Comprende versione **ebook**

Giuseppe **Maulucci** • Tommaso **Marchetti** • Cassandra **Serantoni**

Intelligenza Artificiale in Medicina

Dalla teoria alla pratica: esercitazioni guidate e applicazioni cliniche

EdiSES università

Accedi ai contenuti **digitali**

Espandi le tue risorse

un libro che **non pesa**
e si **adatta** alle dimensioni
del **tuo lettore**!

COLLEGATI AL SITO **EDISES.IT**

ACCEDI AL **MATERIALE DIDATTICO**

SEGUI LE **ISTRUZIONI**

Utilizza il codice personale contenuto nel riquadro per registrarti al sito **edises.it** e accedere ai contenuti digitali.

Scopri il tuo **codice personale** grattando delicatamente la superficie

Il volume NON può essere venduto, né restituito, se il codice personale risulta visibile.
L'**accesso ai contenuti digitali** sarà consentito **per 18 mesi**.

Per attivare i **servizi riservati**, collegati al sito **edises.it** e segui queste semplici istruzioni

Se sei registrato al sito

- clicca su *Accedi al materiale didattico*
- inserisci email e password
- inserisci le ultime 4 cifre del codice ISBN, riportato in basso a destra sul retro di copertina
- inserisci il tuo **codice personale** per essere reindirizzato automaticamente all'area riservata

Se non sei già registrato al sito

- clicca su *Accedi al materiale didattico*
- registrati al sito **edises.it**
- attendi l'email di conferma per perfezionare la registrazione
- torna sul sito **edises.it** e segui la procedura già descritta per *utenti registrati*

EdiSES università

I contenuti digitali sono accessibili dalla propria **area riservata** secondo la procedura indicata nel frontespizio.

Dalla sezione **materiali e servizi** della tua area riservata potrai accedere all'**Ebook**, ovvero la versione digitale del testo in formato epub, standard dinamico che organizza il flusso di testo in base al dispositivo sul quale viene visualizzato. Fruibile mediante l'applicazione gratuita BookShelf, consente una visualizzazione ottimale su lettori e-reader, tablet, smartphone, iphone, desktop, Android, Apple e Kindle Fire.

L'accesso ai contenuti digitali sarà consentito per **18 mesi**.

EdiSES università

G. Maulucci • T. Marchetti • C. Serantoni

Intelligenza Artificiale in Medicina

Dalla teoria alla pratica: esercitazioni guidate e applicazioni cliniche

EdiSES università

Giuseppe Maulucci, Tommaso Marchetti, Cassandra Serantoni
Intelligenza Artificiale in Medicina. Dalla teoria alla pratica: esercitazioni guidate e applicazioni cliniche

9 8 7 6 5 4 3 2 1 0
2029 2028 2027 2026 2025

Le cifre sulla destra indicano il numero e l'anno dell'ultima ristampa effettuata

Stampato presso
Vulcanica S.r.l. – Nola (NA)

per conto della
EdiSES Edizioni S.r.l. – Piazza Dante Alighieri, 89 – Napoli

www.edises.it
assistenza.edises.it

ISBN 978 88 3623 245 1

Prefazione

Viviamo in un'epoca in cui l'Intelligenza Artificiale non è più solo materia di ricerca specialistica, ma una realtà che trasforma la vita quotidiana e la pratica clinica. Negli ultimi anni, i progressi – dal *Deep Learning* ai Large Language Models – hanno rivoluzionato il modo in cui interpretiamo immagini, testi e dati clinici. Questa rivoluzione non riguarda solo ingegneri e informatici: coinvolge direttamente i medici, gli operatori sanitari e tutti coloro che si occupano di salute.

La medicina è da sempre una disciplina fondata sull'analisi e sull'interpretazione di dati complessi: immagini radiologiche, parametri di laboratorio, storie cliniche e segnali biologici. L'Intelligenza Artificiale si propone come un alleato potente, capace di supportare il professionista nella diagnosi, nella prognosi e nella personalizzazione delle cure, con un livello di *precisione* e velocità che supera le capacità umane. Per questo, diventa indispensabile per il medico moderno acquisire una comprensione dei principi generali dell'Intelligenza Artificiale: non per sostituire il data scientist, ma per dialogare con queste tecnologie e valutarne criticamente il contributo.

Il valore formativo di questo testo risiede proprio qui: fornire una base culturale solida che consenta allo studente di riconoscere i concetti chiave, orientarsi tra modelli e approcci diversi, e condividere un linguaggio comune con specialisti di altri ambiti.

L'obiettivo non è offrire soluzioni preconfezionate, ma stimolare un nuovo modo di pensare: sviluppare uno spirito critico capace di cogliere potenzialità e limiti dell'Intelligenza Artificiale. La formazione del medico del futuro non potrà prescindere da questa competenza trasversale, che rappresenta ormai una componente essenziale della ricerca biomedica e della pratica clinica.

Con queste pagine, lo studente potrà apprendere i fondamenti teorici dell'Intelligenza Artificiale e toccarne con mano le applicazioni più rilevanti in medicina, ponendo così le basi per una nuova alleanza tra scienza, tecnologia e cura della persona.

Questo volume è organizzato in modo da accompagnare progressivamente lo studente alla scoperta dell'Intelligenza Artificiale applicata alla medicina, alternando momenti teorici a esercitazioni pratiche guidate.

Nel **Capitolo 1** introduciamo i concetti di base dell'Intelligenza Artificiale: dalla definizione generale, alla sua storia, fino agli approcci simbolici e connessionisti, con una panoramica delle principali applicazioni in ambito medico.

Il **Capitolo 2** affronta i fondamenti dell'apprendimento automatico (machine learning), spiegando concetti chiave come *overfitting*, *underfitting* e generalizzazione. Vengono presentati i principali algoritmi di regressione, classificazione e clustering, insieme alle tecniche di riduzione dimensionale, con riferimenti a casi d'uso clinici.

Nel **Capitolo 3** ci concentriamo sulle reti neurali artificiali, dalle architetture *feedforward* alle reti ricorrenti, fino alle applicazioni cliniche più rilevanti.

Il **Capitolo 4** è dedicato all'imaging medico, con una panoramica sulle tecniche di analisi automatica delle immagini, dalla radiomica alle reti neurali convoluzionali, fino ai più recenti modelli basati su Transformer.

Il **Capitolo 5** introduce l'elaborazione del linguaggio naturale, mostrando come l'Intelligenza Artificiale possa essere applicata a testi clinici, referti o cartelle elettroniche, fino ad arrivare ai Large Language Models e ai moderni sistemi conversazionali.

Il **Capitolo 6** discute gli aspetti etici, regolatori e sociali dell'Intelligenza Artificiale in medicina, sottolineando questioni cruciali come privacy, equità, *bias*, trasparenza e integrazione nella pratica clinica.

Un elemento distintivo del libro è l'approccio pratico. Grazie al software open source ***Orange Data Mining***, lo studente potrà sperimentare direttamente le tecniche di Intelligenza Artificiale senza conoscenze di programmazione. Attraverso un'interfaccia visuale, basata su moduli e flussi intuitivi, sarà possibile esplorare *dataset* clinici, applicare algoritmi di *machine learning* e osservare i risultati in tempo reale. Questo rende l'Intelligenza Artificiale accessibile anche a chi non ha un background informatico, ma desidera comprenderla e usarla in contesto medico.

In questo modo, il lettore può costruire passo dopo passo non solo una solida comprensione teorica, ma anche una concreta familiarità con gli strumenti dell'Intelligenza Artificiale applicata alla medicina.

Roma, ottobre 2025

Gli Autori

Indice

CAPITOLO 3
Le Reti Neurali Artificiali

3.1 Introduzione alle Reti Neurali Artificiali

Le **reti neurali artificiali** (*Artificial Neural Networks*, *ANN*) sono modelli computazionali di *machine learning* ispirati IA meccanismi del cervello biologico. In altre parole, cercano di replicare in forma semplificata il comportamento di una rete di neuroni umani: ciascun *neurone artificiale* svolge un piccolo compito (ad esempio, sommare segnali in ingresso e produrre un segnale in uscita), e l'intelligenza emerge dalla cooperazione di molti neuroni. Ne abbiamo visto un esempio nel paragrafo 1.3 quando abbiamo introdotto il paradigma connessionista. Questa analogia con il cervello ha motivato sin dagli inizi lo sviluppo delle ANN, nella speranza di conferire IA calcolatori capacità di apprendimento e di risoluzione di problemi complessi in modo simile a un cervello umano. In generale, una rete neurale artificiale è composta da un gran numero di semplici unità di calcolo (i neuroni artificiali) interconnesse tra loro. Ciascun neurone elabora dei numeri in ingresso e produce un numero in uscita; organizzando molti neuroni in architettura gerarchica, la rete può approssimare funzioni anche molto complesse. A differenza dei programmi tradizionali, in cui il programmatore deve specificare esplicitamente ogni regola, una rete neurale **impara** le relazioni dai dati di addestramento. Questo le consente di risolvere problemi difficili da codificare con regole fisse (come riconoscere *pattern* in immagini o segnali) e di **generalizzare** a dati nuovi dopo l'addestramento. In medicina, tale capacità di apprendere dai dati è preziosa perché molti segnali biologici o immagini diagnostiche sono troppo complessi per essere analizzati con soli criteri deterministici: le reti neurali offrono un approccio *data-driven* (*guidato dai dati*) per affrontare queste sfide. Questo capitolo fornisce un'introduzione di base alle reti neurali artificiali: i concetti più avanzati, in particolare relativi alle CNN e alle loro applicazioni in imaging medico, saranno approfonditi nel capitolo successivo.

3.2 Struttura e funzionamento delle reti neurali

Dal punto di vista strutturale, una rete neurale artificiale è composta da **neuroni artificiali** organizzati a strati e connessi tramite **pesi sinaptici**. Vediamo i concetti chiave:

Neurone artificiale: è l'unità di base della rete (Figura 41). Ogni neurone riceve uno o più valori in ingresso (provenienti dai dati iniziali o dagli altri neuroni dello strato precedente), calcola una combinazione di questi ingressi e genera un segnale in uscita. In pratica, ciascun neurone esegue una somma pesata degli ingressi più un termine costante detto ***bias*** (che funge da soglia) e poi applica una **funzione di attivazione** al risultato. Ad esempio, se $x_1, x_2, \ldots\ldots, x_n$ sono gli input ricevuti e $w_1, w_2, \ldots\ldots, w_n$ i rispettivi pesi sinaptici, il neurone calcola

$$z = w_1 x_1 + w_2 x_2 + \cdots .. + w_n x_n + b \tag{3.1}$$

(dove *b* è il *bias*) e quindi produce in uscita

$$y = f\,(z) = f\,(w_1 x_1 + w_2 x_2 + \ldots\ldots + w_n x_n + b) \tag{3.2}$$

dove *f* è la funzione di attivazione scelta (vedi sotto)

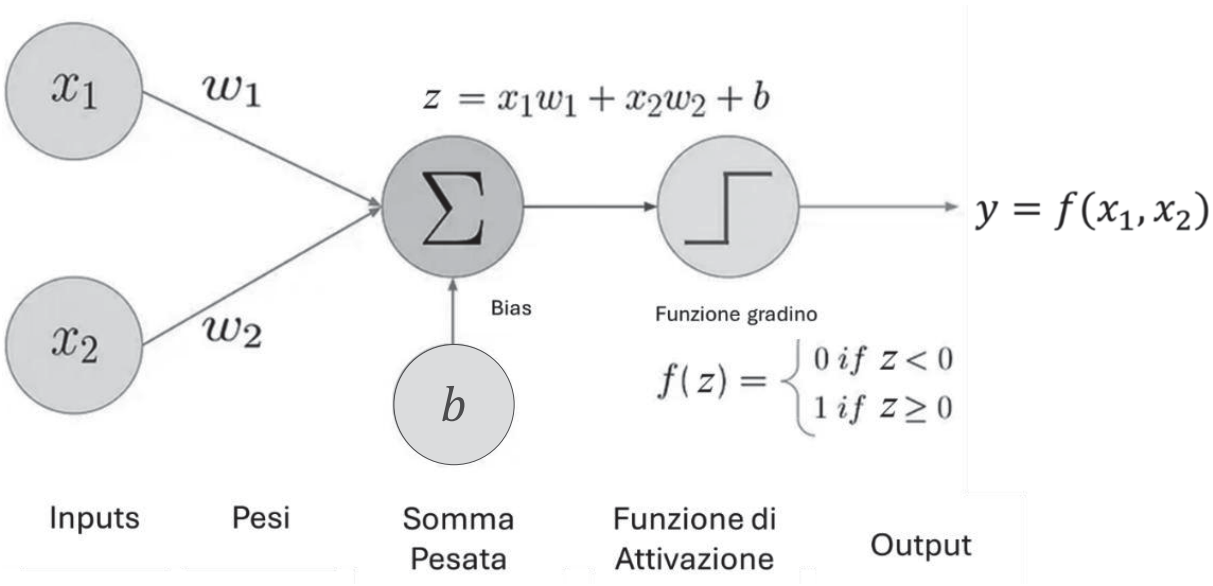

Figura 41. Architettura di un neurone artificiale semplice: gli input x_1 e x_2 vengono moltiplicati per i rispettivi pesi w_1 e w_2, sommati insieme al *bias b* per ottenere una somma pesata z. Il valore ***z*** viene quindi passato a una funzione di attivazione gradino, che restituisce un output binario: 0 se z<0, 1 se z≥0.

Pesi e *bias*: i *pesi* w_i determinano l'importanza di ciascun segnale in ingresso per il neurone, mentre il *bias b* determina la predisposizione del neurone ad attivarsi (in analogia, corrisponde al potenziale di soglia nei neuroni biologici). In fase di apprendimento, la rete aggiusta i pesi e i *bias* per migliorare le proprie prestazioni: essi sono i parametri interni che la rete "impara" dai dati.

Funzione di attivazione: è la funzione (in genere non lineare) applicata alla somma pesata 3.1. Introduce la necessaria **non-linearità** nel sistema, permettendo alla rete

di modellare relazioni complesse (senza una funzione non lineare, una pila di neuroni sarebbe matematicamente equivalente a un singolo strato). Storicamente, la prima funzione di attivazione usata è stata una funzione a **soglia** (*step function*) che attiva il neurone (output 1) solo se z supera una certa soglia, altrimenti il neurone resta inattivo (output 0) – questo era il caso del perceptron. In reti più moderne si usano funzioni continue come la **sigmoide** (che produce un output tra 0 e 1, adatta a probabilità), la **tangente iperbolica** o la popolare **ReLU** (*Rectified Linear Unit*), che restituisce 0 per valori negativi di z e z stesso per valori positivi. Queste funzioni rendono il comportamento della rete differenziabile e facilitano l'apprendimento.

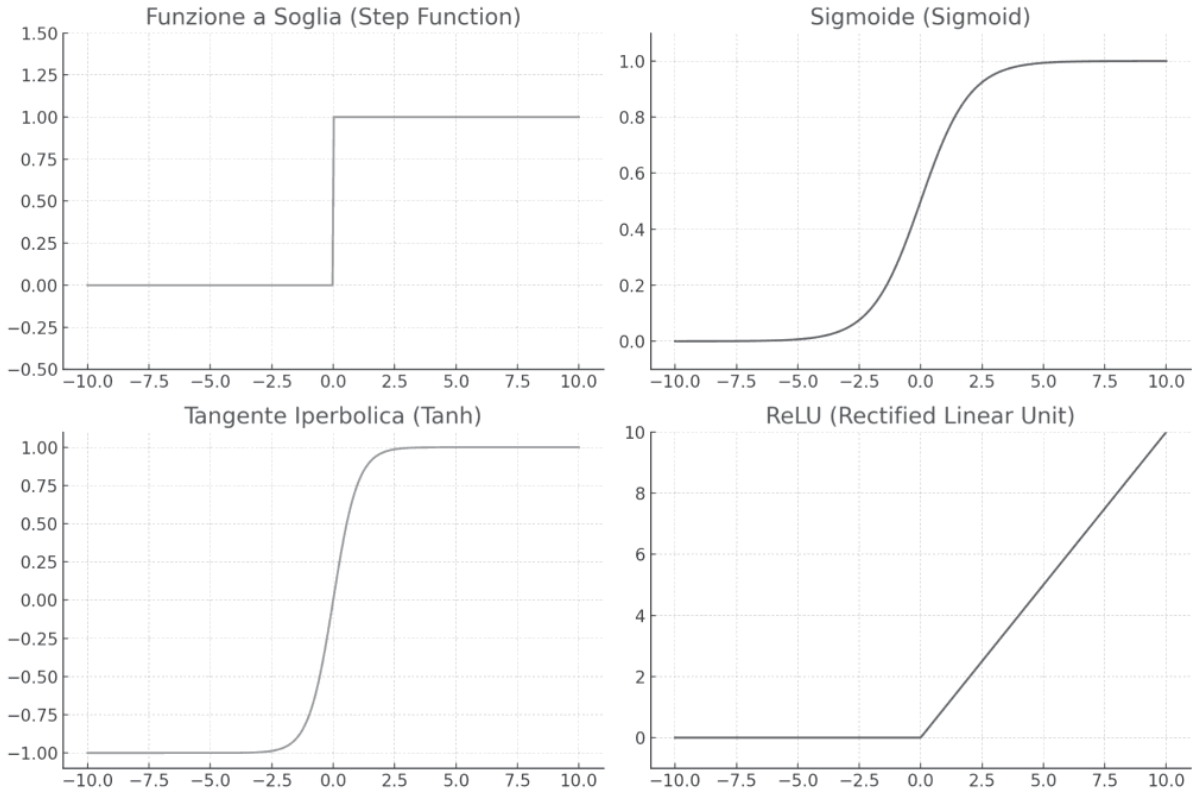

Figura 42. Principali funzioni di attivazione utilizzate nelle reti neurali artificiali. La figura mostra un confronto tra le principali funzioni di attivazione utilizzate nelle reti neurali. La Funzione a Soglia (Step Function) restituisce 0 per input negativi e 1 per input non negativi, ma non è derivabile e quindi poco adatta al training con *backpropagation*. La Sigmoide (Sigmoid) produce output compresi tra 0 e 1, ed è utile per problemi di classificazione binaria, anche se può soffrire del problema del vanishing gradient. La Tangente Iperbolica (Tanh) ha una forma simile alla sigmoide ma restituisce valori tra -1 e 1, ed essendo centrata sullo zero, migliora spesso la convergenza durante l'addestramento. Infine, la ReLU (Rectified Linear Unit) restituisce zero per input negativi e un valore lineare per input positivi; è semplice, computazionalmente efficiente e oggi è una delle funzioni più utilizzate nelle reti profonde.

Architettura a strati: i neuroni sono organizzati in livelli sequenziali. Tipicamente distinguiamo **strato di ingresso** (*input layer*), che riceve i dati iniziali (ad es. i pixel di un'immagine, o i valori di segnali/parametri clinici); uno o più ***layer* nascosti (***hidden layers***)** composti da neuroni interni che elaborano progressivamente l'informazione; e uno **strato di uscita** (*output layer*) i cui neuroni producono il risultato finale della rete (ad es. le classi previste, oppure un valore numerico di output) (Figura 43). Ogni connessione tra neuroni ha un peso associato, e l'architettura completa definisce quali neuroni sono connessi (di solito tutti contro tutti tra strati adiacenti, nel caso di reti *completamente connesse*). Possiamo dunque immaginare la rete neurale come una complessa funzione matematica con molti parametri (i pesi e *bias*) suddivisa in blocchi (*neuroni* e *strati*). Inizialmente la rete si comporta in modo quasi casuale, ma aggiustando opportunamente i pesi in base a esempi noti (addestramento), riesce gradualmente a "modellare" i dati e a fornire output utili.

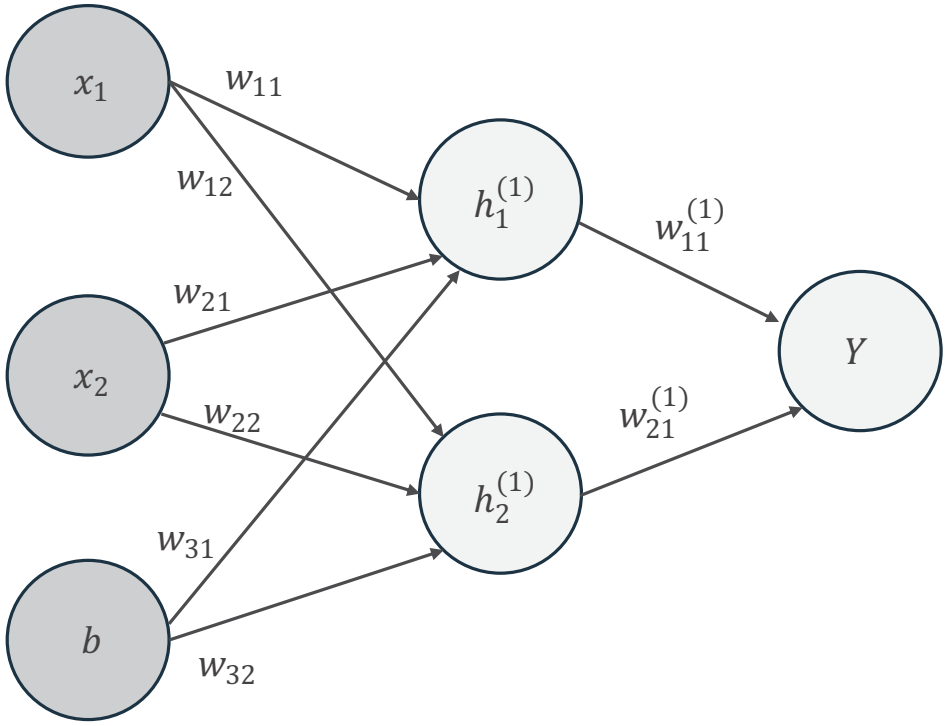

Figura 43. Schema di una rete neurale artificiale composta da tre livelli. Il livello di input contiene due nodi di input x_1 e x_2, che rappresentano le caratteristiche del dato in ingresso, e un nodo di *bias*. Il livello nascosto include due neuroni $h_1^{(1)}$ e $h_2^{(1)}$, che ricevono input combinati dai nodi precedenti attraverso i pesi w_{11},w_{12},w_{21},w_{22},w_{31},w_{32}. Il livello di output è formato da un singolo neurone Y collegato ai neuroni del livello nascosto tramite i pesi $w_{11}^{(1)}$ e $w_{21}^{(1)}$. Ogni connessione è associata a un peso sinaptico w_{ij} , dove i è il nodo di partenza e j quello di arrivo. Il nodo di *bias* fornisce un contributo costante che migliora la capacità di apprendimento del modello.

3.3 Reti neurali *feedforward*

Una rete neurale ***feedforward*** è il tipo base di rete neurale artificiale, in cui le connessioni tra neuroni non formano cicli. L'architettura *feedforward* prevede che gli input attraversino gli strati nascosti fino all'output seguendo una sola direzione (dall'ingresso verso l'uscita). Ciò significa che ogni neurone influenza solo quelli degli strati successivi (mai sé stesso o neuroni precedenti). Questa struttura, priva di retroazioni, rende il comportamento della rete più semplice da analizzare ed è adatta a compiti in cui l'output dipende esclusivamente dall'input corrente (e non da stati precedenti, come invece vedremo nelle reti ricorrenti).

3.3.1 Architettura e forward propagation

In una rete *feedforward* classica (detta anche **Multi-*Layer* Perceptron** se è a più strati), tutti i neuroni di un dato strato sono connessi a *tutti* i neuroni del *layer* successivo (architettura **fully connected**). Ad esempio, consideriamo una rete con un *layer* di input con 3 neuroni (che ricevono 3 caratteristiche in ingresso, poniamo: età, pressione sanguigna e glicemia di un paziente), un *layer* nascosto con 4 neuroni, e un *layer* di output con 1 neurone (che produca, ad esempio, la probabilità che il paziente abbia una certa malattia). La ***forward propagation*** (propagazione in avanti) consiste nel calcolare l'uscita della rete per un certo input, strato dopo strato: I neuroni di input prendono i rispettivi valori iniziali (nell'esempio, 3 valori numerici dai dati del paziente) e li inoltrano come segnale al *layer* successivo. Ogni neurone del *layer* nascosto calcola la somma pesata dei 3 segnali di input che riceve, aggiunge il *bias* e applica la sua funzione di attivazione, producendo un valore di attivazione. Si ottengono così 4 valori (uscite dei neuroni nascosti). Il neurone di output riceve questi 4 valori in ingresso, li combina con i propri pesi e *bias*, applica la funzione di attivazione (ad esempio una sigmoide per ottenere una probabilità tra 0 e 1) e calcola l'output finale della rete, ad esempio 0.85 che potremmo interpretare come "85% di probabilità che il paziente abbia la malattia". Questo flusso è chiamato *forward pass*: in generale, dati gli input e fissati i pesi, è immediato ottenere l'output della rete eseguendo questi passaggi in cascata.

3.3.2 Training via backpropagation

La vera potenza delle reti neurali sta nel modo in cui imparano dai dati. L'addestramento di una rete *feedforward* avviene tipicamente in modo supervisionato: si forniscono alla rete molti esempi di input con il rispettivo output atteso (etichetta o valore target), e la rete aggiusta i propri parametri per approssimare quella relazione ingresso–uscita. Il processo è reso possibile da un

algoritmo chiamato ***backpropagation*** (retropropagazione del gradiente), combinato con metodi di ottimizzazione come la discesa del gradiente (Figura 44).

In breve, il funzionamento è questo: un input viene passato attraverso la rete (*forward propagation*) producendo un output predetto; questo viene confrontato con il target corretto e dalla differenza si calcola un errore (*loss*), corrispondente alla funzione obbiettivo introdotta nel capitolo 2. L'errore viene propagato all'indietro per calcolare quanto ciascun peso interno ha contribuito, e i parametri (pesi e *bias*) vengono aggiornati muovendosi in direzione opposta al gradiente, in proporzione al learning rate. Ripetendo il ciclo su migliaia di esempi e per molte epoche di training, la rete riduce progressivamente l'errore complessivo fino a un livello accettabile.

In pratica, i parametri vengono modificati per minimizzare la funzione di perdita attraverso un algoritmo molto efficiente e rapido dal punto di vista computazionale, anche se i passaggi matematici restano complessi e non sempre intuitivi. È come uno studente che, affrontando esercizi diversi, corregge gradualmente i propri errori e impara a generalizzare i concetti.

Al termine dell'addestramento, la rete è in grado di effettuare **inferenza**, cioè applicare le regole apprese a nuovi casi, ad esempio predire la diagnosi di un paziente mai visto. Naturalmente, l'addestramento di reti profonde richiede grandi quantità di dati e notevole capacità computazionale; inoltre è fondamentale evitare fenomeni come l'*overfitting*, adottando tecniche di regolarizzazione e validazione incrociata.

Un altro problema tipico nell'addestramento delle reti profonde è quello del *vanishing* e *exploding gradient*, che rende difficile apprendere dipendenze a lungo termine o in reti molto profonde. Per mitigarlo si adottano funzioni di attivazione come ReLU e tecniche come la *batch normalization*, che stabilizza la distribuzione degli input negli strati, e il *dropout*, che disattiva casualmente neuroni durante l'addestramento per ridurre l'*overfitting*.

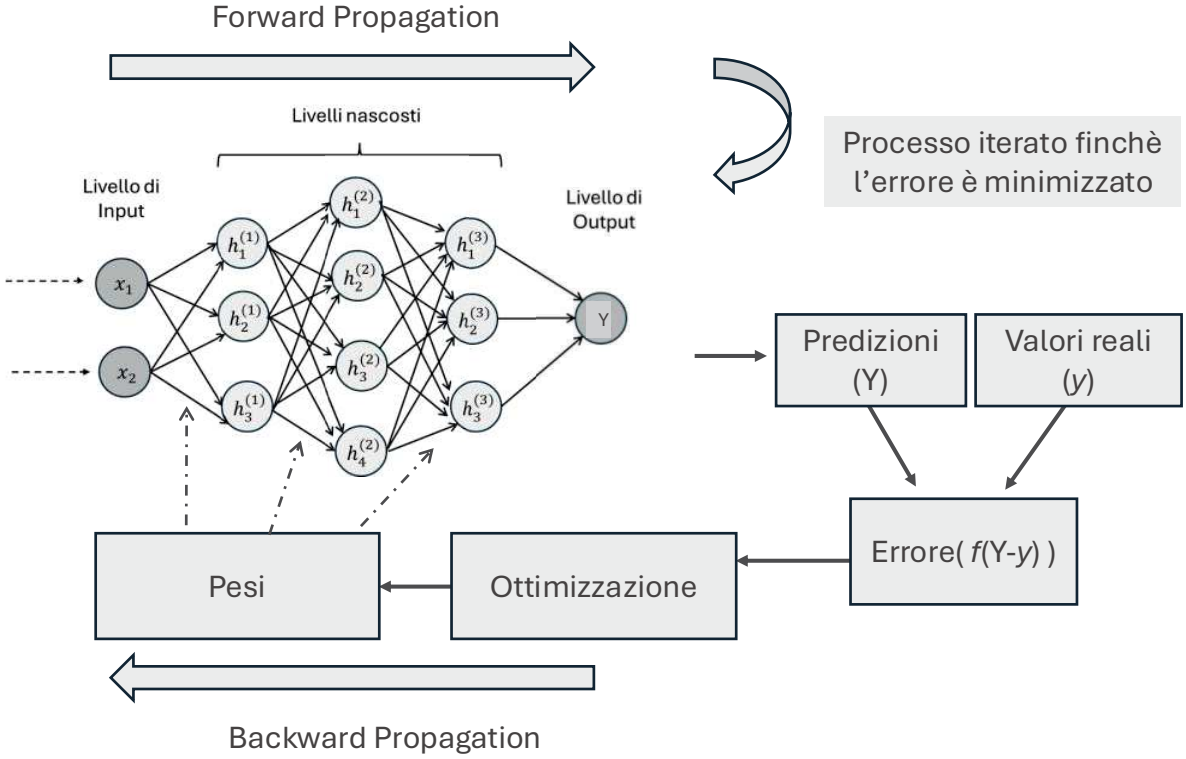

Figura 44. Schema del processo di apprendimento in una rete neurale tramite *Forward propagation* e Backward Propagation. Nella fase di forward, gli input x_1 e x_2 attraversano la rete e generano una predizione Y. Quest'ultima viene confrontata con i valori veri y per calcolare l'errore attraverso una funzione di perdita (*Loss* Function). Nella fase di backward, l'errore viene retropropagato attraverso la rete per aggiornare i pesi mediante un ottimizzatore, riducendo progressivamente l'errore. Il processo viene iterato fino a raggiungere una minima differenza tra predizione e realtà.

3.3.3 Esempio pratico in ambito medico (rete feedforward)

Per fissare le idee, consideriamo un esempio semplice di utilizzo di una rete neurale *feedforward* in campo medico. Supponiamo di voler realizzare un sistema che aiuti a diagnosticare il **diabete** a partire da alcuni parametri clinici di base di un paziente (esempio puramente illustrativo). Potremmo progettare una rete neurale con:

Input: età del paziente, indice di massa corporea (BMI), livello di glicemia a digiuno, pressione arteriosa, e magari altre misure di laboratorio (ognuno di questi è un neurone di input che codifica un valore numerico).

Strato nascosto: ad esempio 5-10 neuroni che combinano non linearmente questi input per estrarre *pattern* nascosti nei dati (relazioni tra età, glicemia, ecc. che possano indicare rischio di diabete).

Output: un neurone che emette la probabilità (0 a 1) che il paziente abbia il diabete. Addestrando questa rete su centinaia di casi noti (pazienti con i loro parametri e la

conferma se diabetici o meno), la rete imparerebbe ad associare i *pattern* di parametri con l'esito diagnostico. Dopo l'addestramento, fornendo i parametri di un nuovo paziente, la rete restituirebbe ad esempio un valore vicino a 1 se i valori sono compatibili con un paziente diabetico, vicino a 0 se compatibili con un non diabetico, con valori intermedi per indicare incertezza. Un tale modello, se accurato, potrebbe supportare il medico nel decidere se raccomandare ulteriori test (es. curva da carico di glucosio) per confermare la diagnosi.

Questo esempio illustra come una rete *feedforward* può svolgere un compito di **classificazione binaria** in medicina. Ovviamente, in contesti clinici reali le reti neurali vengono impiegate su scale molto più ampie e con architetture più complesse (spesso reti profonde, con decine di neuroni e più strati), ma il principio di base rimane lo stesso. Storicamente, già negli anni '90 reti neurali simili furono sperimentate per diagnosi mediche, ad esempio per identificare **infarti** basandosi su parametri di esami del sangue e segni vitali: si notò che queste reti potevano cogliere combinazioni di indicatori difficilmente traducibili in regole statiche, migliorando la sensibilità diagnostica rispetto ad algoritmi tradizionali.

3.4 Reti neurali ricorrenti (RNN)

Le **reti neurali ricorrenti** (Recurrent Neural Networks, RNN) sono una famiglia di reti neurali progettate per elaborare dati sequenziali, cioè informazioni che si sviluppano nel tempo (o in una sequenza ordinata). La caratteristica distintiva di una RNN è la presenza di **connessioni ricorrenti**, ovvero di loop all'interno della rete: l'output (o lo stato interno) di un neurone in un dato istante può influenzare nuovamente l'attivazione di quel neurone (o dei neuroni a monte) in istanti successivi. In altre parole, le RNN hanno una sorta di **memoria interna** che consente di tenere traccia di ciò che è avvenuto nelle step precedenti della sequenza.

Una RNN semplice può essere immaginata come una rete *feedforward* "estesa" nel tempo. Pensiamo a un neurone ricorrente che ad ogni passo temporale riceve un input esterno (es. il valore di un segnale al tempo *t*) e un input addizionale proveniente dal proprio stato al tempo precedente (*t-1*). Così, l'output al tempo *t* dipenderà sia dal nuovo input sia da un *riassunto* dello stato passato, permettendo di accumulare informazioni. Spesso si rappresenta una RNN *srotolandola* (*unrolling*) lungo la sequenza temporale: appare così come copie multiple della stessa rete in cascata, dove ciascuna copia passa il proprio stato interno alla successiva (Figura 45). Questo *stato ricorrente* funge da memoria. Grazie a questa architettura ad anello, le RNN possono, in teoria, mantenere informazioni su sequenze anche lunghe e modellare dipendenze temporali tra i dati. Ad esempio, una RNN che

analizza un segnale ECG potrà "ricordare" il *pattern* dei battiti cardiaci precedenti mentre elabora il battito corrente, oppure una RNN che analizza testo può tenere conto delle parole precedenti per contestualizzare la parola attuale.

In pratica le RNN semplici soffrono di problemi nel memorizzare dipendenze a lungo termine (fenomeni di *vanishing/exploding gradient* durante l'addestramento li rende difficili da addestrare su sequenze lunghe). Per ovviare a ciò, sono state sviluppate varianti più sofisticate come le **LSTM** (*Long Short-Term Memory*) e le **GRU** (*Gated Recurrent Unit*), che introducono meccanismi interni (*gate*) per conservare o dimenticare informazioni in modo controllato, riuscendo a catturare dipendenze anche lontane nel tempo (Hochreiter & Schmidhuber, 1997). Queste architetture avanzate sono molto utilizzate per sequenze complesse. Le LSTM introducono meccanismi di controllo (gate di *input*, *output* e *forget*) che permettono di mantenere o eliminare selettivamente informazioni, mentre le GRU semplificano questa struttura mantenendo alte prestazioni. Entrambe sono oggi impiegate con successo in medicina, ad esempio per l'analisi di segnali ECG o EEG, dove è cruciale riconoscere *pattern* che si estendono nel tempo.

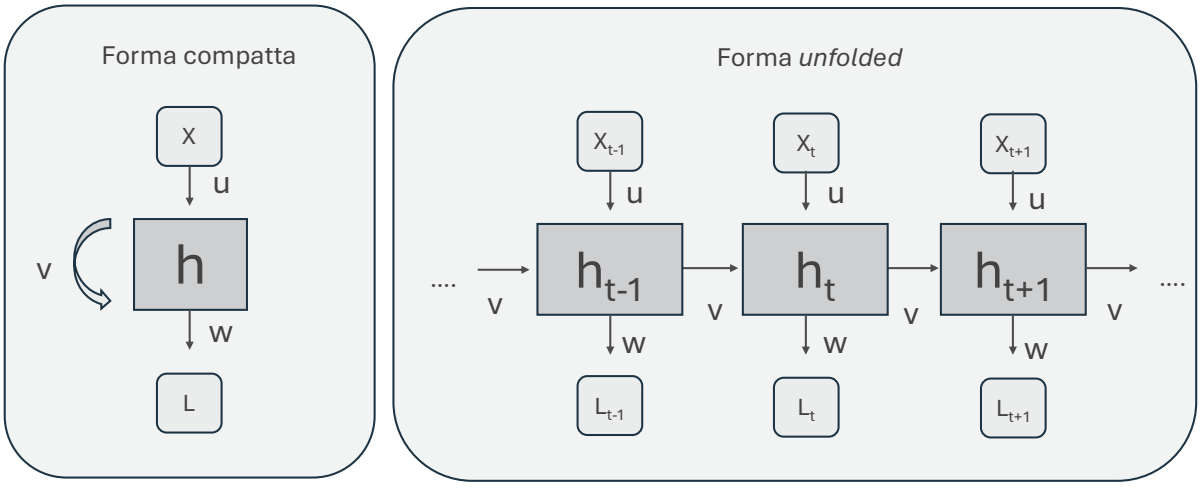

Figura 45. Schema esplicativo del meccanismo di *unfolding* di una Rete Neurale Ricorrente (RNN). A sinistra, la rete è rappresentata nella sua forma compatta con un singolo nodo ricorrente h che riceve input x, produce output L e reinserisce il proprio stato tramite il collegamento ricorrente. A destra, la rete viene "aperta nel tempo" mostrando come l'informazione venga propagata attraverso diverse fasi temporali: ogni stato nascosto h_t è influenzato dallo stato precedente h_{t-1} e dall'input corrente x_t, e genera un output L_t. I pesi u, v e w sono condivisi in ogni istante temporale.

Approfondimento (per chi vuole capire come funziona tecnicamente):

La parte restante del paragrafo è dedicata a un approfondimento teorico sul funzionamento delle RNN. Tale sezione non è indispensabile per comprendere i

concetti di base del modello: chi lo desidera può quindi passare direttamente al paragrafo successivo senza pregiudicare la comprensione complessiva. La Figura 45 mostra due rappresentazioni della stessa RNN: una forma compatta, che ne sintetizza il comportamento, e una forma espansa nel tempo (*unfolded*) che ne esplicita la dinamica passo dopo passo.

Rappresentazione compatta

Nel diagramma a sinistra, la rete è rappresentata in modo astratto tramite un singolo nodo ricorrente, indicato con **h**, che corrisponde allo stato nascosto della rete. A ogni istante temporale, il nodo riceve due informazioni: l'input corrente **X** e il proprio stato precedente, rappresentato dalla freccia ricorrente che rientra su **h**. A partire da queste informazioni, la rete genera un output **L**. Il comportamento del modello è governato da tre matrici di pesi fondamentali:

- **U** collega l'input **X** allo stato nascosto **h**, traducendo il dato grezzo nello spazio interno della rete.
- **W** collega lo stato precedente $\mathbf{h_{t-1}}$ al nuovo stato $\mathbf{h_t}$, consentendo alla rete di mantenere e aggiornare la memoria del contesto passato.
- **V** collega lo stato nascosto $\mathbf{h_t}$ all'output $\mathbf{L_t}$, trasformando la rappresentazione interna in un segnale osservabile.

Questa forma compatta racchiude tutto il comportamento dinamico della rete in un unico nodo, ma non mostra esplicitamente l'evoluzione temporale.

Rappresentazione *unfolded* nel tempo

Per comprendere meglio come opera una RNN, è utile osservare la rappresentazione *unfolded* (a destra). Qui la rete è esplicitata su più passi temporali successivi. Ogni nodo (h_{t-1}, h_t, h_{t+1}, ...) rappresenta lo stato della rete a un dato istante, mentre gli input istante (X_{t-1}, X_t, X_{t+1}, ...) e gli output (L_{t-1}, L_t, L_{t+1}, ...) variano nel tempo. Il collegamento tra gli stati nascosti rappresenta il trasferimento di memoria lungo la sequenza: ciò consente alla rete di ricordare le informazioni passate e usarle per interpretare quelle future. È importante notare che i pesi **U**, **W** e **V** rimangono invariati a ogni passo temporale: non vengono ricalcolati a ogni iterazione, ma sono appresi durante la fase di addestramento e riutilizzati per elaborare l'intera sequenza.

Un esempio pratico.

Per capire il funzionamento di una RNN, immaginiamo di addestrarla a predire il carattere successivo in una parola, ad esempio "SOLE", un carattere alla volta.

- Al primo passo la rete riceve "S" come input. Partendo da uno stato iniziale nullo, calcola uno stato nascosto h_1 e produce un output che predice "O".
- Al passo successivo riceve "O", aggiorna il proprio stato e predice "L".
- Al terzo passo riceve "L", aggiorna di nuovo la memoria e predice "E".

Questo processo si ripete fino alla fine della sequenza: a ogni passo la rete riceve un nuovo input, aggiorna il proprio stato interno e produce una nuova previsione basata sia sull'informazione appena ricevuta sia sulla memoria accumulata.

La presenza del collegamento ricorrente (pesi **W**) è cruciale: senza di esso, ogni previsione dipenderebbe soltanto dall'ultimo input ricevuto e non dal contesto complessivo. In altre parole, la rete perderebbe la capacità di comprendere la struttura sequenziale dei dati, che è invece la caratteristica principale delle RNN.

3.5 Applicazioni cliniche delle reti neurali

Le reti neurali hanno trasformato soprattutto le analisi end-to-end su immagini, segnali e testi clinici. Le applicazioni della diagnostica per immagini verranno ampiamente approfondite nel prossimo capitolo, in cui verranno presentate le *Convolutional Neural Networks* (CNN) che apprendono direttamente dai pixel a riconoscere, localizzare e segmentare strutture patologiche: dalla polmonite nelle radiografie del torace alla classificazione dermoscopica di lesioni cutanee, dallo screening autonomo della retinopatia diabetica su fundus e OCT alla patologia digitale con rilevazione di metastasi e mitosi su vetrini. Le stesse architetture, adattate alla segmentazione, delineano organi e lesioni su RM e TC e supportano la stadiazione o il *grading*, mentre tecniche di *image enhancement* migliorano la qualità diagnostica delle immagini.

Sui segnali biologici, architetture ricorrenti (GRU/LSTM), convoluzionali temporali e, sempre più spesso, Transformer per serie storiche (che vedremo nel capitolo 5) analizzano in tempo reale ECG ed EEG per individuare aritmie, *pattern* epilettiformi e fasi del sonno, o prevedono a breve termine l'andamento di parametri fisiologici, come la glicemia nei pazienti con microinfusore. In terapia intensiva, modelli sequenziali anticipano crisi cliniche con minuti o ore di anticipo, abilitando un monitoraggio proattivo. In questo solco si colloca anche il *Personalized Metabolic Avatar* (PMA) basato su GRU, che integra dati da indossabili per stimare la risposta

individuale a schemi nutrizionali e supportare piani personalizzati di gestione del peso (Abeltino et al., 2022).

Quando l'obiettivo è la predizione di *outcome* clinici, le reti profonde integrano sorgenti eterogenee — clinica, laboratorio, genomica e imaging — in modelli multimodali capaci di cogliere interazioni complesse. In oncologia si stimano recidiva e sopravvivenza e si predice la risposta a regimi terapeutici; in cardiologia si valutano i rischi di eventi maggiori e si ottimizzano dosaggi (come nel caso del warfarin) combinando parametri clinici e genetici. Queste previsioni guidano la personalizzazione del follow-up e delle strategie terapeutiche, intensificando l'attenzione sui pazienti ad alto rischio.

3.6 Conclusione

Le reti neurali artificiali rappresentano oggi uno degli strumenti più potenti dell'Intelligenza Artificiale applicata alla medicina, capaci di affrontare compiti complessi con risultati che in alcuni casi si avvicinano o superano quelli dell'esperienza clinica umana. Il loro principale punto di forza è la capacità di apprendere direttamente dai dati grezzi: non richiedono che vengano codificate manualmente regole diagnostiche, ma riescono a estrarre autonomamente *pattern* nascosti anche in *dataset* molto vasti. Questo le rende estremamente adatte all'analisi dei big data clinici, dove spesso le relazioni tra variabili non sono immediatamente evidenti. Una rete ben addestrata può raggiungere livelli di accuratezza e sensibilità molto elevati, cogliendo minime variazioni nei segnali o nelle immagini e producendo diagnosi con rapidità estrema, al punto da permettere screening automatizzati su larga scala. Inoltre, la possibilità di aggiornare continuamente i modelli con nuovi dati consente un miglioramento progressivo delle prestazioni e una notevole adattabilità a contesti diversi.

Accanto a questi vantaggi, emergono però limiti significativi. Anzitutto, le reti neurali dipendono in modo cruciale dalla quantità e dalla qualità dei dati: *dataset* limitati, poco rappresentativi o distorti rischiano di generare modelli inaccurati o addirittura pericolosi. A ciò si aggiunge il problema della scarsa interpretabilità: le decisioni della rete derivano da combinazioni di migliaia o milioni di pesi, difficili da tradurre in ragionamenti clinici comprensibili. Per questo si parla di "scatola nera", un limite delicato in un ambito dove comprendere il perché di una diagnosi è spesso fondamentale. Esistono approcci di intelligenza artificiale spiegabile, come le mappe di attenzione, ma la trasparenza rimane una sfida aperta. Altri rischi riguardano la possibilità di *bias*: se i dati di partenza sono sbilanciati, la rete può amplificare pregiudizi sistematici e produrre errori inaccettabili, come falsi negativi

o falsi positivi che in medicina hanno conseguenze potenzialmente gravi. Non vanno trascurati infine gli ostacoli pratici, come le risorse computazionali necessarie, le competenze specialistiche, i vincoli normativi e le questioni legate alla privacy e alla responsabilità legale.

Le reti neurali hanno già dimostrato di poter affiancare con successo i clinici in molteplici compiti – dalla diagnostica per immagini alla predizione di eventi clinici – ma non devono essere considerate soluzioni infallibili. Sono strumenti potenti che richiedono cautela, validazione rigorosa e un'integrazione attenta nei flussi di lavoro, sempre sotto la supervisione del medico. Solo attraverso la collaborazione tra ingegneri, data scientist, medici e istituzioni sarà possibile garantire un utilizzo etico, equo ed efficace di queste tecnologie, trasformandole in reali alleate della medicina senza perdere di vista il ruolo insostituibile del giudizio umano.

Di seguito alcune **domande conclusive** per stimolare la riflessione e la discussione in aula:

-Secondo voi, quali possono essere i **maggiori ostacoli o rischi** nell'uso di reti neurali in ambito clinico? Pensate sia più critico il problema dei *bias* nei dati, la mancanza di trasparenza, o altro? Come si potrebbero mitigare questi rischi?

-In quali situazioni cliniche ritenete **fondamentale l'interpretabilità** di un modello di intelligenza artificiale? Ad esempio, accettereste un "consiglio" diagnostico da una rete neurale senza spiegazioni? Discutete l'importanza di poter spiegare le decisioni di un'IA nel rapporto medico-paziente.

-Provate a immaginare una **nuova applicazione** delle reti neurali in medicina che non sia stata citata in questa lezione. Quale problema risolverebbe? Perché una rete neurale sarebbe adatta a quel compito? Confrontate le vostre idee con quelle dei colleghi per esplorare futuri possibili sviluppi dell'IA in ambito medico.

E6 Esercitazione 6: Reti Neurali

In questa esercitazione useremo *Orange* per sperimentare con le reti neurali artificiali sul *dataset Diabetes*, lo stesso che abbiamo utilizzato nella seconda esercitazione, che si focalizzava sugli algoritmi di classificazione. Anche questa volta l'obiettivo è classificare i pazienti in due categorie: diabetici e non diabetici. Per farlo costruiremo tre reti neurali con complessità diversa, per osservare direttamente i fenomeni di *underfitting*, good fit e *overfitting*.

E6.1 Il workflow Orange

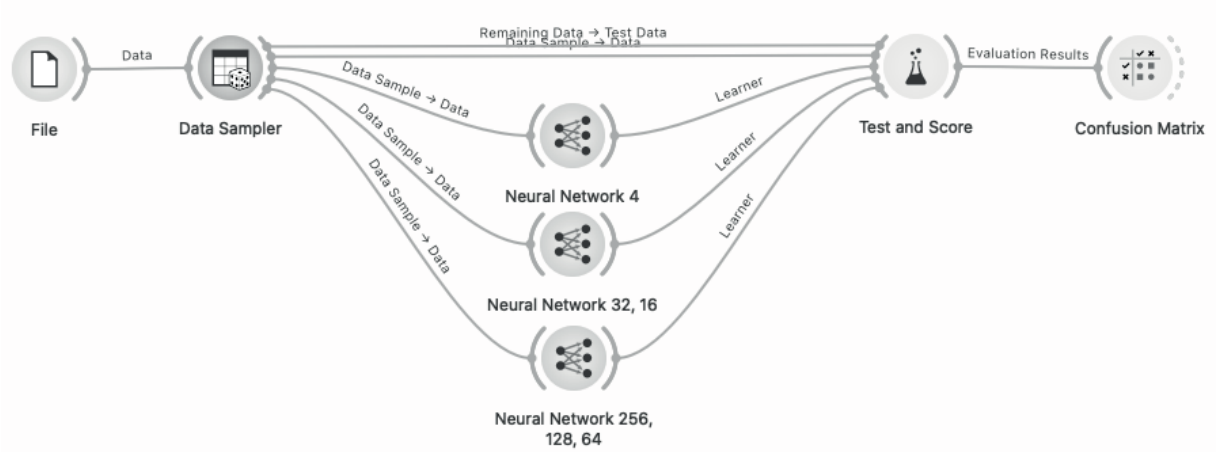

Figura 46. Workflow *Orange* completo per la sesta esercitazione. I dati vengono caricati e suddivisi in training e test, quindi utilizzati per addestrare tre reti neurali con architetture di diversa complessità. Le performance dei modelli vengono confrontate con il *widget Test and Score* e visualizzate tramite la Confusion Matrix, per identificare l'architettura più adatta al dataset.

Il primo passo consiste nel caricare il *dataset* tramite il *widget File* e nel collegarlo a un *Data Table* per visualizzarne la struttura. Ogni riga rappresenta un paziente, descritto da diverse caratteristiche cliniche (come il numero di gravidanze, la pressione sanguigna o lo spessore cutaneo), mentre la colonna *Diabetes* rappresenta la variabile da predire. Si tratta quindi di un tipico problema di classificazione binaria.

Il *workflow Orange* completo per questa esercitazione è mostrato in Figura 46.

E6.2 Il widget Neural Network

A questo punto possiamo passare alla costruzione dei modelli. Per rendere l'esperimento più interessante, creeremo tre diverse reti neurali, ognuna con un livello di complessità differente, in modo da osservare i fenomeni di *underfitting*, good fit e *overfitting*. La prima rete sarà molto semplice, con un solo livello nascosto composto da quattro neuroni. Si tratta di un modello con pochissima capacità

rappresentativa, destinato a non riuscire a cogliere tutte le relazioni tra le variabili: è la situazione tipica dell'*underfitting*. La seconda rete sarà leggermente più complessa, con due livelli nascosti da sedici e otto neuroni ciascuno. Questa architettura intermedia rappresenta un buon compromesso tra semplicità e capacità predittiva, ed è quella che ci aspettiamo funzioni meglio. Infine, costruiremo una terza rete estremamente complessa, con tre livelli nascosti da 256, 128 e 64 neuroni. Questo modello ha un'enorme capacità di apprendimento e rischia di adattarsi troppo IA dati di training, imparandoli quasi a memoria: una situazione tipica di *overfitting*.

Dal punto di vista pratico, la costruzione delle tre reti neurali in *Orange* è molto semplice e intuitiva, grazie al *widget Neural Network*. Una volta collegato il *widget* al *Data Sampler* o direttamente al *dataset*, con un clic si apre una finestra di configurazione in cui è possibile specificare i parametri principali del modello. Tra questi, l'opzione più importante per il nostro esperimento è **Neurons in *hidden layer*s**, che permette di definire il numero di strati nascosti della rete e il numero di neuroni presenti in ciascuno strato. Ogni architettura si scrive come una sequenza di numeri separati da virgole, dove ciascun numero rappresenta il numero di neuroni in un livello nascosto. Ad esempio, scrivere *16, 8* indica che la rete avrà due *hidden layer*, il primo con sedici neuroni e il secondo con otto neuroni.

Per il nostro confronto, configuriamo il *widget* in questo modo:

- Nel caso di ***underfitting***, inseriamo semplicemente *4* come valore in *Hidden layers*. In questo modo la rete avrà un unico livello nascosto composto da quattro neuroni, troppo pochi per catturare relazioni complesse tra le variabili del *dataset*.
- Nel caso del **good fit**, inseriamo *16, 8*. Così la rete avrà due *hidden layer*, con rispettivamente sedici e otto neuroni. Si tratta di un'architettura più ricca, capace di modellare relazioni non lineari senza diventare eccessivamente complessa.
- Infine, per l'***overfitting***, inseriamo *256, 128, 64*. Questa configurazione crea tre *hidden layer* molto ampi, che conferiscono alla rete un'enorme capacità di apprendimento. Vedremo come dai questo porti a un'accuratezza perfetta sul *training set*, ma a un peggioramento delle performance sul *test set*.

Tutti gli altri parametri del *widget*, come la funzione di attivazione, l'ottimizzatore (*solver*) o le tecniche di regolarizzazione, possono essere lasciati IA valori predefiniti: lo scopo dell'esercitazione non è ottimizzare finemente la rete, ma osservare l'impatto che il solo cambiamento dell'architettura ha sulle performance.

E6.3 Il widget Test and Score

Per valutare i tre modelli, colleghiamo ciascuna rete al *widget Test & Score*, che ci permette di calcolare automaticamente le principali metriche di performance sia sul *training set* sia sul *test set*. Le metriche che consideriamo sono l'***accuracy***, che indica la percentuale complessiva di predizioni corrette; la ***precision***, che misura quanti dei pazienti classificati come diabetici lo sono davvero; il ***recall***, che indica la capacità del modello di individuare correttamente i pazienti diabetici; e l'***F1-score***, che rappresenta una sintesi equilibrata tra *precision* e *recall*.

I risultati mostrano in modo chiaro i tre comportamenti attesi, come possiamo vedere dalla Tabella 4. La rete più piccola, con soli quattro neuroni, ottiene performance basse sia sul training sia sul *test set*: semplicemente non ha abbastanza capacità per rappresentare adeguatamente la complessità del problema, ed è un tipico esempio di *underfitting*. La rete intermedia, con due livelli da sedici e otto neuroni, raggiunge buone performance, sia in addestramento sia in test: non impara a memoria i dati, ma riesce a generalizzare, ed è quindi un modello ben bilanciato. La rete più grande, infine, con centinaia di neuroni distribuiti su tre strati, impara perfettamente i dati di training (ottenendo un'accuratezza pari al 100%), ma mostra un calo significativo delle prestazioni sul *test set*: è un caso evidente di *overfitting*, in cui il modello memorizza i dati invece di estrarre regole generali valide per nuovi pazienti.

***Test set*:**				
Modello	**CA**	**P**	**R**	**F1**
Rete Neurale (4)	0.712	0.595	0.759	0.667
Rete Neurale (16, 8)	0.732	0.655	0.621	0.637
Rete Neurale (256, 128, 64)	0.706	0.623	0.569	0.595
***Training set*:**				
Modello	**CA**	**P**	**R**	**F1**
Rete Neurale (4)	0.737	0.589	0.757	0.662
Rete Neurale (16, 8)	0.831	0.773	0.714	0.743
Rete Neurale (256, 128, 64)	1.000	1.000	1.000	1.000

Tabella 4 .Performance delle diverse reti neurali sul test set e sul training set.

E6.4 Conclusioni

Questa esercitazione mette in evidenza un principio cardine del *machine learning*: la complessità di un modello deve sempre essere calibrata rispetto alla quantità e

alla qualità dei dati disponibili. Un modello troppo semplice rischia di non cogliere le relazioni nascoste, mentre uno troppo complesso tende a memorizzare il rumore presente nei dati, perdendo la capacità di generalizzare. In ambito clinico, questo si traduce nella necessità di progettare con attenzione l'architettura delle reti neurali, tenendo conto non solo del problema specifico, ma anche delle dimensioni e caratteristiche del *dataset* e degli obiettivi dell'analisi.

I risultati sperimentali ottenuti in questa semplice esercitazione lo confermano chiaramente. La rete più semplice, con un unico livello da 4 neuroni, ottiene sul *test set* un'accuratezza di circa il 71% con un F1 di 0.667: prestazioni non eccezionali e in linea con quanto osservato sul *training set*, dove i valori si mantengono simili. La rete intermedia, con due livelli da 16 e 8 neuroni, raggiunge l'accuratezza più alta sul *test set* (circa 73%) con un F1 di 0.637, mostrando quindi il miglior compromesso tra capacità di apprendimento e generalizzazione. Diversa la situazione per la rete più complessa, con tre livelli profondi da 256, 128 e 64 neuroni: se da un lato ottiene sul *training set* prestazioni perfette (accuratezza e F1 pari a 1.0, chiaro segnale di *overfitting*), dall'altro mostra un netto calo sul *test set*, con un'accuratezza intorno al 71% e un F1 di appena 0.595, inferiore persino alla rete più semplice.

Questi risultati illustrano in modo concreto l'effetto del ***trade-off* tra *bias* e varianza**: un modello troppo semplice soffre di *underfitting*, mentre uno eccessivamente complesso cade nell'*overfitting*. Il modello intermedio rappresenta quindi la scelta più appropriata per questo *dataset*, riuscendo a mantenere un buon equilibrio tra capacità predittiva e robustezza.

In conclusione, l'esercitazione sottolinea come la progettazione di una rete neurale non possa basarsi su tentativi casuali, ma debba essere guidata da una valutazione attenta delle performance su training e *test set*. Solo così si può evitare il rischio di reti troppo semplici o troppo complesse, garantendo modelli realmente utili in contesti applicativi sensibili come quello clinico.

Giuseppe **Maulucci** • Tommaso **Marchetti** • Cassandra **Serantoni**

Intelligenza Artificiale in Medicina

Dalla teoria alla pratica: esercitazioni guidate e applicazioni cliniche

Accedi ai **contenuti digitali** › **Espandi** le tue risorse › con un libro che **non pesa** e si **adatta** alle dimensioni del tuo **lettore**

All'interno del volume il **codice personale** e le istruzioni per accedere ai **contenuti digitali**.
L'accesso alle risorse digitali è **gratuito** ma limitato a **18 mesi dalla attivazione del servizio**.

EdiSES università

ISBN 978-88-3623-245-1