dati grezzi);

- **Motori di ricerca.** Gli algoritmi analizzano le informazioni delle pagine web, riconoscendo il tipo di contenuto per poi fare apparire suggerimenti in base alle ultime ricerche fatte dagli utenti;

- **Navigare.** Quando sono collegate a Internet, le reti neurali monitorano la congestione del traffico e suggeriscono il percorso migliore (navigatore satellitare);

- **Agricoltura.** Un sistema di reti che distingue le erbe infestanti dalle piante coltivate può essere installato sulle macchine da raccolta per ottenere una maggiore qualità;

- **Meteorologia.** I radar meteorologici trasmettono i dati alle reti neurali che forniscono previsioni del tempo per le ore successive dopo aver analizzato i modelli di precipitazioni;

- **Sistemi di sicurezza.** Le ANN vengono utilizzate nello sviluppo di telecamere per il riconoscimento facciale. Il monitoraggio dei sistemi aziendali consente di reagire tempestivamente ad accessi non autorizzati, introduzione di virus, attacchi DDoS, ecc.

Il lavoro è suddiviso in quattro capitoli, il primo concerne la descrizione della struttura o architettura delle principali reti neurali mentre nel secondo vengono illustrate metodologie e funzionamento. Il terzo capitolo propone un'analisi empirica su pazienti diabetici mettendo a confronto con un' ANN un modello di regressione logistica; nel quarto capitolo, si confrontano i risultati ottenuti.

1.1 Storia delle reti neurali

La macchina di Turing (o più brevemente MdT), è una macchina ideale che manipola i dati contenuti su un nastro (il quale funge da memoria), secon-

do determinate regole, cioè, si tratta di un modello astratto che definisce una macchina in grado di eseguire vari algoritmi e dotata di un nastro potenzialmente infinito su cui può leggere e/o scrivere dei simboli (Douglas *et al.*, 2012). Nel 1943, W.S. McCulloch e W. Pitts (figura 1.2) in un famoso lavoro intitolato "A Logical Calculus of the Ideas Immanent in Nervous Activity" si spiega come una macchina di Turing può essere realizzata utilizzando una rete finita di neuroni al fine di dimostrare che i neuroni sono l'elemento fondamentale di base del cervello artificiale.

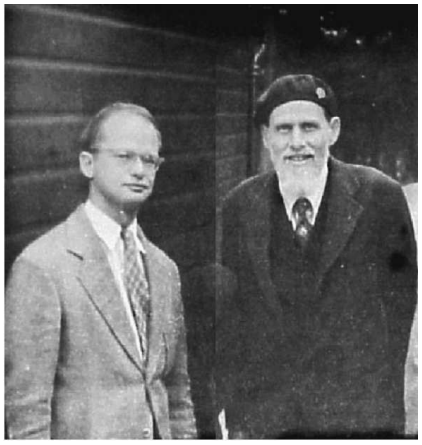

Figura 1.2: McCulloch (a destra) e Pitts (sinistra) nel 1949.

Nel 1949 Donald Olding Hebb (figura 1.3), uno dei primi scienziati ad approfondire il legame tra sistema nervoso e comportamento, ipotizza la possibilità di istruire le macchine con un apprendimento (learning Hebb) in grado di emulare quello alla base dell'intelligenza dell'uomo (Hebb, 1949). Nel modello formale di apprendimento elaborato dallo psicologo americano, l'apprendimento è spiegabile secondo tre ipotesi.

- La prima ipotesi è che i neuroni corticali rafforzino le loro connessioni quando risultano frequentemente contemporaneamente attivi. Que-

sto principio di apprendimento associativo sembra essere valido per la maggioranza dei neuroni corticali (regola di Hebb).

- La seconda ipotesi è che la corteccia sia un'enorme memoria associativa in cui il rafforzamento delle sinapsi abbia luogo non solo tra neuroni vicini ma anche tra neuroni in aree corticali distanti. Questa seconda ipotesi trae sostegno da studi neuroanatomici[3].

- Secondo la terza ipotesi, la contemporanea e frequente attivazione di un gruppo di neuroni che dà luogo al rafforzamento sinaptico ha conseguenze funzionali importanti. I neuroni fortemente connessi probabilmente agiscono insieme come un'unità funzionale; se vengono attivati solo alcuni dei neuroni, si attiverà l'intero gruppo, a causa delle forti connessioni tra i membri del gruppo stesso. Se l'insieme è attivo, i suoi neuroni sono attivi simultaneamente, o mostrano, schemi di funzionamento sincronizzati in modo preciso quando l'attività neuronale si diffonde nell'insieme.

Gli insiemi hebbiani di cellule si possono definire come unità funzionali composte da molti neuroni che si formano in una rete associativa, la corteccia, come risultato di una frequente attività neuronale simultanea che causa un rafforzamento sinaptico.

[3]https://coralieflorino.com/it/regola-di-hebb-con-unanalogia-psicologia-e-neuroscienze.

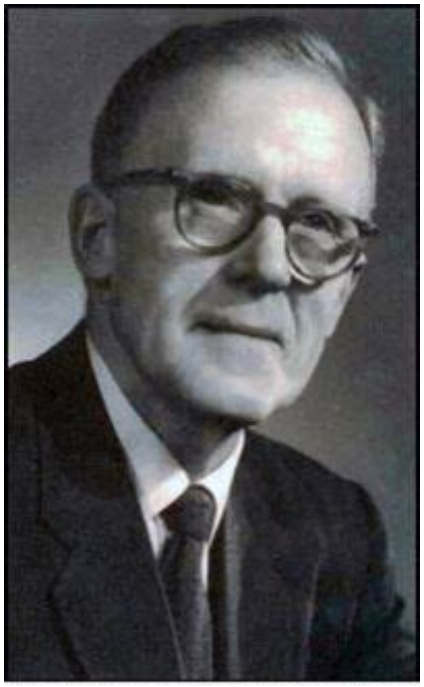

Figura 1.3: Donald Olding Hebb.

Nel 1958, J. Von Neumann nella sua opera "The Computer and the Brain", esamina le soluzioni proposte dai precedenti autori sottolineando la scarsa precisione delle strutture nello svolgimento di operazioni complesse. Nello stesso anno, Frank Rosenblatt (figura 1.4) nella rivista "Psychological review", introduce il primo schema di rete neurale, detto percettrone (perceptron), precursore delle attuali reti neurali, per il riconoscimento e la classificazione di forme utilizzando un sistema lineare per la risoluzione di problemi e la classificazione di dati di input divisibili linearmente all'interno dello spazio osservabile. Il modello probabilistico di Rosenblatt è mirato all'analisi, in forma matematica, di funzioni quali l'immagazzinamento delle informazioni e della loro influenza sul riconoscimento dei pattern. Esso costituisce un progresso decisivo rispetto al modello binario di McCulloch e Pitts, in quanto

i pesi sinaptici sono variabili. Il modello è semplice e possiede solo gli strati fondamentali di input e output senza alcun elemento intermedio (hidden layer) il che comporta l'impossibilità di risolvere problemi non caratterizzati da separabilità lineare delle soluzioni.

Figura 1.4: Frank Rosenblatt.

Nel 1969 nell'opera "Perceptrons. An Introduction to Computational Geometry", Marvin Minsky e Seymour A. Papert mostrano i limiti operativi del percettrone di Rosenblatt. Nel 1974, Paul Werbos nella sua tesi di dottorato fornisce un modello matematico su come addestrare una multi-layers perceptron (MLP). Nel 1986 David E. Rosenblatt, G. Hinton e R. J. Williams basandosi sul modello creato da Werbos propondono un algoritmo di addestramento delle reti neurali feed-forward[4] in grado di modificare, in modo

[4]Si tratta di una rete neurale artificiale dove le connessioni tra le unità non formano

iterativo, i pesi tra i nodi avvicinandosi ad un risultato desiderato (backpropagation algorithm). Il backpropagation algorithm si basa sul metodo della discesa del gradiente.

1.2 Architettura delle reti e overfitting

L'unità fondamentale che compone la rete neurale è il neurone (o nodo). Essi sono raggruppati in strati (layers) in modo tale che i vari neuroni siano collegati solo ai nodi del layer precedente e del successivo. Ciascun neurone riceve simultanei segnali di input (x) attraverso pesi di connessione (w) e produce un solo segnale di output (y). I pesi sinaptici (i quali indicano la forza di una connessione fra due nodi e quindi l'importanza che l'input ha nella trasmissione dell'impulso finale del neurone) sono dinamici, permettendo alla macchina di apprendere, in un modo simile, anche se molto più elementare, a quello delle reti neurali biologiche. I neuroni elaborano le informazioni in input, i pesi e un valore di soglia (b_0) attraverso una funzione di combinazione e produce un valore chiamato potential p, la funzione di attivazione σ (figura 1.5) trasforma il potential in segnale di output, il quale, sarà l'input di un altro neurone oppure l'output definitivo nel caso di strato finale dell'output. Considerando un neurone j avente n segnali di input x tali che $x = [x1, \ldots, xn]$ e pesi w associati tali che $w = [w1j, \ldots, wnj]$, la funzione

cicli (differenziandosi dalle reti neurali ricorrenti), le informazioni si muovono solo in una direzione, avanti, rispetto a nodi d'ingresso, attraverso nodi nascosti (se esistenti) fino ai nodi d'uscita quindi nella rete non ci sono cicli. Le reti feedforward non hanno memoria di input avvenuti a tempi precedenti, per cui l'output è determinato solamente dall'attuale input, nelle reti feedback è possibile che le informazioni tornino negli strati precedenti.

di combinazione (quindi il potential) sarà data da:

$$p_j = \sum_{i=1}^{n} (x_i w_{ij} + b_0) \tag{1.1}$$

La funzione di attivazione trasforma il potential in segnale di output:

$$\hat{y} = \sigma(x, w) = \sigma(p_j) = \sigma \sum_{i=1}^{n} (x_i w_{ij} + b_0) \tag{1.2}$$

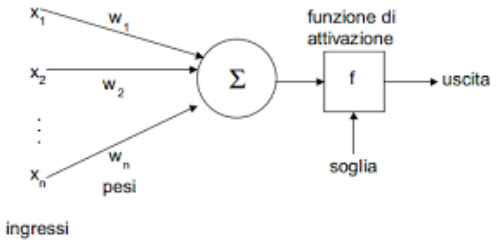

Figura 1.5: Generica funzione di attivazione.

Gli strati a seconda del loro ruolo possono essere input layer, hidden layer (strato nascosto) o output layer (figura 1.6).

- **Input layer:** i neuroni in tale strato hanno il compito di recepire informazioni solo ed esclusivamente dall' ambiente esterno al fine di trasmetterle ai neuroni negli strati successivi. Ogni neurone in questo strato corrisponde ad una variabile esogena quindi si hanno neuroni pari al numero delle caratteristiche in input, in tale strato non c'è funzione di combinazione in entrata ma solo in uscita verso neuroni degli altri strati;

- **Hidden layer:** si chiamano in tale modo perché non entrano in contatto con l'ambiente esterno essendo posti tra lo strato di input e quello di output, possono essere assenti (nei modelli più semplici come, ad esempio, nel perceptron) o molteplici come in quelli più complessi (reti profonde o deep), questi strati sono utilizzati solo ed esclusivamente per l'analisi con il compito di fare in modo che la relazione tra x e y sia più vicina possibile ai dati osservati;

- **Output layer:** i neuroni situati nello strato di output restituiscono il risultato finale, quindi, sono collegati solo ed esclusivamente con i neuroni dello strato precedente, in genere ognuno di questi neuroni corrisponde alle variabili di risposta y, di conseguenza, nello strato finale possiamo trovarne solo uno (problemi di classificazione in due classi) o molti (classificazione in più classi).

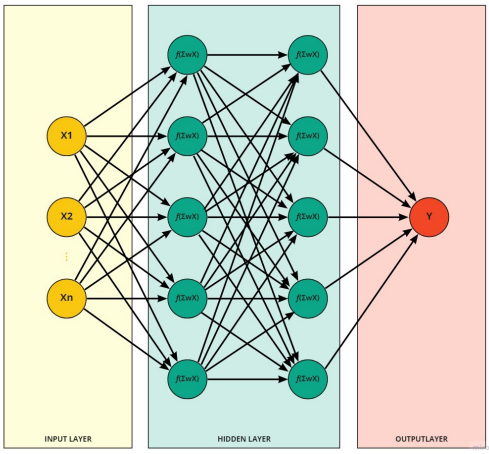

Figura 1.6: I differenti tipi di layer in una ANN.

Giudici e Figini (2014) differenziano le ANN in base a quattro caratteristiche:

- **Grado di differenziazione tra strati di input e di output:** il numero di nodi di input corrisponde al numero delle caratteristiche in nostro possesso in quanto ogni nodo di input corrisponde ad un vettore di dati, mentre il numero dei nodi di output è deciso in base all'obiettivo di analisi cioè, variano a seconda se si ha uno scopo predittivo (più neuroni di output) o descrittivo (nella classificazione binaria è sufficiente un solo neurone che restituisca un'etichetta 0 o 1);

- **Numero dei layer:** a seconda del numero di hidden layer la rete è in grado di far fronte a problemi più complessi, per esempio le reti single layer non possono essere applicate a problemi non lineari a differenza delle multilayer o delle reti neurali profonde superiori;

- **Direzione del flusso di calcolo:** le reti feedforward trasmettono l'impulso dagli input layer fino agli output layer ma ad esempio, nelle reti ricorrenti, l'impulso è in grado di tornare indietro creando un loop tra gli strati uno dei quali viene utilizzato per la memorizzazione del risultato precedente;

- **Tipo di connessione:** le reti neurali possono utilizzare tipologie di funzioni di attivazione differenti.

Nel data mining, un algoritmo di apprendimento viene allenato usando un training set cioè un determinato numero di esempi nei quali l'output è già conosciuto a priori. Lo scopo dell'apprendimento è quello di predire l'output di nuovi dati (test set) basandosi sull'addestramento fatto dal training set, cioè si assume che il modello di apprendimento sarà in grado di generalizzare.

Nei casi in cui l'apprendimento è stato effettuato troppo a lungo o nel caso di uno scarso numero di esempi nel training set, il modello potrebbe adattarsi a caratteristiche che sono specifiche solo del training set (overfitting). In presenza di overfitting, quindi, il modello è caratterizzato da un alto errore di generalizzazione nel test set (figura 1.7).

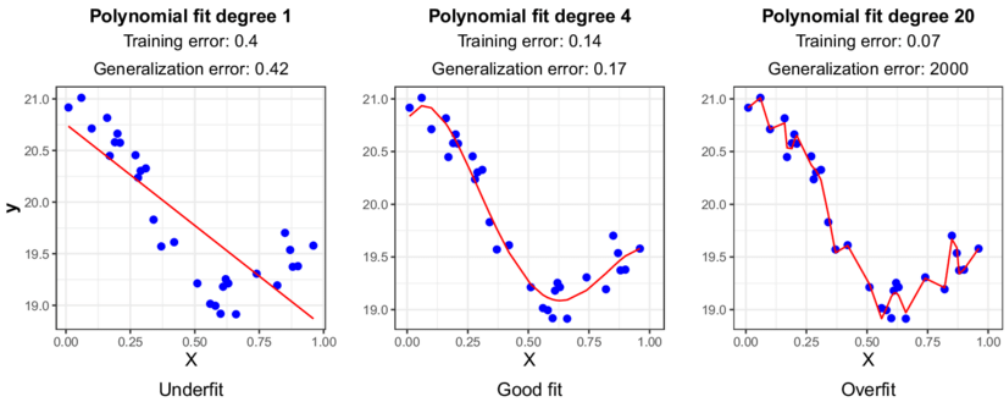

Figura 1.7: Differenza tra underfitting, buon fit e overfitting.

Le reti posso essere supervisionate o non supervisionate. Le supervisionate considerano rapporti di causalità tra le variabili, consentono di rispondere a problemi di predizione, la "supervisione" consiste nel conoscere che risposta ci si aspetta da un determinato input considerando che in tali algoritmi l'output in uscita $\hat{y}$ va confrontato con quello desiderato y al fine poi di modificare il modello. Le non supervisionati, hanno lo scopo di classificare dati in etichette (o classi) grazie ad un modello di classificazione utilizzando pattern contenuti nel dataset. Le reti supervisionate sono:

- Single layer perceptron;
- Multilayer perceptron;
- Convolutional neural network;

- Recurrent neural network;
- Reti "deep".

Mentre l'unica rete ad essere non supervisionata è la Kohonen network.

1.2.1 Singol layer perceptron (SLP)

Il single layer perceptron (figura 1.8) ha la struttura più semplice tra le ANN. Capace di restituire solo ed esclusivamente un output con valore 1 oppure 0 utilizzando una funzione di attivazione detta di soglia (threshold) è costituito da un solo strato di input collegato direttamente all'output, è impiegato esclusivamente nella classificazione binaria (quindi nella discriminazione in classi) mediante pattern lineari. Il percettrone a singolo strato può essere considerato come il più semplice modello di rete neurale feedforward in quanto gli input alimentano direttamente l'unità di output.

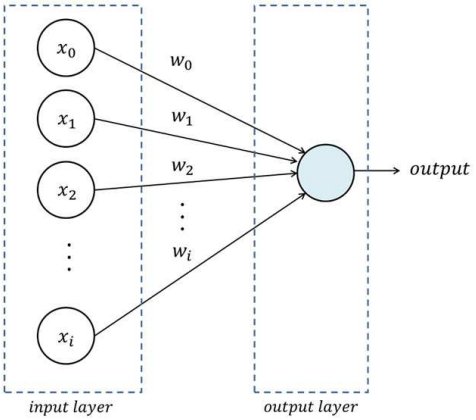

Figura 1.8: Singol layer perceptron.

1.2.2 Multilayer perceptron (MLP)

Reti feedforward e fully connected[5] costituiscono l'evoluzione del perceptron a singolo strato; potendo utilizzare più tipologie di funzione di attivazione anche simultaneamente e avendo in aggiunta uno o più strati nascosti con lo scopo di fare analisi, consentono quindi una discriminazione (o anche previsione) non basata solo ed esclusivamente su pattern lineari ma anche non lineari. Gli strati intermedi determinano la profondità della rete e permettono di raffinare le informazioni derivanti dai neuronoi di input. Ogni hidden layer costituisce un livello di astrazione differente[6] e a seconda della quantità di quest'ultimi la complessità della rete aumenta, aumentando di conseguenza, il rischio di overfitting. Non potendo stimare con precisione cosa avviene fra gli strati di input e di output, le Multilayer perceptron vengono dette "Black Box" (figura 1.9).

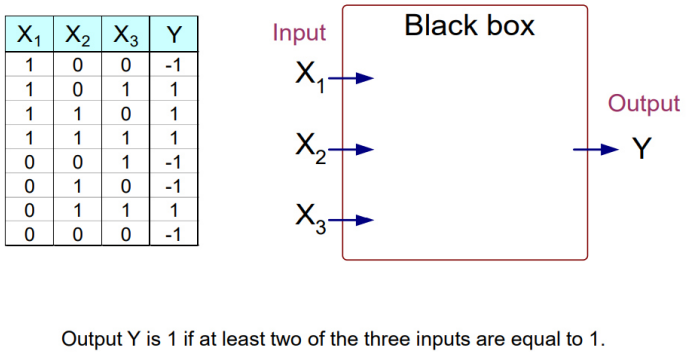

Figura 1.9: MLP rappresentata come una black box.

[5] ANN nella quale ogni neurone dello strato precedente è collegato ad ogni neurone del successivo.

[6] In informatica indica quanto il codice scritto in un linguaggio di programmazione si distacchi dal linguaggio macchina. Delle istruzioni scritte in Java, per esempio, sono molto più vicine al linguaggio comprensibile all'uomo piuttosto che a quello comprensibile dalla macchina (alto livello di astrazione), mentre, delle istruzioni scritte in Assembly sono abbastanza vicine (ma non uguali) alle istruzioni in formato comprensibile alla macchina (basso livello di astrazione).

1.2.3 Convolutional neural network (CNN)

Unica rete neurale non fully connected presentata fino ad ora, le convolutional neural networks (figura 1.10) o reti neurali convoluzionali (CNN) sono delle particolari reti che uniscono i neuroni dello strato successivo solo con una parte (sottoinsieme) dei neuroni dello strato precedente, pertanto, sono una versione regolarizzata e meno complessa di multilayer perceptron, quindi meno soggette ad overfitting tipico delle connessioni complete. Hanno applicazioni nel riconoscimento di immagini e video come, ad esempio, nei sistemi di raccomandazione, classificazione, segmentazione e analisi delle immagini mediche e non, elaborazione del linguaggio naturale, interfacce cervello-computer e serie temporali finanziarie.

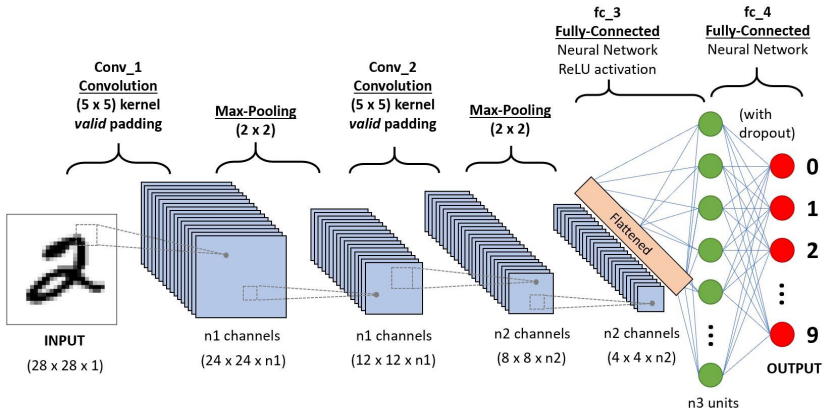

Figura 1.10: Convolutional neural network (CNN).

Anche tali reti si ispirano a processi biologici, in questo caso all' organizzazione della corteccia visiva animale. I singoli neuroni corticali rispondono agli stimoli solo in una regione ristretta del campo visivo nota come campo recettivo. I campi ricettivi di diversi neuroni si sovrappongono parzialmente in modo tale da coprire l'intero campo visivo. Le CNN utilizzano una

preelaborazione relativamente ridotta rispetto ad altri algoritmi di classificazione delle immagini inquanto utilizzano un apprendimento automatico, negli algoritmi tradizionali questi filtri sono progettati manualmente.

1.2.4 Recurrent neural network (RNN)

Nelle reti neurali ricorrenti (figura 1.11), lo strato di output è collegato con quello di input o di hidden a loop; questa interconnessione tra strati permette l'utilizzo di uno dei layer come memoria di stato fornendo in ingresso una sequenza temporale. Di conseguenza l'output finale risulta essere influenzato dai risultati al tempo $t-1$, ciò rende tali reti ideali nell'analisi predittiva su sequenze di dati quali possono essere ad esempio il riconoscimento della grafica e il riconoscimento vocale.

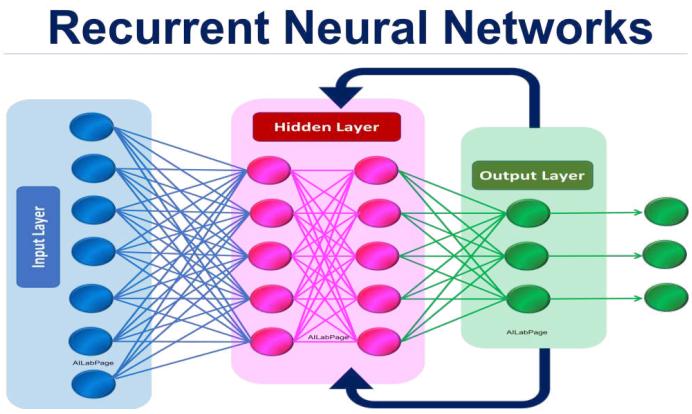

Figura 1.11: Recurrent neural network (RNN).

1.2.5 Deep learning

Nel momomento in cui gli strati nascosti sono cinque o più, alla rete viene attribuita la nomenclatura di "deep" (profonda) (Vipin *et al.*, 2018).

Tali reti mostrano un enorme potenziale di approssimazione ma l'addestramento risulta essere difficoltoso a causa del gran numero di livelli nascosti, oltre che alle ragioni legate alle risorse computazionali limitate e alle architetture hardware. In primo luogo, è difficile completare un addestramento avente come risultato un basso errore di classificazione o di forecast a causa di un problema detto "saturazione dell'output", in secondo luogo, l'apprendimento delle reti neurali profonde è abbastanza sensibile ai valori iniziali dei parametri del modello, principalmente a causa della funzione di costo utilizzata e della conseguente lenta convergenza del gradiente, in fine, tali reti, essendo molto complesse, sono suscettibili all'overfitting. Per risolvere tali problemi, oltre all'utilizzo di nuove tecnologie per il calcolo computazionale come GPU rispetto alle più classiche CPU, sono stati fatti passi avanti anche dal punto di vista degli algoritmi. Tali miglioramenti includono l'uso di combinazioni più reattive di funzioni di costo (o di perdita) e funzioni di attivazione, una migliore inizializzazione dei parametri del modello, nuove tecniche di regolarizzazione, progetti di architettura più agili e tecniche migliori per l'apprendimento del modello e la selezione di iperparametri. Tramite la tecnica di regolarizzazione chiamata "drop out" (figura 1.12), è possibile ridurre le connessioni tra i neuroni evitando una rete fully connected con la conseguente riduzione dei costi di computazione e del rischio di overfitting. L'obiettivo principale del drop out è evitare l'apprendimento di caratteristiche spurie in nodi nascosti, che si verificano a causa dell'overfitting del modello. Utilizza l'intuizione di base che le caratteristiche spurie spesso si "coadattino" mostrando quindi buone prestazioni di allenamento solo se utilizzate in combinazioni altamente selettive, quindi le interrompe.

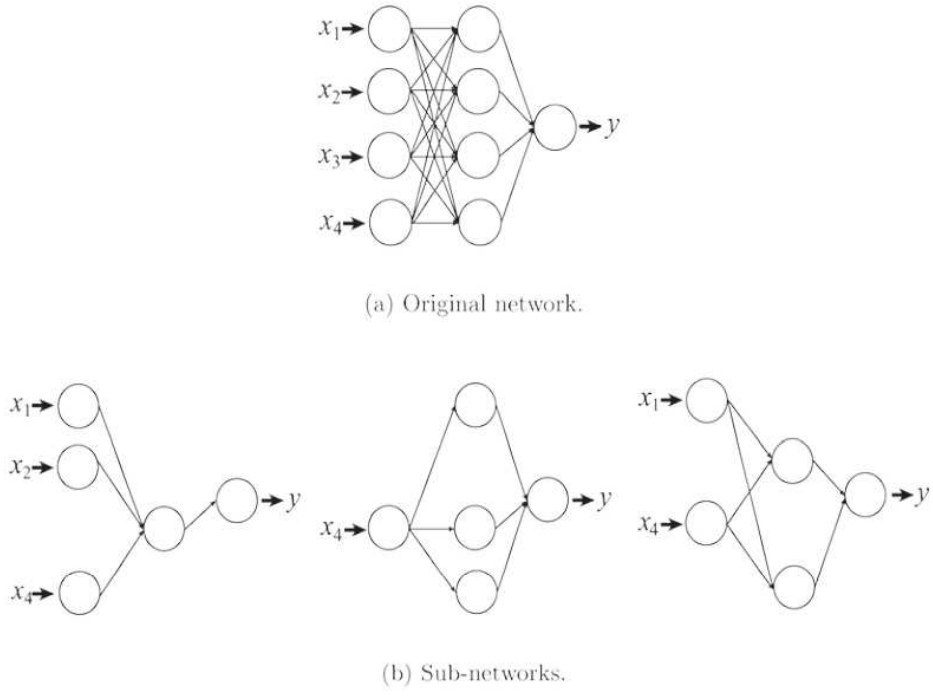

Figura 1.12: Drop out.

1.2.6 Kohonen network (SOM)

A differenza dei modelli di rete descritti finora, le reti di kohonen (figura 1.13) o self organizing maps (SOM) sono reti non supervisionate impiegate per problemi di discriminazione dei dati dividendo gli input in categorie tra loro omogenee. Una SOM, tramite un metodo di apprendimento detto "competitivo", crea una mappa topografica raggruppando quelli definiti simili, ad esempio, è possibile fornire ad una SOM una serie di colori e fare in modo che essa raggruppi i colori assomiglianti. Nascono dallo studio della topologia della corteccia del cervello umano e tengono conto non solo delle connessioni sinottiche tra neuroni ma anche della influenza che può avere un neurone sul vicino. Nei sistemi biologici, i neuroni che sono fisicamente vicini

a neuroni attivi hanno i legami più forti mentre quelli più lontani vengono inibiti, questa caratteristica è alla base delle SOM, la variazione dei pesi avviene solo in nodi vicini ad un neurone scelto. A livello strutturale, una SOM è costituita solo da uno strato di input (il quale come in una normale rete serve a pesare i dati in entrata) e da uno strato di output, i neuroni che calcolano il risultato della rete sono organizzati su una griglia posta su un piano. Ciascun neurone di input è connesso a tutti i neuroni di tale griglia, l'algoritmo di apprendimento è eseguito tramite le interconnessioni laterali tra neuroni vicini.

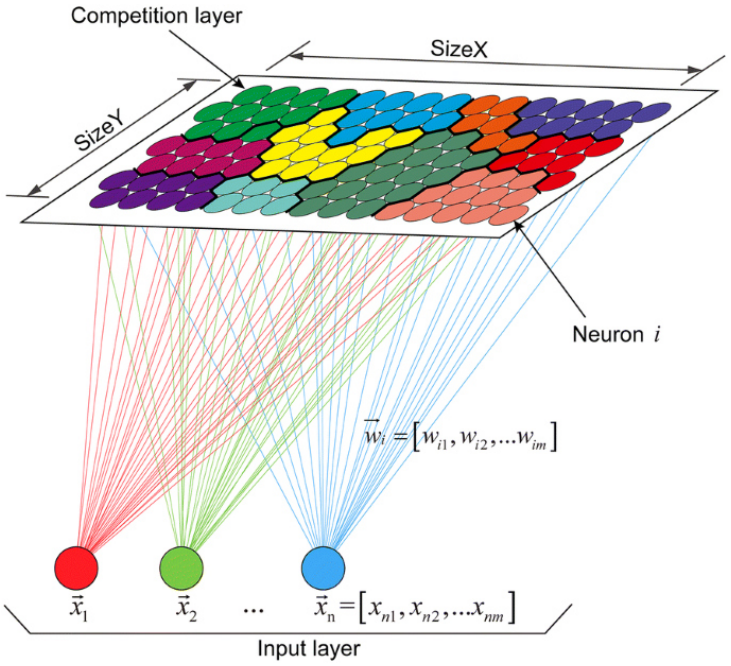

Figura 1.13: Self organizing map (SOM).

Tale rete è paragonabile ad un metodo di clusterizzazione non supervisionato basato sui centroidi. Il suo obiettivo è quello di trovare un insieme di vettori di riferimento che fungono da centroidi attorno ai quali gli oggetti del dataset vengono classificati a seconda delle proprie caratteristiche, c'è un

neurone associato a ciascun centroide. Infatti, come nel caso del k-means, gli oggetti dati vengono elaborati uno alla volta e il baricentro più vicino viene aggiornato, ma a differenza di quest'ultimo, la SOM, imponendo un ordinamento topografico sui centroidi aggiona anche quelli vicini. L'elaborazione dei punti continua fino al raggiungimento di un limite predeterminato o fino a quando non ci sia più una variazione significativa dei centroidi. L'output finale della SOM è un insieme di centroidi che definiscono implicitamente i cluster costituiti dai punti più vicini a un particolare baricentro o vettore di riferimento, in altre parole, i centroidi che sono vicini l'uno all'altro nella griglia sono più strettamente correlati tra loro rispetto ai centroidi che sono più lontani. Ogni neurone identifica una classe a cui il dato in entrata appartiene formando "bolle di attivazione" (figura 1.14) che identificano input omogenei. Nello strato di output un neurone soltanto, per ogni input che viene fornito alla rete, deve risultare "vincente", viene scelto in base al suo valore di attivazione (il più grande), assumerà quindi il valore 1, mentre tutti gli altri assumeranno valore 0 secondo il principio WTA (Winner Takes All). Lo scopo di una rete di Kohonen è quello di ottenere bolle di attivazione adiacenti aventi dati in input con caratteristiche somiglianti.

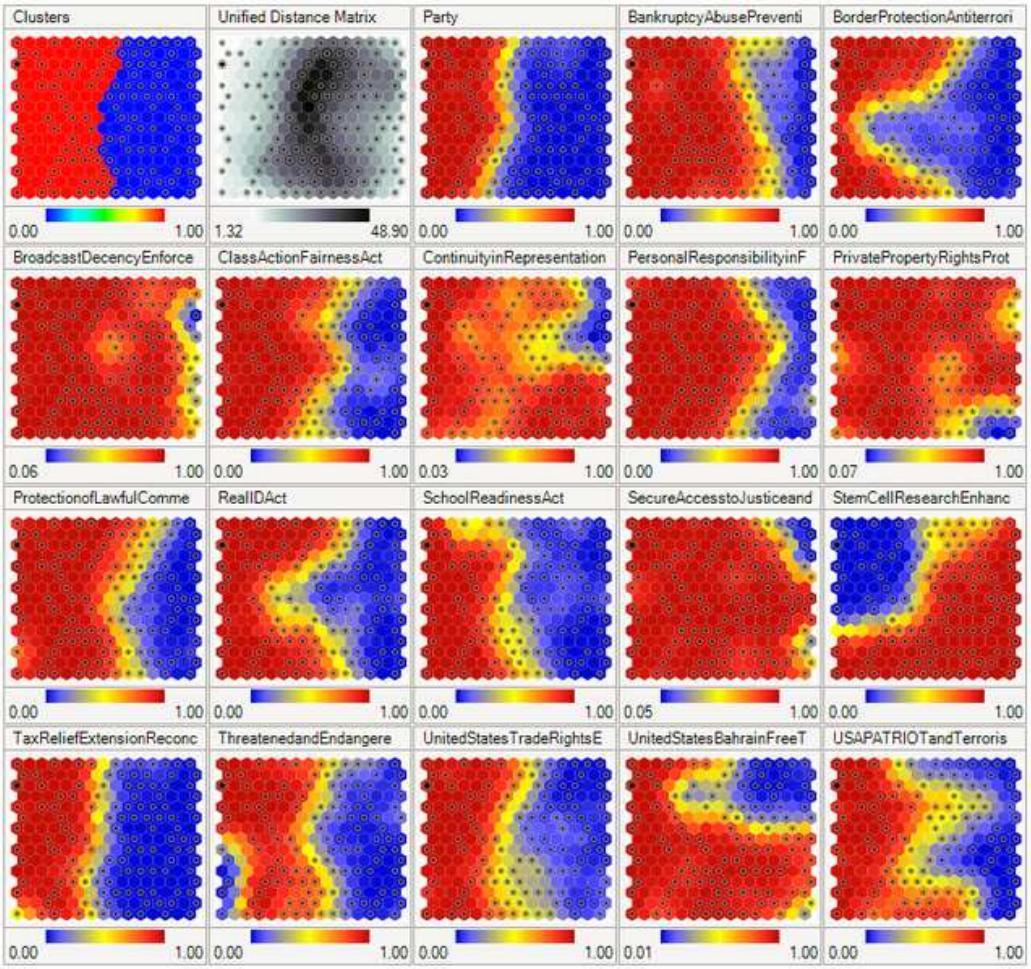

Figura 1.14: Bolle di attivazione.

Capitolo 2

FUNZIONAMENTO DELLE RETI

Questa parte è dedicata al funzionamento delle reti neurali considerandone le differenti funzioni di attivazione e i differenti approcci di apprendimento.

2.1 Funzioni di attivazione

Una delle principali differenze tra le rete neurali consiste nella funzione di attivazione che esse utilizzano. Le reti neurali più semplici (SLP) sono in grado di utilizzare una sola funzione di attivazione posta tra lo strato di input e quello di output, mentre, le reti più complesse, essendo multistrato, hanno la possibilità di utilizzare più funzioni di attivazione anche se, in genere, la funzione di attivazione posta tra lo strato di input e il primo hidden layer è lineare. Ricordiamo che l'output di una rete neurale non è altro che la combinazione lineare dei valori assegnati a ciascun neurone e dei pesi che

collegano quest'ultimi tra i vari strati, nell'equazione 2.1, il potential p è dato dalla sommatoria del prodotto tra i vari neuroni x_i e tali pesi di connessione w_i.

$$p = \sum_{i=1}^{n}(x_i w_i + b_0) \qquad \text{dove } b_0 \text{ è il valore di bias} \tag{2.1}$$

Una volta calcolato quest'ultimo, l'output osservato $\hat{y}$ della rete neurale sarà in funzione dell'algoritmo di attivazione cioè:

$$\hat{y} = \sum_{i=1}^{n}\sigma(p) \tag{2.2}$$

Dall'equazione 2.1 e 2.2 otteniamo:

$$\hat{y} = \sum_{i=1}^{n}\sigma(x_i w_i + b_0) \tag{2.3}$$

Le principali funzioni di attivazione sono:

- funzione di attivazione di soglia (o threshold);
- funzione di attivazione sigmoidale;
- funzione di attivazione a tangente iperbolica;
- funzione di attivazione lineare rettificata (o ReLU).

2.1.1 Funzione di soglia

Il percettrone a singolo strato, prendendo un insieme di input binari $x = [x_1, \ldots, x_n]$ è in grado di restituire un solo output binario, tuttavia, le informazioni in entrata possono avere differente valenza, perciò viene assegnato

un peso $w = [w_1, \ldots, w_n]$ ad ogni input. Ciò comporta che, se l'informazione ponderata è maggiore di un certo threshold, la risposta in quanto binaria sarà 0 oppure 1. La funzione di attivazione di soglia (figura 2.1) quindi, esattamente come un neurone biologico, può essere vista come una funzione che assegna un valore di output a seconda della funzionalità del neurone considerandolo come acceso (valore 1) oppure spento (valore 0). Il valore dell'output di una rete neurale che utilizza tale funzione di attivazione sarà:

$$y = \begin{cases} 0 & \text{se p} < 0 \\ 1 & \text{se } p \geq 0 \end{cases} \tag{2.4}$$

Tale funzione di attivazione non è indicata per problemi complessi perchè in grado di assegnare al valore di output solo un valore binario (0,1).

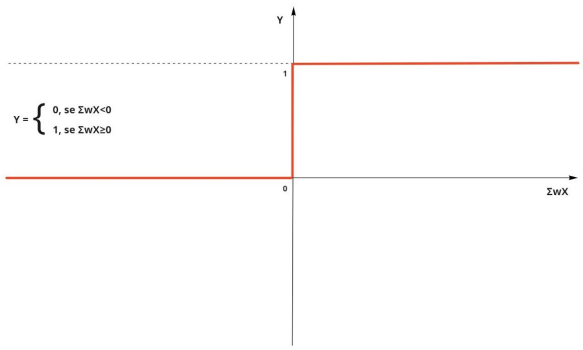

Figura 2.1: Funzione threshold.

Anche le più piccole variazioni dei valori di bias o dei pesi, considerando il modo in cui tale funzione di attivazione è costruita, potrebbero causare l'accensione di un neurone rimasto sempre spento fino ad un attimo prima, generando una reazione tuttaltro che graduale del valore dell'output.

2.1.2 Funzione sigmoidale

Considerando le problematiche date dalla funzione di soglia era necessaria una funzione di attivazione che rendesse il risultato della rete neurale più graduale (smoother) e che quindi non dovesse causare il "brutale" spegnimento o accensione del neurone ma regolarne l'intensità su una scala da 0 ad 1 a seconda dell'importanza dell'informazione. Si è iniziato ad utilizzare la funzione di attivazione sigmoidale mostrata in figura 2.2.

$$y = \frac{1}{1 + e^{p}} \tag{2.5}$$

La funzione riesce quindi a mantenere le caratteristiche limite del single layer perceptron perchè comunque compresa tra [0,1] ma risulta essere più utilizzabile perchè avente un comportamento più graduale. Questa funzione attraverso bias e pesi può essere modulata in modo tale da ottenere progressivamente la risposta corretta, inoltre, la funzione risulta ottima dal punto di vista della derivazione il che le conferisce un ulteriore punto di forza. La derivazione di una funzione sigmoide risulta essere analiticamente semplice e dal punto di vista computazionale (tempo e memoria utilizzati) efficiente; sebbene la funzione sigmoidale sia ampliamente utilizzata nel deep learning per problemi non risolvibili linearmente, purtroppo, è soggetta a problemi di saturazione dell'output: valori in entrata lontani dallo 0 (molto alti o molto bassi) fanno si che il gradiente di discesa della funzione di perdita dell'output si avvicini molto allo 0 portando a tempi di convergenza a volte molto lunghi. Tale problema, diventa ancora più grave man mano che ci spostiamo tra gli strati a causa degli effetti combinati della saturazione su più livelli (problema

del gradiente di fuga o vanishing gradient problem).

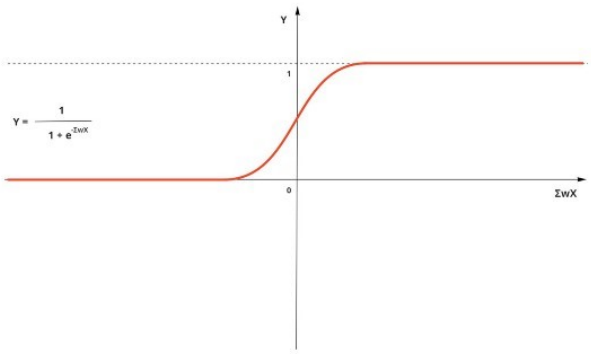

Figura 2.2: Funzione sigmoide.

2.1.3 Funzione a tangente iperbolica

La hyperbolic tangent function o tanh (figura 2.3) è una funzione di attivazione molto simile alla sigmoidale (in quanto è una funzione sigmoidea in scala) con l'unica differenza che il suo codominio ha un intervallo di output tra [-1, +1] ed è definita come:

$$y = \frac{1 - e^{-2p}}{1 + e^{-2p}} \tag{2.6}$$

Risulta essere preferibile alla sigmoidale perchè in tanh la discesa del gradiente è più rapida rispetto alla controparte non lineare in oltre l'imput negativo verrà considerato come fortemente negativo mentre quello posititvo come fortemente positivo anche se, tale caratteristica fa si che spesso la funzione sia soggetta al problema di saturazione dell'output.

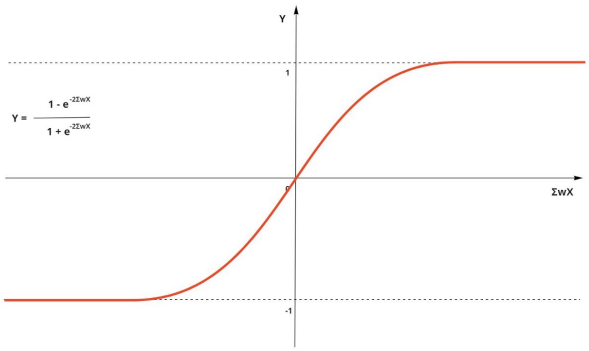

Figura 2.3: Tangente iperbolica.

2.1.4 Funzione lineare rettificata

La rectified linear unit (ReLU) o funzione lineare rettificata (figura 2.4), è una funzione lineare a tratti che emette un segnale di input se e solo se positivo:

$$y = \max\left(0, \sum_{i=1}^{n}(x_i w_i)\right) \tag{2.7}$$

E' divenuta la funzione di attivazione più utilizzata all'interno delle reti neurali non solo per le sue analogie con il neurone biologico ma anche per le sue caratteristiche matematiche, una ANN che utilizza tale funzione sarà più facile da addestrare e di conseguenza avrà prestazioni migliori perchè in una funzione lineare non si fanno trasformazioni. Le funzioni non lineari sono preferite nel caso in cui le strutture dei dati sono più complesse, le funzioni di attivazione non lineari più utilizzate sono la sigmoidale, la softmax (variante della sigmoidale) e la tangente iperbolica (o tanh) ma spesso il loro output satura, il risultato spesso si ferma al valore 0 o 1 nel caso della sigmoidale e -1, +1 nel caso dell'iperbolica, inoltre, le funzioni sono realmente sensibili solo ai cambiamenti intorno al punto medio del loro input, come 0,5 per

sigmoide e 0,0 per tanh, una volta saturato, diventa difficile per l'algoritmo di apprendimento continuare ad adattare i pesi per migliorare le prestazioni del modello.

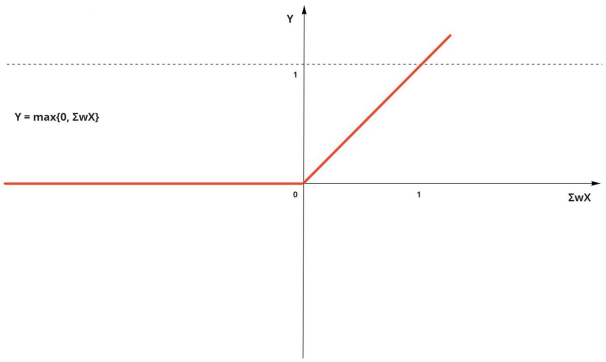

Figura 2.4: Funzione ReLU.

Nel deep learning la funzione ReLU è preferita alle altre a causa della facilità di calcolo del gradiente di discesa e di conseguenza alla "semplicità" di aggiornamento dei pesi con il back propagation algorithm. Tale funzione riesce a garantire una elevata efficienza computazionale grazie alla sua linearità ma allo stesso tempo di trattare i dati con una certa complessità. Spesso, le reti che utilizzano la funzione rettificata per i livelli nascosti vengono chiamate reti rettificate, tali modelli di deep learning, tuttavia, utilizzano la ReLU solo per i livelli nascosti poiché induce la scarsità cioè in un certo numero di neuroni si ritrovano valori nulli o missing values, quando gli strati nascosti sono esposti a più valori di input, la funzione porterà a più zeri con conseguente attivazione di meno neuroni diminuendo il numero di interazioni all'interno della rete neurale (dying ReLU) limitando però la regolarizzazione tramite drop out. Per controllare tale conseguenza alcune reti utilizzano versioni dif-

ferenti della ReLU come ad esempio la leaky ReLU la quale, come mostrato in figura 2.5, restituisce un piccolo valore nonostante l'input sia minore di 0.

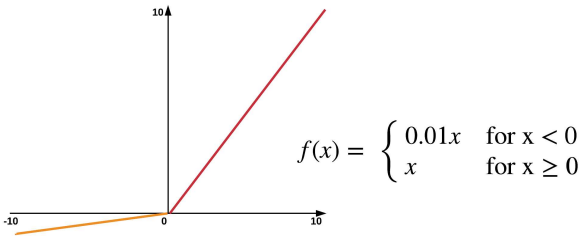

Figura 2.5: Leaky ReLU.

Quindi l'adozione di tale funzione di attivazione è considerabile come una pietra miliare del deep learning in quanto ha consentito l'utilizzo di reti neurali molto profonde.

2.2 Apprendimento

Dall'arrivo del single layer perceptron il neurone possiede pesi variabili i quali vengono ottimizzati (o calibrati) tramite addestramento al fine di diminuire l'errore uscente dalla rete. Nelle reti feedforward l'informazione (features) viaggia all'interno della rete dagli strati di input a quelli di output, in ogni layer le features sono ponderate per ogni neurone e, la funzione di attivazione calcola l'output dello strato precedente restituendolo in input al successivo, definiamo quindi l'informazione ponderata ricevuta dall' l-esimo strato come:

$$a^l = \sigma(z^l) = \sigma(w^l a^{l-1} + b^l) \tag{2.8}$$

Cioè con z^l (output del precedente layer a^{l-1} ponderato per il peso w^l e considerando un valore di bias b^l) moltiplicato per una determinata funzione di attivazione σ, ma come avviene l'apprendimento (aggiornamento dei pesi) di una rete neurale ?

L'addestramento di una rete neurale a strato singolo è differente rispetto a quello di una rete a due strati o multi strato, la prima è calibrabile con algoritmi di addestramento non supervisionati come l'apprendimento hebbiano o il competitivo (tipico della rete SOM) modelli più complessi hanno bisogno di algoritmi di addestramento supervisionati come ad esempio la discesa del gradiente (utilizzato all'interno del back propagation) o l'apprendimento stocastico.

2.2.1 Apprendimento hebbiano

Questo apprendimento è stato proposto da Hebb nel 1949, si basa sulla regolazione correlativa dei pesi. Le coppie di pattern di input e output sono associate a una matrice di pesi W la quale è aggiornata in modo iterativo in base alla differenza tra output atteso (y) ed osservato ($\hat{y}$) finchè risulti prossima allo 0, cioè, la regola di apprendimanto risulta essere:

$$w_j^{k+1} = w_j^k + \eta(y_i - \hat{y}_i^k)x_{ij} \tag{2.9}$$

Dove k si riferisce al numero dell'iterazione, il parametro η è detto learning rate (tasso d'apprendimento), quindi x_{ij} sarà il neurone i associato al peso j, quindi, il peso all'iterazione successiva sarà in funzione del peso all'iterazione attuale e a Δ y. L'aggiornamento dei pesi è in funzione di Δy e dell'errore e:

- se $y \approx \hat{y} \rightarrow e \approx 0$, non ci sarà necessità di aggiornare;
- se $y < \hat{y} \rightarrow e < 0$, $\hat{y}$ deve essere decrementato quindi vanno diminuiti i pesi;
- se $y > \hat{y} \rightarrow e > 0$, $\hat{y}$ deve essere incrementato quindi vanno aumentati i pesi.

2.2.2 Apprendimento competitivo

L'apprendimento competitivo è un tipo di apprendimento non supervisionato tipico delle reti di Kohonen. E' detto competitivo perchè esiste un neurone chiamato vincente che risulta essere il migliore a rappresentare i dati in input

cioè la tecnica di apprendimento per i pesi in una mappa di Kohonen si basa sulla competizione tra i neuroni di output per l'assegnazione dei vettori di input. Per ogni nodo di input assegnato, viene selezionato un neurone di output vincente in base ad una funzione di distanza, i pesi del neurone vengono aggiornati in modo tale da rappresentare al meglio i dati in input, rendendolo più simile ai valori dei coefficienti dell'input si "avvicinano" anche i neuroni vicini secondo il principio per cui concetti simili sono memorizzati in aree adiacenti. Le fasi attraverso le quali avviene l'apprendimento sono:

1. **Inizializzazione:** Sia K il numero di iterazioni dell'algoritmo e poniamo $k = 0$, dopo aver scelto la dimensione della griglia di output, si inizializza ogni peso in modo casuale;

2. **Selezione di un vincitore:** Per ogni neurono in input x_j si seleziona un neurone output vincitore i^* tale che si minimizzi la distanza euclidea $\|x_j - w_i^k\|$ tra il vettore in input x_j ed il peso w_i associato al vettore di output;

3. **Aggiornamento dei pesi:** sia $N(i^*)$ l'intorno del vettore di output vincente, per ogni vettore di output i tale che $i \in [N(i^*), i^*]$, i pesi sono aggiornati secondo la formula $w_i^{k+1} = w_i^k + \eta(x_j - w_i^k)$, si noti che il meccanismo di apprendimento è simile al quello hebbiano con la differenza che qui non si cerca di minimizzare un errore dato dalla differenza di differenti output ma una distanza, in questo modo si aggiornano solo i pesi dei neuroni di output vicini al vincente;

4. **Normalizzazione dei pesi:** Dopo l'aggiornamento i pesi sono normalizzati in modo che siano in scala rispetto ai valori dati in input;

5. **Loop:** i precedenti passaggi sono ripetuti cioè $k = k + 1$ finchè si raggiungono un numero massimo di iterazioni o ci sia un criterio di arresto.

Decidere quando siamo abbastanza vicini a un insieme stabile di centroidi e quindi quando arrestare il tutto è una questione importante. Idealmente, l'iterazione dovrebbe continuare fino a quando non si verifica la convergenza, cioè fino a quando i vettori di riferimento non cambiano o cambiano molto poco, il tasso di convergenza dipenderà da una serie di fattori, come i dati ed η. Si consideri anche che in generale, la convergenza può essere lenta e non garantita, quindi, potremmo migliorare l'algoritmo introducendo un differente intorno cioè dopo aver identificato i^*, potremmo ricalcolare il tutto considerando solo i pesi rilevanti, oppure, potremmo utilizzare algoritmi sensibili e dipendenti alle allocazioni passate il che ci permetterebbe di risolvere un tipico problema comune alle clusterizzazioni non gerarchiche cioè cluster troppo popolati rispetto agli altri.

2.2.3 Discesa del gradiente

Una regressione lineare $f(x)$, considerando un valore input x, restituisce un valore output $\hat{y}$ tale che può essere confrontato con un valore y reale, quindi, potremmo immaginare una rete neurale come una regressione avente una funzione di costo tale da minimizzare la differenza tra i due. Consideriamo quindi una funzione di costo quadratica (come ad esempio l'entropia la quale

viene utilizzata nei problemi di classificazione) del tipo:

$$C(w,b) = \frac{1}{2n}\sum_{i=1}^{n}\|y_i - \hat{y}_i\|^2 \tag{2.10}$$

Il metodo di discesa del gradiente (figura 2.6) permette di trovare un minimo locale in uno spazio a N dimensioni, viene utilizzato per minimizzare la funzione di costo utilizzata, ad ogni iterazione, il procedimento garantisce la diminuzione della funzione obiettivo ($C_l < C_{l-1}$).

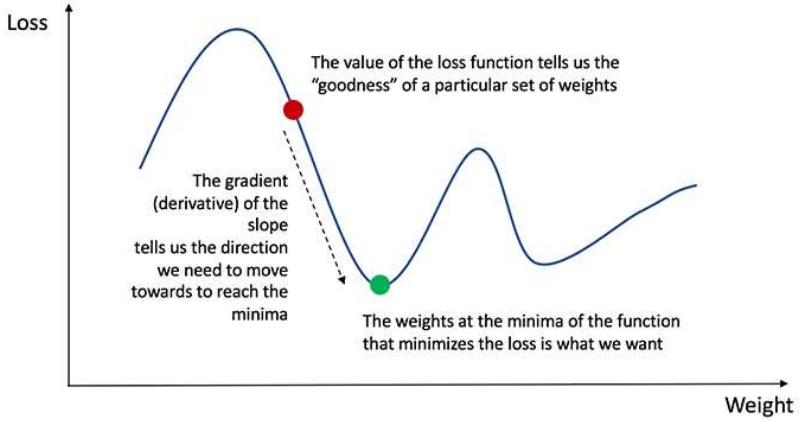

Figura 2.6: Algoritmo di discesa del gradiente.

Assumendo per semplicità che non vi siano valori bias, una piccola variazione nei pesi implica una proporzionale variazione della funzione di costo:

$$\Delta C \approx \sum_{i=1}^{N_l}\frac{\partial C}{\partial w_i}\Delta w_i \qquad \text{con } N_l \rightarrow \text{numero di neuroni nell'layer l} \tag{2.11}$$

Semplificando ulteriormente, assumendo che la nostra funzione di costo di-

penda dalla variazione di soli 2 pesi:

$$\Delta C \approx \frac{\partial C}{\partial w_1}\Delta w_1 + \frac{\partial C}{\partial w_2}\Delta w_2 = \nabla C' * \Delta w \tag{2.12}$$

Il nostro scopo è la minimizzazione di ΔC, quindi, l'ideale sarebbe rendere tale differenza negativa, potendo agire solo su Δw, possiamo utilizzare un coefficiente strettamente positivo η tale che $\Delta w = -\eta \nabla C$, di conseguenza:

$$\Delta C \approx -\eta \nabla C' \nabla C = -\eta \|\nabla C\|^2 \leq 0 \tag{2.13}$$

Questo è il punto di forza dell'algoritmo, ΔC sarà sempre negativo cioè ad ogni iterazione scenderà fino ad arrivare circa allo 0, però, qui l'algoritmo mostra un punto debole, cioè, raggiunto un punto di minimo non c'è la certezza che sarà un minimo globale, potrebbe essere un semplice minimo locale quindi non essere la soluzione più efficiente. Il parametro η ha un forte impatto sul tempo di convergenza dell'algoritmo (figura 2.7) in quanto se troppo grande l'algoritmo è incapace di raggiungere un punto di minimo, mentre, se troppo piccolo avremo dei tempi di convergenza troppo lunghi.

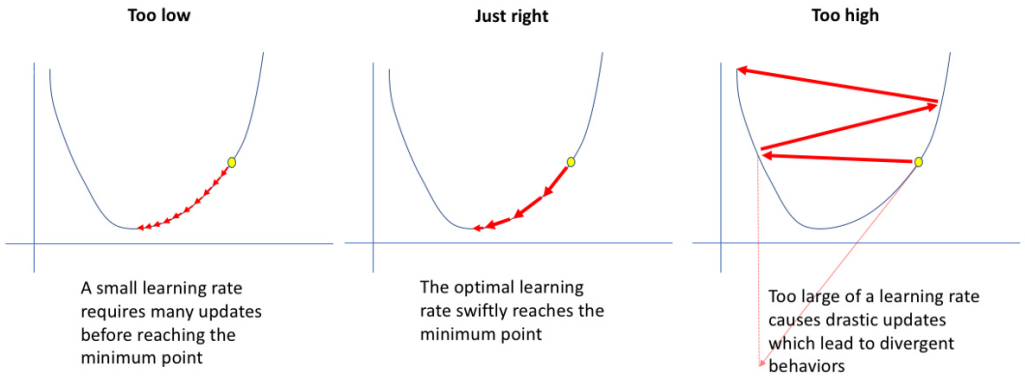

Figura 2.7: differenze tra un coefficiente di apprendimento η piccolo e grande.

Per ottenere il gradiente totale dobbiamo prima calcolare il gradiente per ogni singola osservazione e poi farne una media:

$$\nabla C = \frac{1}{n} \sum_{i=1}^{n} \nabla C_i \tag{2.14}$$

Ciò vuol dire che ogni strato avrà una matrice di pesi composta da w_k e una matrice di valori bias composta da b_h tali che:

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \quad ; \quad b_h \to b'_h = b_h - \eta \frac{\partial C}{\partial b_h} \tag{2.15}$$

Questo ha un costo computazionale enorme, una soluzione a ciò è lo Stochastic Gradient Descent (SGD)(figura 2.8) dove m osservazioni sono raggruppate randomicamente in mini-batch, l'algoritmo di discesa non si applica quindi a tutte le osservazioni ma solo alle selezionate, se il mini-batch è abbastanza amplio si può pensare che la media dei ∇C_i (con i = 1,..., numero totale dei mini-batch) sia approssimabile al ∇C complessivo, quindi, dall'algoritmo otterremo solo un intorno del punto di minimo ma lo avremo in tempi molto più brevi. Di conseguenza pesi e bias risulteranno essere:

$$w_k \to w'_k = w_k - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C_i}{\partial w_k} \quad ; \quad b_h \to b'_h = b_h - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C_i}{\partial b_h} \tag{2.16}$$

I valori del mini-batch m saranno quindi aggiornati considerando i loro precedenti valori ottimizzati al $m - 1$. L'algoritmo itera utilizzando sempre lo stesso valore di η (a meno che non vi sia una regola di updating la quale è presente nelle versioni più nuove), una volta che ogni mini-batch viene analizzato, si dichiara completata ciò che è definita un' "epoca", in genere af-

finchè il risultato sia l'ideale si procede per altre epoche al fine di minimizzare ulteriormente la funzione di costo.

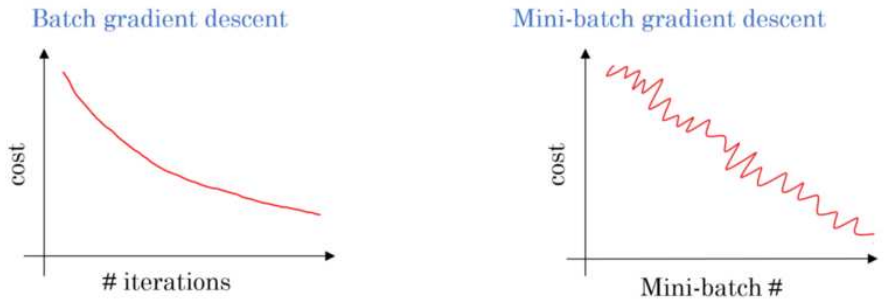

Figura 2.8: Stochastic gradient descent.

Ora si pone un ulteriore problema, quando dovremmo fermare l'algoritmo ? Considerando la funzione di costo e le caratteristiche della nostra ANN non è detto che nonostante il gradiente sia prossimo allo 0 la soluzione attuale sia la più efficiente, potremmo trovarci in prossimità di un punto di minimo migliore ma proseguendo potremmo imbatterci in elevati costi computazionali e di tempo, quindi, vengono fissate un numero di epoche massime prima di fermare il tutto, si presentano però ulteriori inconvenieti:

- il valore delle epoche è un altro iperparametro[1] a cui fare attenzione
- problema della convergenza prematura (premature converge)

La convergenza prematura è uno degli aspetti più complicati nell'addestramento delle reti neurali, non solo dobbiamo gestire il rischio di rimanere bloccati su un minimo locale, ma potremmo anche finire con una soluzione

[1] In statistica bayesiana, un iperparametro è un parametro di una distribuzione di probabilità a priori, è considerabile come un parametro gerarchicamente superiore agli altri cioè non risolve un probema di ottimimizzazione ma influenza i parametri che possono farlo in un rapporto unidirezionale con quest'ultimi.

finale che non è dal punto di vista matematico un minimo in senso formale, inoltre, reti molto grandi potrebbero imbattersi nel vanishing gradient problem dove la discesa del gradiente è prossima allo 0 a causa della saturazione dell'output tipico delle funzioni di attivazione non lineare (ma ricordiamo che tale problema è risolvibile utilizzando una ReLU). Alcuni approcci recenti affrontano questo problema attraverso l'utilizzo di algoritmi di ottimizzazione che non utilizzano la discesa del gradiente ma in genere richiedono un costo computazionale molto elevato. La propagazione feedforward permette di portare l'informazione dagli strati di input a quelli di output assumendo che pesi e valori bias siano noti a priori. L'algoritmo di backpropagation permette l'aggiornamento di questi pesi.

2.2.4 Backpropagation algorithm (BackProp)

Dallo studio di Rosenblatt, Hinton e Williams nel 1986 si sviluppò un algoritmo che permise il calcolo rapido delle derivate parziali necessarie per approssimare la variazione della funzione di costo ad ogni iterazione dell'algoritmo di discesa del gradiente. L'addestramento di tipo backpropagation di una rete neurale avviene in due diversi stadi, forward-pass e backward-pass:

- Nella prima fase (forward-pass) i vettori in input sono applicati ai nodi in ingresso con una propagazione in avanti dei segnali attraverso ciascun livello della rete, durante questa fase i valori dei pesi sinaptici rimangono invariati.

- Nella seconda fase (backward-pass) la risposta della rete viene confrontata con l'uscita desiderata ottenendo il segnale di errore, l'errore calcolato è propagato nella direzione inversa rispetto a quella delle connessioni sinaptiche.

I pesi sinaptici sono modificati in modo da minimizzare la differenza tra l'output attuale e l'output desiderato, nel mentre si torna ai nodi di input, i pesi vengono calibrati, il processo è mostrato in figura 2.9.

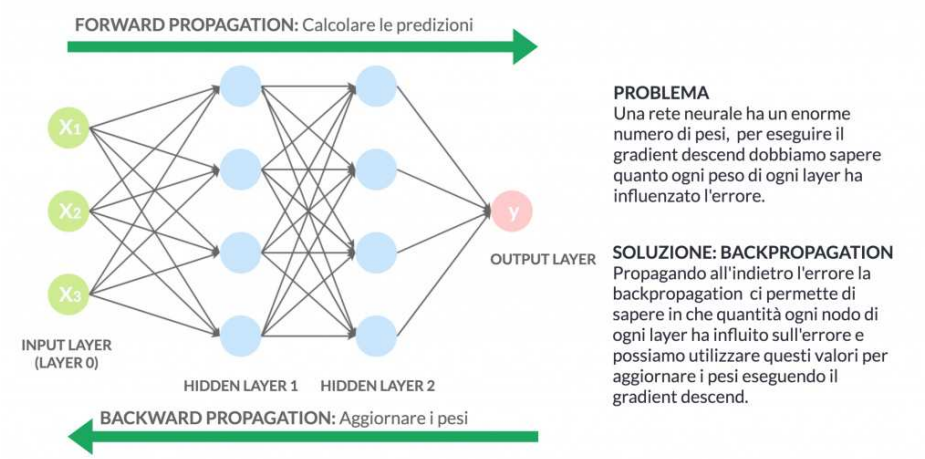

Figura 2.9: Rappresentazione del backpropagation algorithm.

La backpropagation è stata il vero fattore scatenante per l'applicazione delle ANN a problemi empirici, prima della sua comparsa, i calcoli utilizzati nella calibrazione delle reti neurali erano quasi impossibili a causa dell'elevata difficoltà computazionale. Considerando l'equazione 2.8, BackProp si basa su quattro equazioni fondamentali che devono essere risolte in sequenza per calcolare il gradiente dei parametri:

$$1) \quad \delta_i^L \equiv \frac{\partial C}{\partial z_i^L} = \frac{\partial C}{\partial a_i^L}\sigma'(z_i^L) \tag{2.17}$$

Misura dell'errore nel i-esimo neurone dell'output layer, esprime quanto del cambiamento di ΔC è dovuto a piccoli cambiamenti nella funzione di attivazione del neurone di output, quindi, se la forma di σ' fosse conosciuta, potremmo calcolare tali derivate in maniera esplicita, in forma matriciale la funzione è riscrivibile come: $\delta^L = \nabla_a C \bigodot^2 \sigma'(z^L)$, considerando la funzione quadratica

[2]Il prodotto di Hadamard (o prodotto abrasivo) è un prodotto fra due matrici con lo stesso numero di righe e colonne, ogni elemento risultante è dato dal prodotto degli elementi corrispondenti nelle due matrici.

di costo $\frac{\partial C}{\partial a^L} = (a^L - y)$ la quale infine diviene $\delta^L = (a^L - y) \bigodot \sigma'(z^L)$.

$$2) \quad \delta^l = (w'^{l+1}\sigma^{l+1}) \bigodot \sigma'(z^L) \tag{2.18}$$

Espressione dell'errore nel layer corrente l rispetto all'errore del layer successivo $l+1$.

$$3) \quad \frac{\partial C}{\partial b_i^l} = \delta_i^l \tag{2.19}$$

Tasso di cambiamento del costo totale rispetto alla variazione del valore di bias (derivata parziale nella discesa del gradiente rispetto a b_i nel layer l della funzione di costo).

$$4) \quad \frac{\partial C}{\partial w_{ik}^l} = a_k^{l-1}\delta_i^k \tag{2.20}$$

Tasso di cambiamento del costo totale rispetto alla variazione dei pesi, quindi, il vero risultato dell'algoritmo sono le derivate parziali nell'equazioni 2.19 e 2.20, ma, tali risultati non sarebbero ottenibili senza le equazioni 2.17 e 2.18, quindi, con queste quattro equazioni possiamo utilizzare il meccanismo di back propagation al fine di addestrare la rete ovvero possiamo trovare i pesi e le distorsioni in ogni livello che minimizzino la funzione di costo totale, il processo può essere suddiviso nei seguenti cinque passaggi:

1. **Input layer**: calcolare la funzione di attivazione a^1 dell'input layer (spesso è lineare).

2. **Feedforward step**: facendo fluire l'informazione dall'input layer all' output layer si calcolano $z^l = w^l a^{l-1} + b^l$ e $a^l = \sigma(z^l)$ con $l = (2, \ldots, L)$ cioè calcoliamo le funzioni di attivazione in ogni singolo layer il che

ci serve al fine di calcolare la derivata parziale della funzione di costo totale rispetto ai pesi.

3. **Output error** δ^L: calcoliamo $\delta^L = \nabla_a C \bigodot \sigma'(z^L)$.

4. **Error backpropagation**: calcoliamo $\delta^l = (w'^{l+1}\delta^{l+1}) \bigodot \sigma'(z^l)$ con $l = (L-1, L-2, \ldots, 2)$, questo errore è calcolato non dall'ultimo strato ma dal precedente ($l = L-1$) cioè si torna indietro fino al layer $l = 2$, questo step da il nome all'algoritmo in quanto l'informazione parte dal fondo per ternare all'inizio della rete.

5. **Output**: il gradiente della funzione di costo è dato da $\frac{\partial C}{\partial w^l_{ik}} = a^{l-1}\delta^l_i$ e $\frac{\partial C}{\partial b^l_i} = \delta^l_i$

Si noti che questi cinque passaggi si riferiscono a una sola osservazione, la backpropagation è solo uno step (sicuramente il più importante) della procedura di addestramento della rete.

2.2.5 Apprendimento stocastico

Il fine dell'algoritmo di backpropagation è la minimizzazione dell'errore dato dalla differenza tra y atteso e da $\hat{y}$ osservato cioè la minimizzazione della funzione di costo, però, potremmo utilizzare anche metodi stocastici cioè processi basati su leggi probabilistiche e non deterministiche al fine di ottimizzare tale funzione. A parità di fine però, c'è una differenza sostanziale tra i due metodi, il BackProp utilizzando la discesa del gradiente minimizza la funzione di costo quadratica in un punto di minimo locale mentre i metodi stocastici nel punto di minimo globale dove vi è massima ottimizzazione dell'obiettivo desiderato. I metodi di ottimizzazione stocastica vengono applicati quando i metodi enumerativi sono troppo costosi. Questo è genericamente il caso dei problemi di ottimizzazione ad alta dimensione, dove il numero totale di configurazioni possibili cresce esponenzialmente con il numero di variabili. I recenti progressi nel campo della calibrazione delle ANN hanno affrontato l'utilizzo di tecniche di ottimizzazione euristica, come ad esempio lo Swarm Optimization (Ottimizzazione dello sciame) o SO, si consideri che esistono vari algoritmi di questo genere e sono tutti ispiarti a meccanismi di ottimizzazione biologica:

- l'ottimizzazione del formicaio
- l'ottimizzazione dello sciame di particelle
- l'algoritmo delle api
- la ricerca della diffusione stocastica

Queste tecniche si basano sulla ricerca probabilistica dell'ottimo e hanno il vantaggio di saltare completamente la fase di backpropagation (saltando quindi i calcoli matriciali), tuttavia, questi algoritmi hanno la tendenza a chiamare la funzione di costo un numero significativo di volte cioè nel caso delle reti neurali a ripetere più volte lo step di feedfarward, quindi, dal punto di vista computazionale hanno tempi di convergenza più lunghi, questo può rappresentare un problema ma in parte superabile attraverso efficienti librerie di algebra lineare.

Capitolo 3

DATI E METODI

3.1 Analisi sui pazienti diabetici

Utilizzando una rete neurale e una regressione logistica, prediciamo diagnosi di diabete in pazienti aventi differenti caratteristiche.

3.1.1 Descrizione del dataset

Il diabete (il cui nome più appropriato sarebbe diabete mellito) è una malattia cronica caratterizzata dalla presenza di elevati livelli di glucosio nel sangue (iperglicemia) e dovuta ad un'alterazione nei livelli o nella funzione dell'insulina. L'insulina è l'ormone, prodotto dal pancreas, che consente al glucosio di entrare nelle cellule per poter essere utilizzato come fonte energetica. Esistono due tipologie di diabete:

- **il diabetete di tipo 1:** Riguarda circa il 10% delle persone con diabete e in genere insorge nell'infanzia o nell'adolescenza, è considerabile

a tutti gli effetti una malattia autoimmune in quanto il sistema immunitario aggredisce le cellule pancreatiche β responsabili della secrezione dell'insulina la quale, risultando insufficiente, va somministrata giornalmente. Questo danno, che il sistema immunitario induce nei confronti delle cellule che producono insulina, potrebbe essere legato a fattori ambientali (tra i quali, sono stati chiamati in causa fattori dietetici) oppure a fattori genetici, individuati in una generica predisposizione a reagire contro fenomeni esterni, tra cui virus e batteri, in oltre, si potrebbe trasmettere una predisposizione alla malattia attraverso la trasmissione di geni che interessano la risposta immunitaria, i quali, in corso di una banale risposta del sistema immunitario, causerebbero una reazione anche verso le cellule pancreatiche β, con la produzione di anticorpi diretti contro quest'ultime (auto-anticorpi).

- **il diabete di tipo 2:** Riguarda circa il 90% dei casi di diabete, si manifesta verso i 30/40 anni, a differenza del tipo 1 qui le cellule pancreatiche β rimangono inalterate ma i tessuti del paziente non riescono ad utilizzare l'insulina prodotta a causa di una resistenza all'ormone detta insulino-resistenza la quale porta alla malattia. Alla sua insorgenza sono associati alcuni fattori di rischio tra cui: la familiarità per diabete, lo scarso esercizio fisico, il sovrappeso e l'appartenenza ad alcune etnie. Riguardo la familiarità, circa il 40% dei diabetici di tipo 2 ha parenti di primo grado (genitori, fratelli) affetti dalla stessa malattia, mentre nei gemelli monozigoti la concordanza della malattia si avvicina al 100%, suggerendo una forte componente ereditaria per que-

sto tipo di diabete. Questa forma può essere controllata attraverso uno stile di vita più sano ed esercizio fisico per di manterere i livelli glicemici a valori normali. Nel caso in cui il cambiamento dello stile di vita non sia sufficiente, potrebbe essere necessario l'assunzione di farmaci o iniezioni di insulina.

Il dataset contiene osservazioni relative a pazienti di sesso femminile e delle 786 istanze complessive, 376 si sono rivelate prive di validità sperimentale perché per alcuni attributi è stato registrato il valore zero al posto dell'osservazione empirica. Di conseguenza si è scelto di proseguire con 392 osservazioni e 9 features le quali sono descritte nella tabella 3.1.

Nome della variabile	Descrizione	Unità di misura
N° gravidanze	Numero di volte in gravidanza	X
Test glucosio	Concentrazione di glucosio plasmatico dopo 2 ore in un test orale di tolleranza al glucosio. Il test di tolleranza al glucosio orale (OGTT), misura la risposta del corpo al glucosio. E' un test medico utilizzato per lo screening del prediabete, del diabete di tipo 2 e dell'insulinoresistenza. Nell'OGTT di 2 ore, la glicemia plasmatica e le concentrazioni di insulina vengono misurate dopo un digiuno e poi dopo l'assunzione orale di glucosio, a intervalli di 30 minuti. Il test di tolleranza al glucosio identifica le anomalie nel modo in cui il corpo gestisce il glucosio dopo un pasto[1].	mg/dl: milligrammi per decilitro.

[1] https://www.mayoclinic.org.

Pressione sanguigna	Pressione sanguigna diastolica, esiste un forte legame tra diabete e pressione arteriosa: recenti studi scientifici hanno evidenziato che circa il 60% delle persone diabetiche è affetto da ipertensione. La correlazione è talmente nota alla comunità scientifica che ai diabetici si suggerisce di controllare regolarmente i valori della pressione arteriosa e, viceversa, agli ipertesi di eseguire tutti gli esami atti ad escludere la presenza di patologie diabetiche. Nei soggetti con diabete di tipo 1, quasi sempre la pressione alta compare dopo l'insorgere della nefropatia diabetica, che compromette le funzionalità renali di circa il 30-40% dei soggetti malati di diabete da molti anni. In questi individui, l'ipertensione può essere considerata un effetto del diabete, ma allo stesso tempo ne accelera l'evoluzione in una sorta di circolo vizioso.	mm Hg: millimetri di mercurio

Pressione sanguigna	Nei malati di diabete di tipo 2, invece, spesso l'ipertensione si manifesta ancor prima della diagnosi di diabete come conseguenza della resistenza insulinica[2].	mm Hg: millimetri di mercurio.
Spessore pelle	Spessore delle pieghe cutanee. L'ispessimento della pelle è frequentemente osservato nei pazienti con diabete, le aree della pelle interessate possono apparire ispessite, cerose o edematose. Questi pazienti sono spesso asintomatici ma possono avere una riduzione della sensibilità e del dolore[3].	mm: millimetri.
Insulina	Insulina sierica. Concentrazione di insulina misurata durante l'OGTT.	mu U/ml: un milionesimo di unità in un millessimo di litro.
BMI	Indice di massa corporea. BMI e insorgenza di diabete sono direttamente proporzionali[4].	peso in kg/(altezza in m)^2.

Pedigree	Funzione che valuta la probabilità di diabete in base alla storia familiare. Quindi fornisce alcuni dati sulla storia della malattia nei parenti e sulla relazione genetica di quest'ultimi con il paziente.	X
Età	misura in anni	X
Diagnosi	I pazienti non diabetici hanno valore 0 mentre i diabetici 1	X

Tabella 3.1: Features contenute nel dataset utilizzato.

[2]https://www.medelinternational.com.
[3]https://www.ncbi.nlm.nih.gov.
[4]https://www.portalediabete.org.

Si riportano nella tabella 3.2 i relativi indici di dispersione.

N° gravidanze	Test glucosio	Pressione sanguigna	Spessore pelle
Min. : 0.0	Min. : 56.0	Min. : 24.00	Min. : 7.00
1st Qu. : 1.0	1st Qu. : 99.0	1st Qu. : 62.00	1st Qu.:21.00
Median : 2.0	Median :119.0	Median : 70.00	Median :29.00
Mean : 3.3	Mean :122.6	Mean : 70.66	Mean :29.15
3rd Qu. : 5.0	3rd Qu. :143.0	3rd Qu. : 78.00	3rd Qu.:37.00
Max. :17.0	Max. :198.0	Max. :110.00	Max. :63.00

Insulina	BMI	Pedigree	Età	**Diagnosi**
Min. : 14.00	Min. : 24.0	Min. : 4.0	Min. :21.00	Min. :0.00
1st Qu.: 76.75	1st Qu.:262.8	1st Qu.: 245.8	1st Qu.:23.00	1st Qu.:0.00
Median :125.50	Median :328.0	Median : 422.0	Median :27.00	Median :0.00
Mean :156.06	Mean :308.0	Mean : 480.7	Mean :30.86	Mean :0.33
3rd Qu.:190.00	3rd Qu.:368.2	3rd Qu.: 667.8	3rd Qu.:36.00	3rd Qu.:1.00
Max. :846.00	Max. :671.0	Max. :2329.0	Max. :81.00	Max. :1.00

Tabella 3.2: Indici di dispersione delle features.

Dalla frequenza assoluta e relativa delle classi contenute nella variabile outcome Diagnosi (tabella 3.3), si deduce un problema di rebalancing dei dati, cioè, i quantitativi di soggetti appartenenti alle classi non sono proporzionati.

Diagnosi	Frequenza assoluta	Frequenza relativa
Sano	262	66.84
Malato	130	**33.16**

Tabella 3.3: Frequenza assoluta e relativa delle classi contenute in Diagnosi.

Il problema è più visibile utilizzando un grafico a torta (figura 3.1).

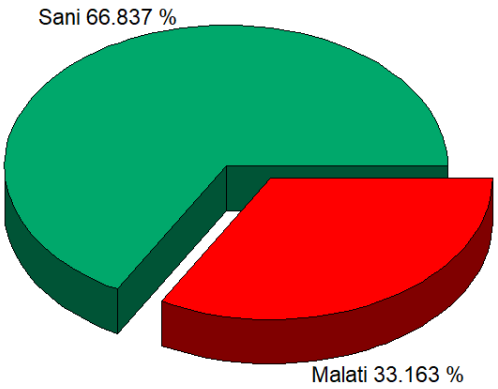

Figura 3.1: Frequenza relativa delle classi Sani e Malati ed evidente problema di rebalancing.

Tramite grafici a violino, combiniamo boxplot e curve di densità delle features (figura 3.2). Nei riquadri le caratteristiche sono confrontate con Diagnosi.

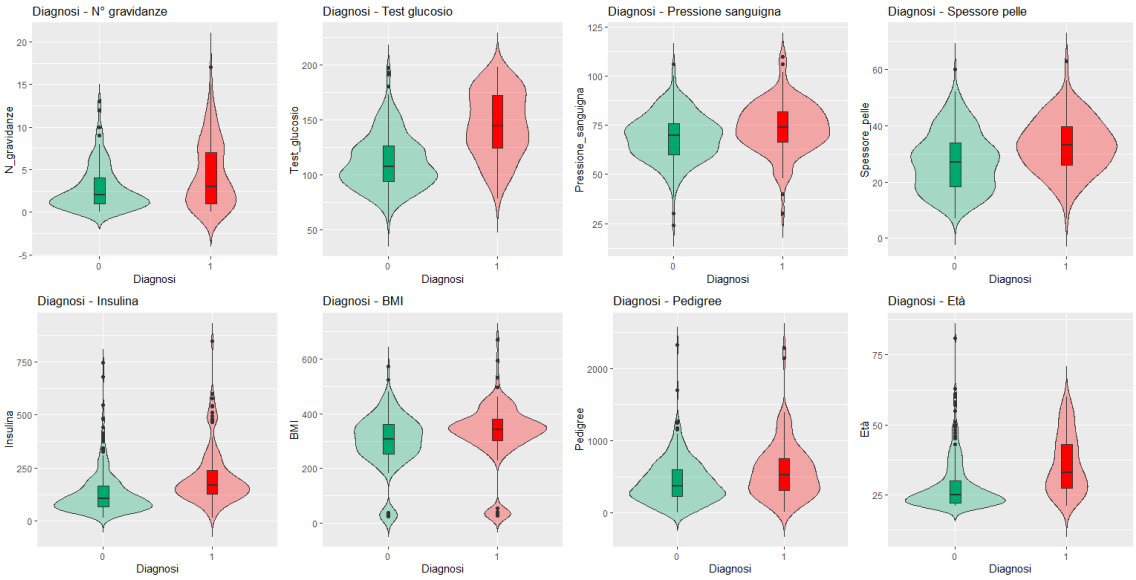

Figura 3.2: Con un grafico a violino è possibile rappresentare in un'unica visualizzazione outliers, IQR (differenza interquartile) e frequenze delle istanze.

Possiamo dedurre che:

- La moda del numero di gravidanze si aggira attorno al primo quartile (con valore 1) ma, nelle pazienti sane è decisamente più alta rispetto alle malate le quali si distribuiscono più uniformemente, sembrerebbe che un aumento del numero di gravidanze porti all' aumento della probabilità di essere diabetici, tuttavia, tale risultato potrebbe essere influenzato dal rebalancing.

- Risulta evidente che i pazienti sani hanno livelli di glucosio più basso rispetto ai malati.

- I pazienti malati sono soggetti ad uno ispessimento della pelle.

- I soggetti presentano livelli di insulina differenti, probabilmente a causa dell'immuno resistenza cioè, è possibile che nei soggetti sani sia più bassa rispetto ai malati perchè in quest'ultimi (se diabetici di tipo 2) viene comunque prodotta dalle cellule β ma non utilizzata.

- Considerando l'indice di massa corporea, la media dei soggetti malati risulta essere nettamente superiore e con una differenza interquartile inferiore la quale ci suggerisce che i soggetti malati hanno un BMI più alto rispetto ai non malati.

- I pazienti sani sono più giovani rispetto ai malati, la causa è attribuibile al fatto che il diabete di tipo 2 in genere si presenta dopo i 30/40 anni di età.

- Dai dati in nostro possesso, non c'è particolare differenza tra le due classi considerando la pressione sanguigna e i casi di insorgenza in famiglia, la differenza delle mode nel box "Diabete - Pedigree" è probabilmente data dalla mancanza di bilanciamento tra le classi.

Si prosegue con un'analisi delle distribuzioni utilizzando:

- **Test Shapiro–Wilk:** determina se una variabile si distribuisce normalmente oppure no partendo dall'ipotesi nulla che lo sia.
  - H0: Distribuzione normale, p-value ≥ 0.5;
  - H1: Distribuzione non normale, p-value < 0.5.
- **Skewness:** indice di asimmetria della distribuzione, è definito come:

$$\gamma_1 = \frac{m_3}{m_2^{3/2}} \tag{3.1}$$

  Dove m_3 e m_2 sono rispettivamente i momenti centrali di ordine 3 e 2 (varianza), puo avere valore:
  - Negativo: coda a sinistra o asimmetrica a sinistra;
  - Positivo: coda a destra o asimmetrica a destra.
- **Eccesso di curtosi:** allontanamento dalla normalità, cioè quanto la distribuzione si discosta dalla curva mesocurtica tipica della normale, la misurazione più nota è quella che utilizza l' indice di Pearson $\beta_2 = \frac{m_4}{m_2^2}$ dove m_4 e m_2 sono rispettivamente il momento centrale di ordine 4 e di ordine 2. L'indice di eccesso di curtosi è calcolabile come:

$$\gamma_2 = \beta_2 - 3 \tag{3.2}$$

Puo avere valore:

- maggiore di 0: la curva si definisce leptocurtica, cioè più "appuntita" di una normale;
- minore di 0: la curva si definisce leptocurtica, cioè più "appuntita" di una normale;
- uguale a 0: la curva si definisce normocurtica (o mesocurtica), cioè "piatta" come una normale.

In figura 3.3 si mostrano le distribuzioni delle variabili in entrata.

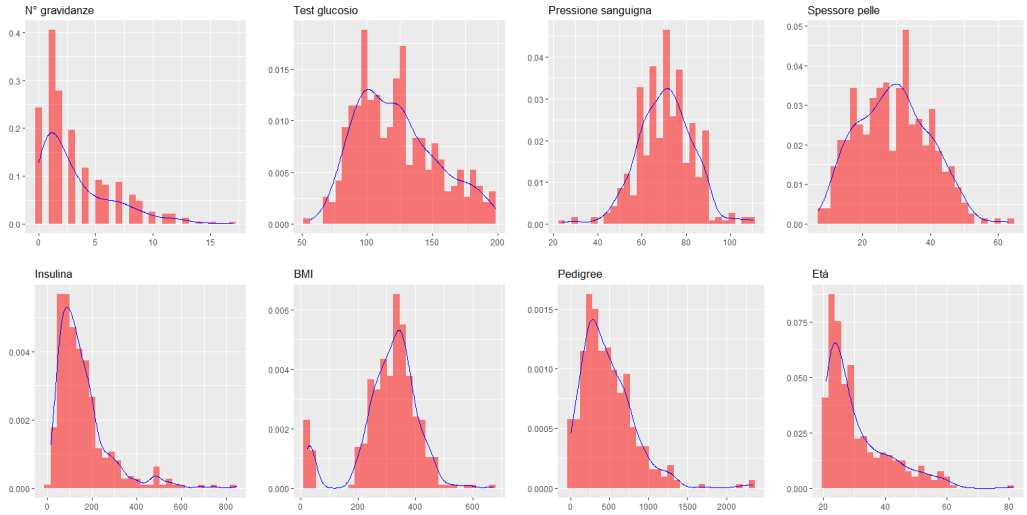

Figura 3.3: Istogrammi rappresentanti le distribuzioni delle varie features, non si riporta la distribuzione della viariabile risposta in quanto dicotomica.

Le rappresentazioni ci conducono alle seguenti considerazioni:

- Nessuna variabile è distribuita normalmente;

- Nessuna feature è simmetrica ma quella che gli si avvicina di più è la pressione sanguigna con $\gamma_1 = -0.08718115$;
- Le distribuzioni sono prevalentemente leptocurtiche.

In statistica due variabili si dicono correlate quando i valori di una variabile x tendono ad essere omogenei ai valori di una variabile y sensa che ci sia un reale rapporto di causa-effetto tra le due (figura 3.4).

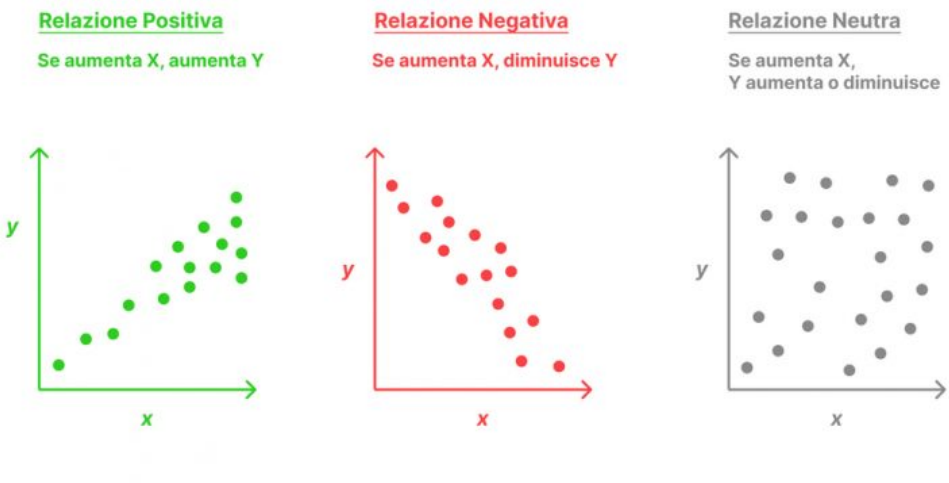

Figura 3.4: Correlazione tra due variabili x e y.

La misurazione più utilizzata è l'indice di correlazione r di Pearson il quale esprime l'esistenza o meno di una relazione lineare tra due variabili statistiche. Date due variabili x e y, è definito come:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{3.3}$$

Cioè tramite il rapporto tra la loro covarianza e il prodotto delle loro deviazioni standard, il coefficiente assume sempre valori [-1,+1]:

- se $\rho_{xy} > 0$ le variabili x e y si dicono correlate positivamente;
- se $\rho_{xy} = 0$ le variabili x e y non sono correlate;
- se $\rho_{xy} < 0$ le variabili x e y si dicono correlate negativamente.

In oltre se correlate si parla di:

- correlazione debole se $0 < |\rho_{xy}| < 0.3$;
- correlazione moderata se $0{,}3 < |\rho_{xy}| < 0.7$;
- correlazione forte se $|\rho_{xy}| > 0.7$.

Utilizzando una matrice di correlazione (figura 3.5) si analizzano le correlazioni tra le nostre variabili:

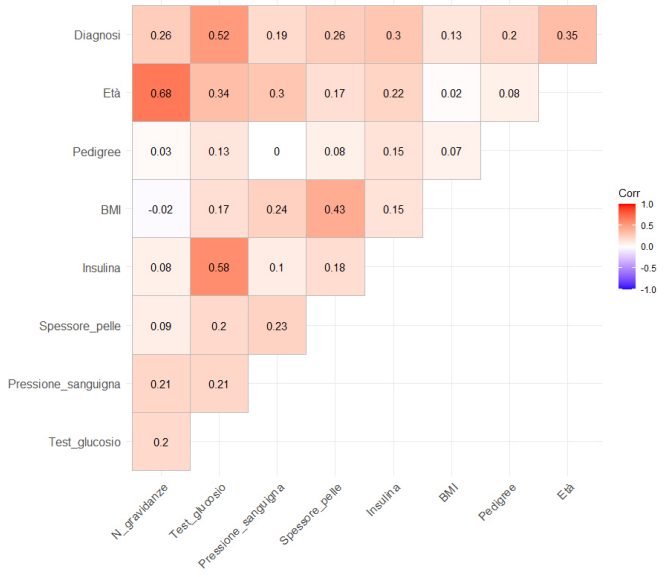

Figura 3.5: Matrice di correlazione del dataset contenete i vari indici r di Pearson tra le variabili, il colore rosso indica una correlazione positiva mentre il blu una negativa (assente nel nostro caso).

Al fine di ridurre la correlazione tra le variabili si ricorre alla principal component analysis (PCA), tecnica di features extraction[5] che scompone il dataset di partenza in vettori (dimensioni) ortogonali tra loro detti componenti principali:

$$\overrightarrow{Pc_i} = \vec{\lambda_i} X \tag{3.4}$$

Con λ vettore degli autovalori e X matrice delle osservazioni originali centrati $(x_i - \mu)$. La prima componente principale (dimensione 1) è identificabile come l'autovettore corrispondente all'autovalore massimo, la seconda all'autovettore corrispondente al secondo autovalore più alto e così via. Applicando la PCA al dataset consideriamo fino alla quarta dimensione che, come mostrato nella tabella 3.4, ha una varianza cumulata sopra il 70% (prenderle tutte equivale a prendere il dataset di partenza).

	PC1	PC2	PC3	**PC4**
Standard deviation	1.56	1.19	1.09	0.96
Proportion of Variance	0.30	0.17	0.15	0.11
Cumulative Proportion	0.30	0.48	0.63	**0.75**

	PC5	PC6	PC7	PC8
Standard deviation	0.85	0.74	0.63	0.54
Proportion of Variance	0.09	0.06	0.05	0.03
Cumulative Proportion	0.84	0.91	0.96	1.00

Tabella 3.4: Cercando di mantere invariato il contenuto informativo originale, la PCA crea otto nuove dimensioni a partire dalle features di partenza.

[5] A differenza delle tecniche di feature selection che creano un subset di dati a partire da uno di partenza con la conseguente perdita di contenuto informativo, le tecniche di feature extraction estraggono da un set di dati caratteristiche totalmente nuove e non correlate tra loro in modo da mantenere la varianza originale invariata.

La figura 3.6 mostra uno spazio bidimensionale creato sulla base delle prime due principal component. A sinistra viene mostrato il contributo delle vecchie features alla determinazione della diagnosi (in giallo le più significative e in blu quelle meno), mentre a destra, le osservazioni rappresentate nello spazio sono clusterizzate sulla base delle nuove dimensioni di analisi.

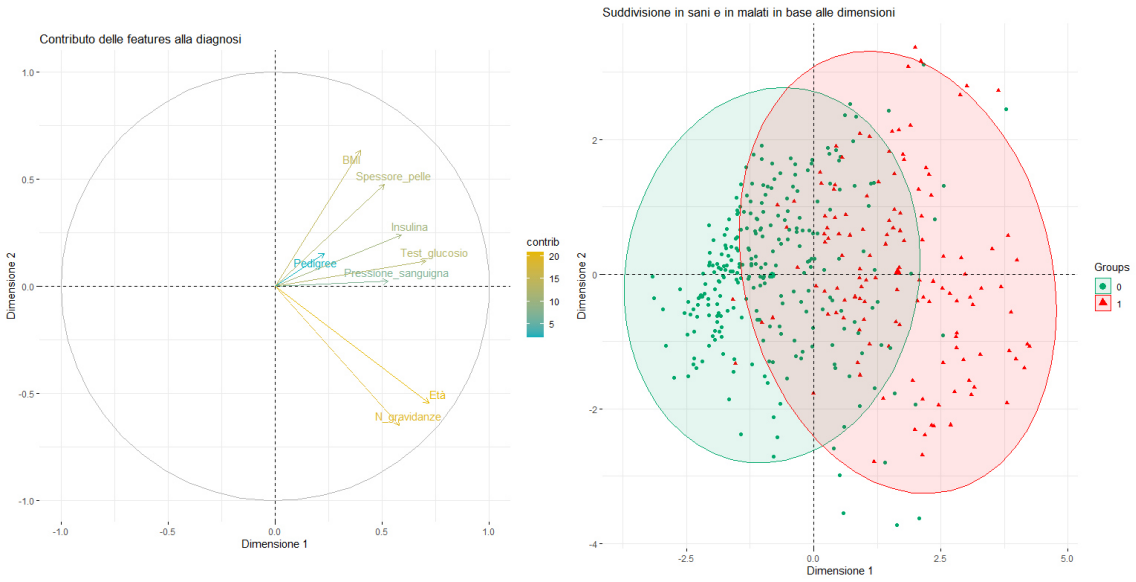

Figura 3.6: Contributo delle features originali alla determinazione della diagnosi (sinistra) e clusterizzazione dei pazienti in sani e malati sulla base delle nuove dimensioni (destra).

Considerando lo scarso contributo delle variabili "Pedigree" e "Pressione_sanguigna" si procede alla creazione di un modello predittivo senza di esse.

3.1.2 Costruzione dei modelli

I modelli costruiti sulla base del dataset normalizzato[6] e con un hold out[7] 80/20 sono:

- Regressione logistica;
- Rete neurale che simula una regressione logistica;
- Multilayer perceptron.

Il modello ottimale è valutato sulla base di metriche calcolate tramite la matrice di confusione dei modelli, cioè, tramite una matrice utilizzata per valutare le prestazioni di una classificazione. Questa matrice è uno strumento per analizzare gli errori compiuti da un modello di classificazione, è detta anche tabella di errata classificazione e restituisce una rappresentazione dell'accuratezza della classificazione. La figura 3.7 mostra la matrice di classificazione binaria con le relative combinazioni.

[6] In statistica, con normalizzazione si intende limitare un insieme di valori x in un intervallo [0,1] creando nuove variabili z tali che $z_i = \frac{x_i - \mu(x)}{\sigma(x)}$ quindi centrate rispetto alla media e divise per la deviazione standard di x.

[7] Per l'addestramento di un modello, nel data mining si suddivide un set di dati in training set e in test set in genere con una percentuale di 70-30% o di 80-20% (holdout) in alternativa, si utilizza la cross validation dove il data set complessivo è suddiviso in k-fold, quest'ultimi sono utilizzati iterativamenente prima singolarmente come test set poi assieme ad altri k-fold come training set, oppure, il bagging dove si creano più training set tramite campionamento con reinserimento (bootstrap) e la restante parte non utilizzata (corrispondente a circa il 37%) come test set (perchè "out of bag"), nel training set si conosce la feature endogena cioè l'output in modo tale da "poter dare qualcosa" al nostro modello su cui basarsi per apprendere mentre il test set è utilizzato per paragonare i risultati ottenuti con i risultati osservati in precedenza.

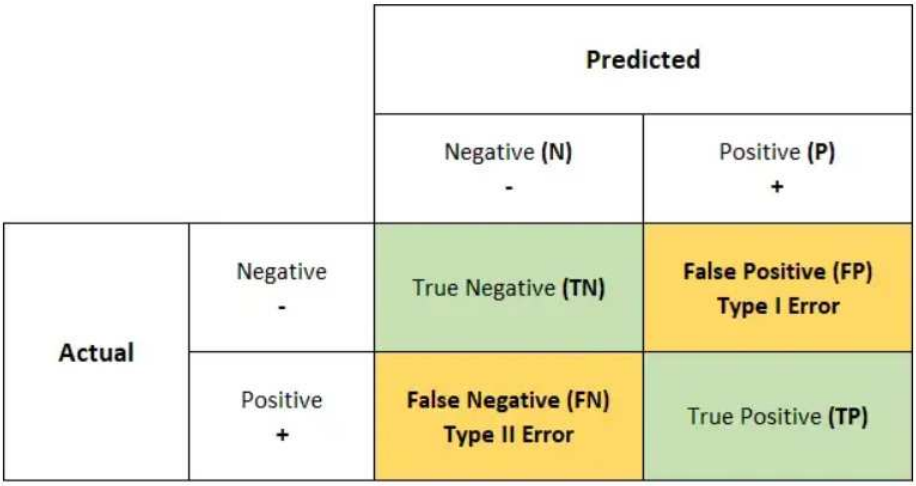

Figura 3.7: Matrice di confusione di un modello di classificazione binario, sulla diagonale principale vengono riportati i casi di corretta classificazione, altrove gli errori di classificazione del modello (FN, FP).

La tabella è suddivisa in:

- TrueNegative (TN): il valore reale è negativo e la previsione è corretta;
- TruePositive (TP): il valore reale è positivo e la previsione è corretta;
- FalsePositive (FP): il valore reale è negativo ma è stato previsto come positivo, è conosciuto anche come errore di prima specie;
- FalseNegative (FN): il valore reale è positivo ma è stato previsto come negativo, è conosciuto anche come errore di seconda specie.

Dalla matrice di confusione ricaviamo le cinque metriche utilizzate con lo scopo di identificare il modello migliore:

- **Accuracy:** l'accuratezza esprime la proporzione di pazienti correttamente classificati, è definita come il rapporto tra le previsioni corrette (somma degli elementi nella diagonale principale della matrice di confusione) e il totale delle previsioni. E' in genere la misura di valutazione

più utilizzata ma non risulta essere la più appropriata agli scopi dell'analisi poichè non tiene conto del problema di rebalancing (il quale causa una previsione maggiore della classe più numerosa), va affiancata quindi ad altre misurazioni;

- **Precision:** la precisione è il rapporto tra il numero delle previsioni corrette di un evento (classe) sul totale delle volte che il modello lo prevede:

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

- **Recall (o sensitivity):** la recall misura la sensibilità del modello. E' il rapporto tra le previsioni corrette per una classe sul totale dei casi in cui si verifica effettivamente:

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

- **F1-score (o F measure):** Detta anche balanced F1-score perchè prende in cosiderazione anche i casi di classificazione errata, è una media armonica tra precision e recall:

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.7}$$

  Un F-score alto indica alti valori di precision e recall, assume valori [0,1].

- **AUC (Area Under the Curve):** Una curva ROC è un grafico che mostra le prestazioni di un modello di classificazione considerandone la

sensibilità (equazione 3.6) e la specificità calcolabile come:

$$specificita' = \frac{FP}{FP + TN} \tag{3.8}$$

La prima misura quante persone sono state affette dalla malattia o quante sono positive (quindi detta anche TPR - True Positive Rate) mentre la seconda misura quante persone non sono affette dalla malattia o quante sono negative (quindi detta anche FPR -False Positive Rate). Poniamo queste metriche in un piano cartesiano:

- specificità nelle ascisse;
- sensibilità nelle ordinate.

La curva ROC viene costruita considerando tutti i possibili valori del test e, per ognuno di questi, si calcola la proporzione di veri positivi e la proporzione di falsi positivi. L'area sottostante alla curva (AUC o Area Under the Curve) è una misura di accuratezza diagnostica. L'area sotto la curva può assumere valori compresi tra 0.5 (assenza di discriminazione) e 1.0 (perfetta discriminazione). Maggiore è l'AUC (la curva si avvicina al vertice sinistro del grafico) maggiore è il potere discriminante nel test. Per l'interpretazione dei valori dell'area sottostante la curva ROC è possibile riferirsi alla classificazione proposta da Swets[8]:

- AUC = 0.5, il test non è informativo;
- $0.5 <$ AUC ≤ 0.7, il test è poco accurato;
- $0.7 <$ AUC ≤ 0.9, il test è moderatamente accurato;

[8]Swets JA. Measuring the accuracy of diagnostic systems. Science 1998; 240: 1285-93.

– AUC = 1, test perfetto.

La figura 3.8 mostra un esempio di curva ROC.

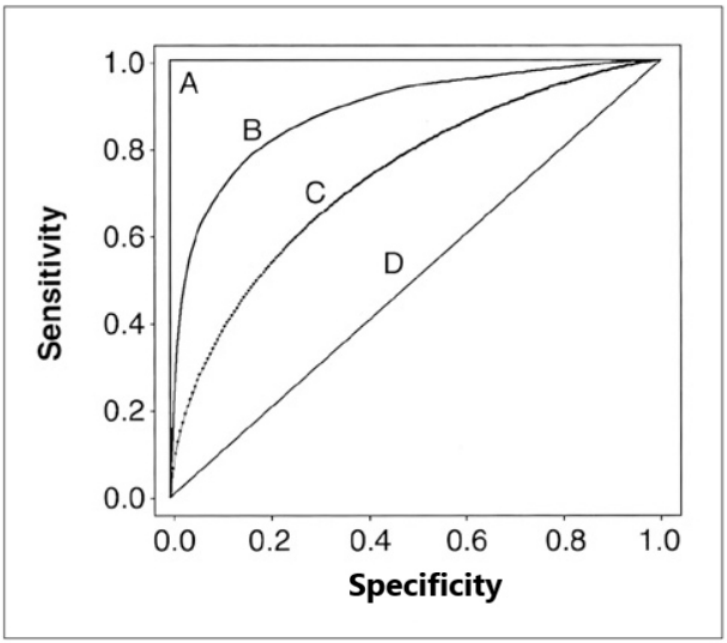

Figura 3.8: Quattro curve ROC con differenti AUC, un test perfetto (A) ha un'area sotto la curva ROC di 1 mentre un test non discriminante (D) ha un AUC di 0.5, i test discriminanti (B e C) si trovano tra questi due estremi, B ha prestazioni diagnostiche migliori rispetto a C.

Regressione Logistica

La regressione lineare:

$$\vec{Y} = \tilde{X}\vec{\beta} + \vec{\epsilon} \tag{3.9}$$

Si basa su cinque ipotesi standard:

- La relazione tra le variabili dipendenti osservate $\hat{y}_i$ e le corrispondenti variabili indipendenti x_i è di tipo lineare, cioè:

$$\hat{y}_i = \beta x_i + \epsilon_i \tag{3.10}$$

Dove ϵ_i è un termine residuo che esprime l'incapacità del modello di riprodurre con esattezza la realtà;

- $E(\epsilon) = 0$, il valore atteso del vettore dei residui è uguale a 0;
- $V(\epsilon) = \sigma^2 I$ dove con I si intende la matrice identità[9], cioè l'omoschedasticità dei residui[10];
- Variabile ϵ distribuita normalmente.
- $\tilde{X}$, matrice contenente le variabili x_i e una colonna di 1, è deterministica e non stocastica;

Nell'equazione 3.9, il vettore Y contiene variabili di risposta y_i comprese tra $(-\infty, +\infty)$, mentre nel modello di regressione logistica il vettore Y contiene y_i tale che assuma valore 1 (successo) con probabilità π e valore 0 (insuccesso) con propabilità $1 - \pi$. L'esito dell'esperimento y_i, è quindi una variabile casuale bernoulliana di parametro π che si verifica con probabilità:

$$Pr(y_i) = \pi_i^y (1 - \pi)^{1-y_i} \tag{3.11}$$

[9] La matrice identità è una matrice avente valore 1 nella diagonale principale e 0 altrove.
[10] In statistica l'omoschedasticità è la proprietà di un insieme di variabili aleatore di avere tutte la stessa varianza finita.

Nel caso in cui l'output sia una variabile dicotomica {0,1}, il modello di regressione lineare fallisce in quanto:

- Considerando la dipendenza di Y rispetto ad una variabile $\vec{x}$ si ha:

$$Y|\vec{x} = \begin{cases} 0 & 1 \\ 1 - \pi(\vec{x}) & \pi(\vec{x}) \end{cases} \tag{3.12}$$

  Il modello di regressione lineare non garantisce il rispetto del campo di variazione [0,1];

- Si viola l'ipotesi di omoschedasticità dei residui:

$$V(y_i) = V(\epsilon_i) = \pi(x_i)(1 - \pi(x_i)) \neq \sigma^2 \tag{3.13}$$

- ϵ non può avere una distribuzione normale ma può assumere solo due valori:

$$Con \qquad y_i = x_i\beta_i + \epsilon_i \tag{3.14}$$

  Se Y = {0,1} allora:

$$0 = x_i\beta_i + \epsilon_i \rightarrow \epsilon_i = -x_i\beta_i \tag{3.15}$$

$$1 = x_i\beta_i + \epsilon_i \rightarrow \epsilon_i = 1 - x_i\beta_i \tag{3.16}$$

Di conseguenza, mentre una regressione lineare non è adeguata ai nostri scopi, una regressione logistica lega una variabile output ad una probabilità che varia tra 0 e 1 che è a sua volta legata ad una variabile input sia discreta

che continua. Sostituendo la variabile dicotomica con l'"odds" (noto anche come "pronostico" o "ragione di scommessa") il quale:

$$odds(\pi) = \omega = \frac{\pi}{1-\pi} \tag{3.17}$$

Otteniamo una misura tale che $\omega(y_i) \subseteq (0, +\infty)$, si utilizza nelle scommesse ed è il rapporto tra la probabilità che si verifichi l'evento favorevole e quella che non si verifichi, di conseguenza possiamo considerare $\pi(x_i)$ come:

$$\pi(x_i) = \frac{\omega_i}{\omega_i + 1} \tag{3.18}$$

Applicando la funzione logaritmica a $\pi(x_i)$ associamo una probabilità [0,1] ad una variabile tra $(-\infty, +\infty)$:

$$logit(\pi) = \ln(\omega) = \ln \frac{\pi}{1-\pi} \quad quindi \quad \pi = \frac{exp(logit(\pi))}{exp(logit(\pi)) + 1} \tag{3.19}$$

Con il termine logit si indica il logaritmo naturale dell'odds, quindi, dall'equazione 3.19 si ricava che la regressione logistica è formulabile come:

$$logit(\pi(\vec{x})) = \vec{x_i}^T \vec{\beta} \tag{3.20}$$

Per legare π_i a x_i, esistono altre funzioni oltre alla logit, ad esempio, la funzione probit che lega le due utilizzando la funzione di ripartizione della normale. Il modello predittivo utilizzato si basa sulla logit.

Regressione logistica simulata tramite ANN

Una regressione logistica è paragonabile ad una rete neurale priva di hidden layer che utilizza una curva sigmoidale come funzione di attivazione e l'entropia come funzione di costo da minimizzare (figura 3.9).

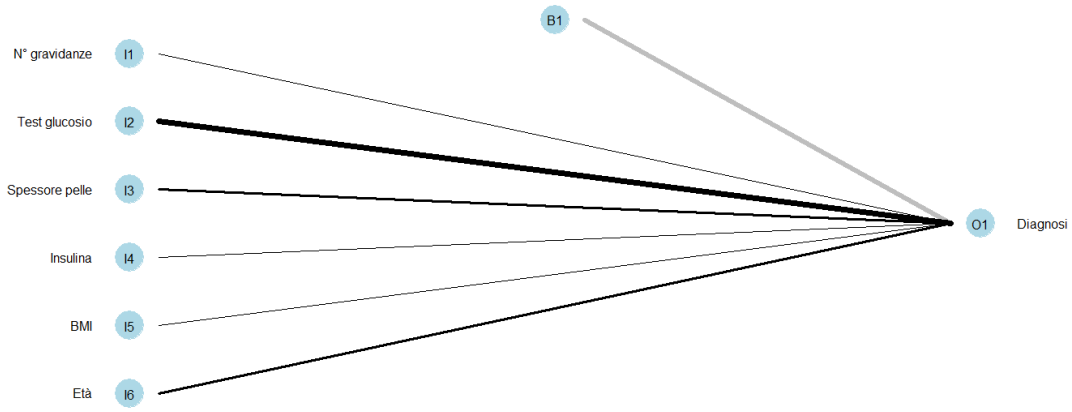

Figura 3.9: Una single layer perceptron, priva di hidden layer simula il comportamento di una regressione logistica, dove B1 indica l'intercetta.

Quest'ultima è una grandezza fisica che misura il quantitativo di "caos", in genere si parla di entropia come "grado di disordine" di un sistema, un aumento del disordine è associato ad un aumento di entropia mentre una diminuzione è associato ad una sua diminuzione.

Facciamo un esempio, nella figura 3.10:

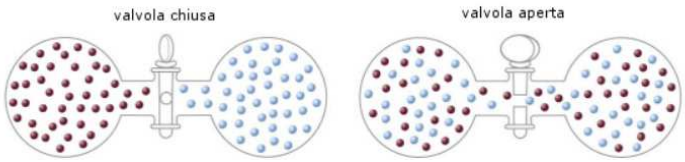

Figura 3.10: Esempio di variazione di entropia all'interno di un sistema fisico.

Due gas contenuti in compartimenti stagni hanno bassi livelli di entropia perchè all'interno dei due sistemi sono contenuti solo molecole del medesimo gas. Aprendo la valvola tra i due, i sistemi smettono di essere isolati, si assiste ad un aumento spontaneo dell'entropia perchè i due si mescolano. In natura avviene uno spontaneo aumento dell'entropia cioè è più probabile trovare sistemi fisici caotici, ciò si ricollega al secondo principio della termodinamica. Utilizzare l'entropia come funzione di costo si traduce come un minimizzare il disordine all'interno della nostra classificazione, i pazienti saranno quindi suddivisi in base alla loro diagnosi.

Multilayer perceptron

Uno dei vantaggi di una rete neurale rispetto ai modelli classici, è la sua flessibilità, cioè, è possibile modificare la sua struttura al fine di ottenere migliori prestazioni. La rete neurale che viene presentata ora, utilizza una curva sigmoidale come funzione di attivazione e l'entropia come funzione di perdita ma, a differenza del modello precedente, possiede uno strato di hidden costituito da due neuroni (figura 3.11).

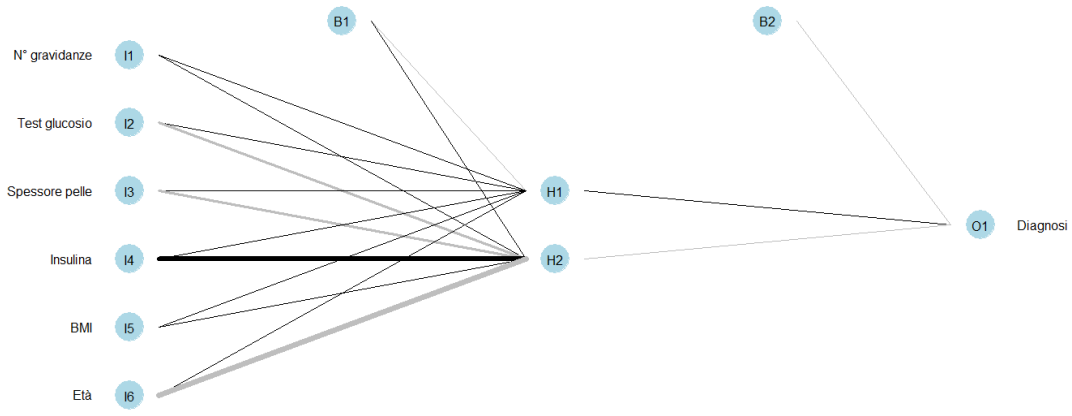

Figura 3.11: Una multilayer perceptron, al precedente modello è stato da uno strato di hidden con lo scopo di migliorarne le prestazioni, con B1 si indica l'intercetta proveniente dall'iinput layer mentre con B2 quello proveniente dallo strato nascosto.

Capitolo 4

RISULTATI

La tabella 4.1 riporta la matrice di confusione ottenuta dal modello di regressione logistica, mentre la figura 4.1 la relativa curva ROC.

	FALSE	TRUE
FALSE	47	9
TRUE	5	18

Tabella 4.1: Matrice di confusione ottenuta dal modello di regressione logisica.

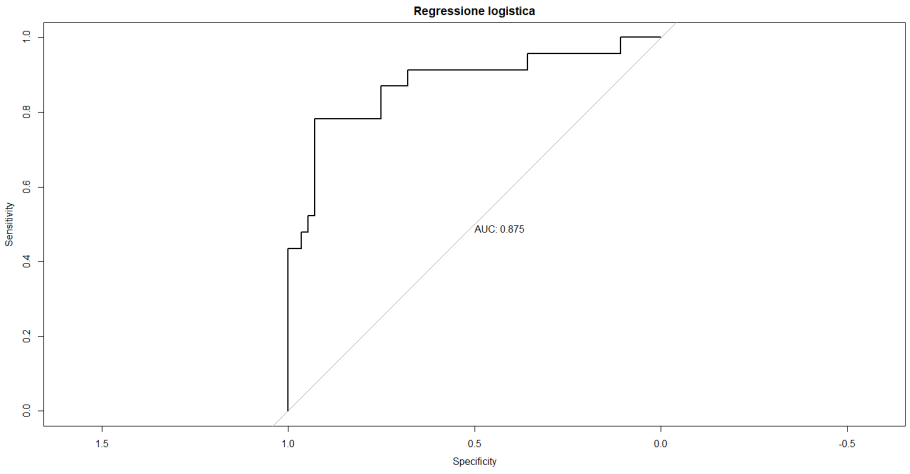

Figura 4.1: Curva ROC del modello di regressione logistica.

La regressione logistica simulata produce la seguente matrice di confusione (tabella 4.2) e curva ROC (figura 4.2).

	FALSE	TRUE
FALSE	47	9
TRUE	5	18

Tabella 4.2: Matrice di confusione ottenuta dalla single layer perceptron.

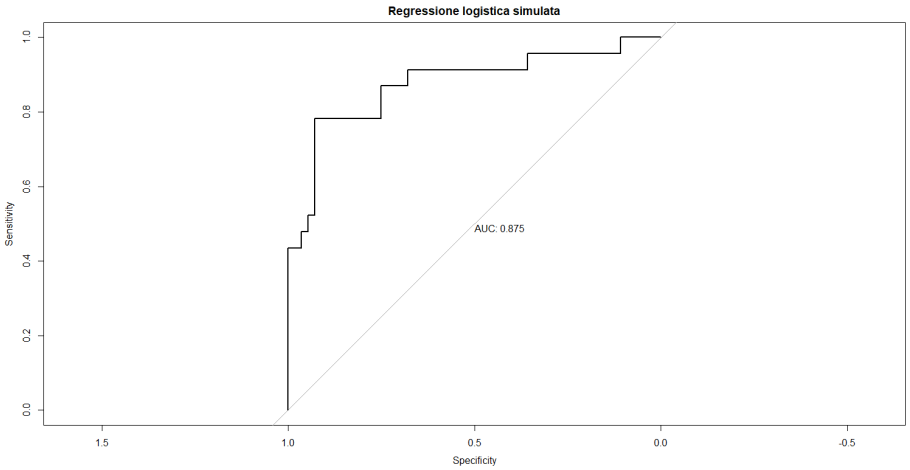

Figura 4.2: Curva ROC del modello di regressione logistica simulata.

I modelli presentano ottime capacità predittive avendo 65 casi di corretta classificazione su 79 e un AUC di 0.875. Si noti come una rete neurale possa simulare perfettamente una regressione logistica. La tabella 4.3 e la figura 4.3 mostrano rispettivamente la matrice di confusione e la curva ROC ottenute dalla MLP.

	FALSE	TRUE
FALSE	49	7
TRUE	3	20

Tabella 4.3: Matrice di confusione ottenuta dalla multilayer perceptron.

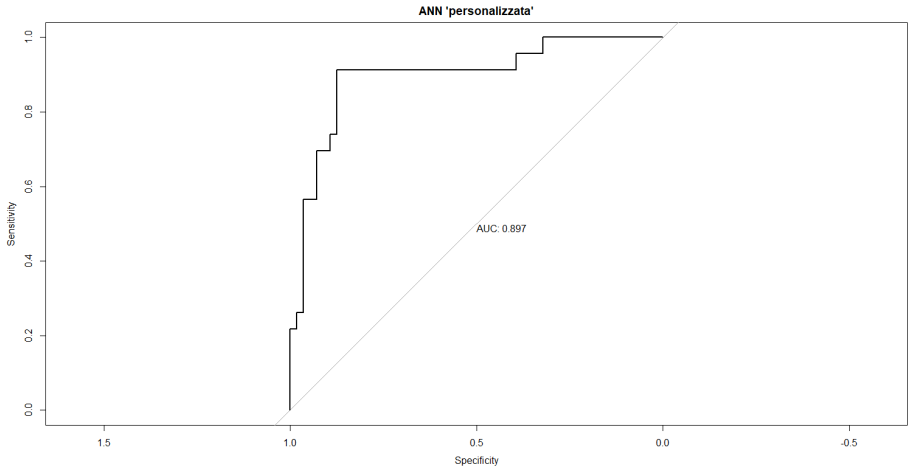

Figura 4.3: Curva ROC della multilayer perceptron.

I modelli predittivi sono comparati sulla base dei valori delle misure di valutazione riportati nella tabella 4.4.

	Regressione logistica	Regressione logistica simulata	MLP
Accuracy	82.278	82.278	87.342
Precision	0.666	0.666	0.741
Recall	0.276	0.276	0.289
F-score	0.391	0.391	0.416
AUC	0.875	0.875	0.898

Tabella 4.4: Metriche di valutazione calcolate nei modelli predittivi.

Da ciò possiamo dire che:

- Tutti i modelli mostrano ottimi risultati, quindi, se non si volesse incorrere nei problemi derivanti dall'utilizzo di una ANN, la regressione logistica si dimostra comunque ottima al fine dell'analisi;
- Una rete neurale è capace di simulare perfettamente una regressione logistica;

- Nonostante la MLP abbia solo due neuroni nascosti in più rispetto alla SLP, si noti un'aumento generale delle performance, tutte e cinque le misure di evaluetion ne hanno beneficiato.

La multilayer perceptron, con un'ottima precisione e recall (errore di prima e di seconda specie rispettivamente di 5 e di 7 pazienti), alti accuracy e f-score, e, con un'AUC prossimo allo 0.9, si dimostra essere il modello predittivo migliore.

Conclusioni

Il concetto di rete neurale, nasce a partire dal 1943 da McCullch e W.Pitts e si evolve fino ad oggi nel deep learning e nei moderni algoritmi i quali sono ora ritenuti i modelli predittivi e di classificazione migliori.
Il lavoro dimostra che una rete neurale può non solo emulare alla perfezione un modello più semplice come una regressione logistica, ma esserne anche superiore. Tuttavia, nonostante l'enorme personalizzazione di questi algoritmi capace di apportarne un aumento nelle performance generali, i problemi derivanti dal loro utilizzo sono molteplici. Il loro addestramento richiede grande potenza di calcolo, hanno tempi di discesa del gradiente a volte molto lunghi e il rischio di overfitting aumenta all'aumentare degli strati.
Di conseguenza, considerando i risultati ottenuti dall'analisi dei pazienti diabetici, per prevedere una diagnosi non è necessario utilizzare un modello così complesso, si possono ottenere degli ottimi risultati con una regressione logistica che ha tempi di addestramento e costi computazionali ridotti.

Bibliografia

[1] Donald O. Hebb (1949), *The organization of behavior; a neuropsychological theory.*

[2] Douglas R. Hofstadter, Alan Turing (2012), *The enigma.*

[3] Kenneth S. Saladin (2012),*Anatomia Umana*, Edizione Italiana a cura di Raffaele De Caro, Terza Edizione.

[4] Kurt Marti (2015), *Stochastic Optimization Methods.*

[5] Ohn C. Hull (2019, 2020), *Machine Learning in Business, an introduction to the word of data science*, seconda edizione.

[6] Pall, J. C. (1992), *Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation.*

[7] Pang-ning tan, Michael steinbach, Anuj karpatne, Vipin kumar (2018) *Introduction to data mining*, seconda edizione.

[8] Paolo Giudici, Silvia Figini (2009) *Applied Data Mining for Business and Industry*, seconda edizione.

[9] Swets JA (1998), *Measuring the accuracy of diagnostic systems.*

Sitografia

http://www.diabete.com.

http://www.nephromeet.com.

https://coralieflorino.com/it.

https://datascience.eu/it.

https://doi.org/10.3348/kjr.2004.5.1.11

https://en.wikipedia.org/wiki/History_of_artificial_neural_networks.

https://machinelearningmastery.com.

https://towardsdatascience.com.

https://virtualmich.com.

https://www.epicentro.iss.it/igea/diabete.

https://www.issalute.it.

https://www.it-impresa.it/blog/cosa-sono-le-reti-neurali.

https://www.mayoclinic.org.

https://www.medelinternational.com.

https://www.ncbi.nlm.nih.gov.

https://www.portalediabete.org.