UNIVERSITÀ
DEGLI STUDI
DI PADOVA

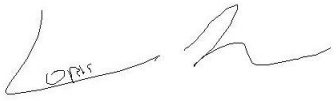
DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea Triennale in Ingegneria Informatica

Reti Neurali Ricorrenti per l'estrazione di informazioni da farmaci

Laureando:
Lorenzo BOCCALON

Relatore:
Prof. Ing. Loris NANNI



Data di Laurea: 18 Marzo 2019
Anno Accademico 2018/2019

Indice

1 Riassunto

Dato un composto medicinale, è possibile prevedere a quale classe (o quali classi) ATC di primo livello appartiene? Predire le classi di appartenenza di un composto può essere utilizzato per dedurne i principi attivi, i suoi effetti terapeutici, farmacologici e le proprietà chimiche. Ciò permetterebbe uno sviluppo più veloce ed efficiente di nuovi farmaci. Utilizzando i metodi dell'apprendimento automatico (o machine learning, in inglese) sono state sviluppate varie soluzioni al problema di classificazione. In questa tesi viene proposta una nuova soluzione, composta da un estrattore di caratteristiche implementato da una rete neurale ricorrente e un classificatore specifico per i problemi multi classe, con classi non esclusive. I risultati ottenuti vengono comparati tramite degli indici di prestazione ai migliori risultati sul campo; essi si allineano allo stato attuale dell'arte, pur non superando la migliore soluzione disponibile finora.

2 Introduzione

2.1 Reti Neurali

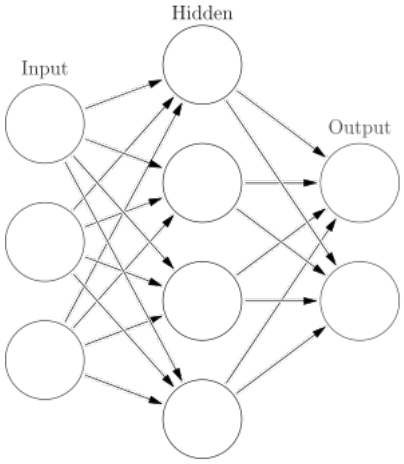


Figura 1: rappresentazione grafica di una ANN

Nel campo dell'apprendimento automatico, una *rete neurale artificiale* (abbreviato in ANN, "Artificial Neural Network", in inglese), normalmente chiamata solo "rete neurale" (abbreviato in NN, "Neural Network", in inglese), è un modello computazionale ispirato dai sistemi nervosi biologici, come il cervello umano [19]. Il modello è costituito da semplici unità di elaborazione chiamate *neuroni*, che, connesse tra di loro secondo una più o meno sofisticata struttura, realizzano un processo di calcolo. Tale sistema "impara" a risolvere un problema basandosi su esempi, generalmente senza essere programmato con nessuna regola specifica al problema; è un sistema adattivo che cambia la sua struttura basata su informazioni esterne o interne che scorrono attraverso la rete durante la fase di apprendimento. In una normale implementazione di ANN, il segnale di connessione tra due neuroni è rappresentato da un numero reale, e l'output di ciascun neurone è calcolato da una funzione. Le connessioni tra neuroni sono chiamati *pesi* e si occupano di trasferire informazioni attraverso la rete. Può essere presente un neurone extra, detto *BIAS*, il quale non è connesso in input con nessun altro neurone. I neuroni possono avere una *soglia di attivazione*, tale che il segnale di output viene emanato solo se il segnale aggregato di input supera tale soglia. Il segnale di input è calcolato tramite una *funzione di attivazione*. Tipicamente i neuroni sono organizzati in *strati* (*layer*, in inglese). Strati

differenti possono effettuare diversi tipi di trasformazioni sui propri input. Il segnale viaggia dal primo layer, detto *strato di ingresso* (*input layer*, in inglese) all'ultimo layer, detto *strato di uscita* (*output layer*, in inglese), passando attraverso uno o più layer, detti *strato nascosto* (*hidden layer*, in inglese). La rete formata può essere vista come un grafo orientato, dove i neuroni sono i nodi e i collegamenti sono gli archi (Figura 1). I pesi e i neuroni vengono modificati dal processo di apprendimento, il quale è dettato da una regola di apprendimento [20].

Riassumendo, le reti neurali sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione. Esse possono essere utilizzate per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare.

2.1.1 Componenti di una Rete Neurale

Una rete neurale è formata dai seguenti elementi

- **Neuroni**: sono le unità elementari di una NN e sono modellati graficamente da nodi in un grafo. Sono funzioni matematiche che ricevono uno o più input e li somma per produrre un output, detto *attivazione*. Ogni input è pesato separatamente, e la somma viene passata attraverso una funzione non lineare detta funzione di attivazione.

- **Connessioni, Pesi e BIAS**: I collegamenti della rete sono realizzati da connessioni neuronali, ogni connessione trasferisce l'output di un neurone i all'input del neurone j. In questo senso i è il predecessore di j e j è il successore di i. Ad ogni connessione è associato un peso w_{ij}. I neuroni formano dei layer, generalmente si identificano tre layer principali: ingresso (*input*), nascosto (*hidden*) e uscita (*output*). Un valore w_{0j} detto BIAS può essere presente e ha il ruolo di modificare l'ingresso della funzione di attivazione del neurone successore.

- **Funzione di Attivazione**: La funzione di attivazione (o propagazione) σ calcola l'input del neurone j tramite gli output forniti dai suoi predecessori e tipicamente è della forma

$$o_j = \sigma(\sum_i o_i w_{ij} + w_{0j}) \tag{1}$$

  Una funzione di attivazione usata frequentemente per problemi di classificazione [9] è la *funzione logistica*, una funzione con grafico a "S". Essa è definita nel seguente modo:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Un'altra funzione di attivazione importante [8] è *ReLU* ("Rectifier Linear Unit", rettificatore), definita come

$$f(x) = x^+ = max(0, x) \tag{3}$$

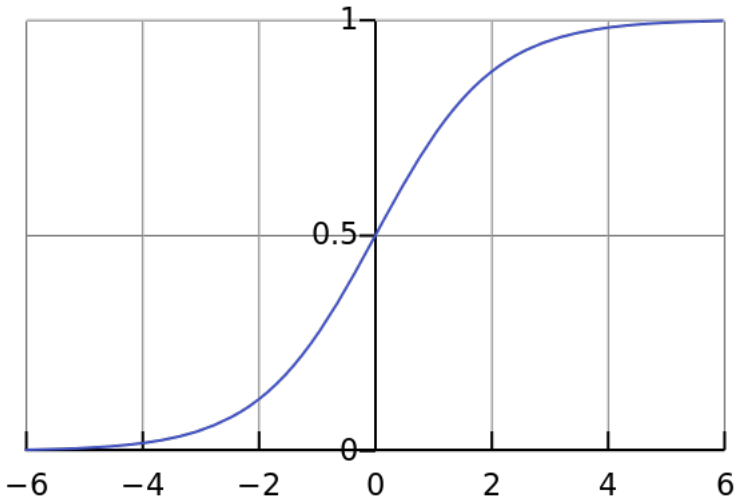


Figura 2: Grafico funzione logistica

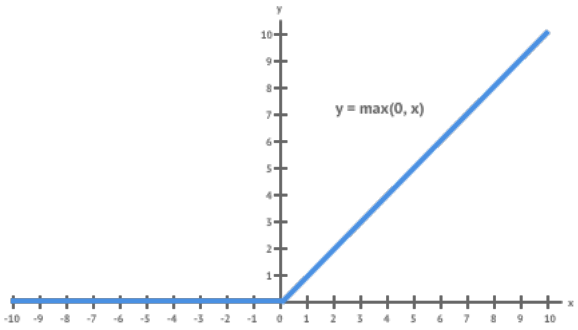


Figura 3: Grafico funzione ReLU

- **Regola di Apprendimento**: La regola di apprendimento è un algoritmo che modifica i paramatri della rete neurale, in modo che un dato input produca un output favorito. Tipicamente il processo di apprendimento consiste nel modificare i pesi e i BIAS della rete. lo standard de facto per l'allenamento delle reti neurali artificiali è l'uso

di *back-propagation* in combinazione di *stocastic gradient descent* [12]: La retropropagazione dell'errore ("backward propagation of errors" in inglese, solitamente abbreviato in back-propagation), è un metodo che calcola il gradiente di una funzione errore, utilizzato per distribuire aggiustamenti ai pesi utilizzati nella rete all'indietro attraverso i layer della rete. Una funzione errore (o di costo) è una funzione che indica quanto distante è l'output della rete dal valore corretto. Per aggiustare i pesi propriamente, si applica un metodo di ottimizzazione non-lineare detto *gradient descent* (abbreviato in GD, "discesa del gradiente", in italiano). Il metodo consiste nel calcolare la derivata della funzione di errore rispetto ai pesi, e cambiare i pesi facendo decrescere l'errore (da cui ne deriva il nome, cioè scendere lungo la superficie della funzione di errore). Una approssimazione stocastica di GD è *stocastic gradient descent* (abbreviato in SGD, "discesa stocastica del gradiente", in italiano). SGD opera similmente a GD ma, ad ogni iterazione di allenamento, sostituisce il valore esatto del gradiente della funzione costo con una stima ottenuta valutando il gradiente solo su un sottinsieme dei parametri della rete. Il vantaggio di SGD è la velocità computazionale.

2.2 Reti Neurali Ricorrenti

Il tipo di rete neurale più semplice è la rete *feed-forward*. Una NN feed-forward (rete neurale con flusso in avanti, in italiano) è una rete neurale artificiale dove le connessioni tra neuroni non formano cicli (figura 1). In questo tipo di rete neurale le informazioni si muovono solo in una direzione, avanti, rispetto ai nodi d'ingresso, attraverso lo strato nascosto fino ai nodi d'uscita. Le reti feed-forward non hanno memoria di input avvenuti a tempi precedenti, per cui l'output è determinato solamente dall'attuale input. Questo tipo di rete viene anche detto *percettrone*.

Le reti neurali ricorrenti (abbreviato in RNN, "Recurrent Neural Network", in inglese) si occupano di risolvere questo problema. Le RNN sono una classe di reti neurali in cui i valori di uscita di uno strato di un livello superiore vengono utilizzati come ingresso ad uno strato di livello inferiore. All'interno della rete sono presenti dei loop che permettono alle informazioni di persistere nel tempo.
Nella figura 4, una parte di rete neurale, A, collega l'input X_t all'output h_t. Un loop permette all'informazione di persistere tra uno passo e l'altro. Questa struttura a catena predispone le RNN all'utilizzo di strutture dati simili, come sequenze e liste. Di conseguenza le RNN risultano particolarmente efficaci per risolvere problemi di speech-recognition, language modeling, tra-

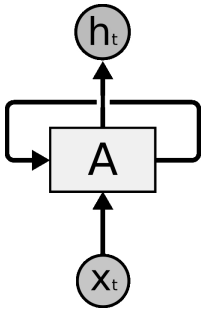


Figura 4: schema di RNN avvolta

duzione e molto altro [18][13]. Per avere un'idea del funzionamento basti pensare alla comprensione di una frase: il senso di ogni parola è in funzione delle parole lette in precedenza e la sequenza di parole crea il significato della frase. Una rete neurale ricorrente può essere pensata come una moltitudine di copie della stessa rete, ognuna delle quali passa un messaggio alla successiva. L'informazione in uscita al tempo t viene passata in ingresso al tempo $t + 1$, in questo modo la rete "ricorda" l'ultima cosa che ha fatto, quindi si comporta come se possedesse una memoria. Tutte le RNN hanno una forma a catena di moduli ripetuti di NN. Teoricamente, le RNN standard dovrebbero essere in grado di ricordare anche le dipendenze a lungo termine nelle sequenze di input, tuttavia in [1] sono state evidenziate delle ragioni per le quali è molto difficile che accada. Il problema è di natura computazionale: quando si allena una RNN usando back-propagation, il gradiente "svanisce" (cioè tende a 0) oppure "esplode" (cioè tende a infinito), a causa dei calcoli effettuati nel processo, che utilizzano una precisione finita dei numeri.

2.3 Reti LSTM

Le reti *Long Short Term Memory* (abbreviato in LSTM, in inglese) sono un tipo particolare di RNN dotate di una memoria a lungo termine. Le LSTM sono composte da unità omonime. Un'unità LSTM è composta da 4 layer:

- **cella**: l'unità di memoria del modulo LSTM, è responsabile di tenere traccia delle dipendenze tra gli elementi nella sequenza di input;
- **input-gate**: controlla la misura in cui un nuovo valore scorre nella cella;
- **output-gate**: controlla la misura in cui il valore nella cella viene utilizzato per calcolare l'attivazione dell'output dell'unità LSTM;
- **forget-gate**: controlla la misura in cui un valore rimane nella cella.

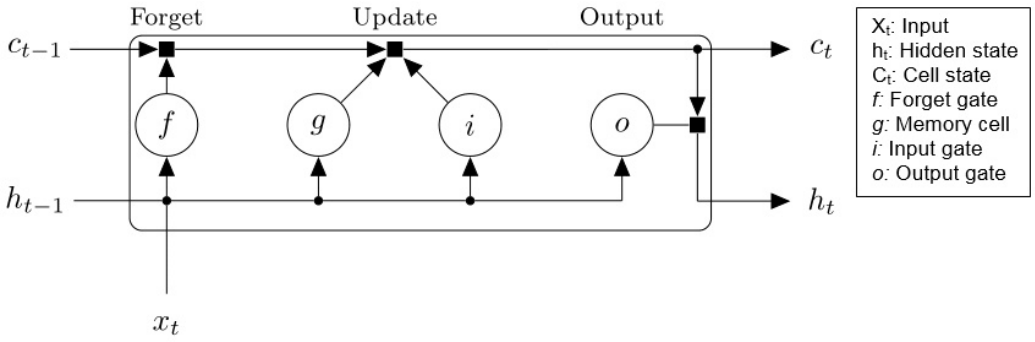


Figura 5: unità LSTM

LSTM risolve il problema del *vaniscing gradient* (cioè il gradiente tendente a 0) utilizzando i gate per selezionare le informazioni rilevanti, dimenticando quelle non rilevanti [7]. Tuttavia possono ancora soffrire dell'*exploding gradient*. In Figura 5 è rappresentato un modulo LSTM, ogni linea trasporta un vettore di valori numerici da un output del nodo all'input dei nodi successivi. I quadrati rappresentano operazioni punto-punto, come addizione tra vettori. Linee che si congiungono rappresentano l'unione dei vettori, mentre le linee che si biforcano rappresentano la copiatura e trasmissione del vettore su nodi differenti.

2.4 Estrazione di informazioni da farmaci

2.4.1 Classificazione

La classificazione è il processo di predire la classe di appartenenza di un insieme di dati in input. Le classi vengono dette anche target o categorie. La modellizzazione predittiva di classificazione è il compito di approssimare una funzione mappa f dalle variabili di input X alle variabili discrete di output y. In questo lavoro si vuole creare un modello che identifica le classi di appartenenza di farmaci. Il tipo di classificazione è *multi-class multi-label*, cioè ci sono più classi possibili (multi-class, MC) e le classi non sono mutualmente esclusive, cioè un input può appartenere a più classi contemporaneamente (multi-label, ML), quindi avere più etichette (o label, in inglese) che indicano a quali classi appartiene. Il classificatore utilizza dei dati di allenamento o addestramento per aggiustare i propri pesi e connessioni e mettere in relazione un dato input alle classi di output. L'insieme di dati utilizzato per l'allenamento è detto *training dataset*. Dopo che il classificatore è stato addestrato sul training dataset, è in grado di predire le classi di appartenenza di un insieme di uno o più input che non vi appartengono, detto *testing dataset*, con una certa precisione. Insieme alla precisione, vengono calcolati vari indici che servono a determinare la qualità del classificatore. La classificazione appartiene alla categoria dell'*apprendimento supervisionato*. L'apprendimento supervisionato è una tecnica di apprendimento automatico che mira a istruire un sistema informatico in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input sulla base di una serie di esempi ideali, costituiti da un insieme di coppie di input e di output, che gli vengono inizialmente forniti.

I classificatori inoltre si distinguono in due categorie:

- **Lazy learners**: i Lazy learners (LL) immagazzinano il training dataset e aspettano finché un dato di testing viene immesso. Quando succede, la classificazione avviene basandosi sui dati più simili all'interno del training dataset. Il tempo di addestramento di un LL è breve, tuttavia impiegano un tempo elevato per la previsione. Esempio di algoritmo LL: *k-nearest neighbor*.

- **Eager learners**: gli Eager learners (EL) costruiscono un modello di classificazione basato sul training dataset prima di ricevere dati da classificare. Deve essere in grado di formulare una singola ipotesi che copre l'intero spazio delle istanze. Per la costruzione del modello gli EL impiegano molto tempo di addestramento, tuttavia la previsione è molto veloce. Esempio di algoritmo EL: *reti neurali LSTM*.

2.4.2 Sistema ATC

Il sistema di classificazione anatomico, terapeutico e chimico, sigla ATC (dall'inglese Anatomical Therapeutic Chemical classification system), viene usato per la classificazione sistematica dei farmaci ed è controllato dall'Organizzazione Mondiale della Sanità (OMS). Il sistema considera simultaneamente la distribuzione anatomica, gli effetti terapeutici e le caratteristiche chimiche [2]. L'ATC è un sistema di classificazione di tipo alfanumerico che suddivide i farmaci in base a uno schema costituito da 5 livelli gerarchici e include diverse classi che si sovrappongono.

La scoperta di farmaci è un processo costoso sia in termini di tempo che di denaro: la procedura per portare un singolo farmaco sul mercato può impiegare più di un decennio e costare milioni di USD. Il motivo principale dei fallimenti nei test clinici è la mancanza di efficacia e gli effetti collaterali dei farmaci [16]. Per contrastare queste difficoltà e con l'obbiettivo di trovare un nuovo uso per i farmaci già approvati, si vuole sviluppare degli strumenti che predicano le indicazioni terapeutiche di un farmaco e i suoi effetti collaterali accuratamente. Un metodo per realizzare questo obbiettivo è basato sulla predizione automatica delle classi ATC di un dato composto, le quali sono utili per dedurne gli ingredienti, le proprietà terapeutiche, farmacologiche e chimiche, e sostanzialmente aumentare la velocità con cui si sviluppa un farmaco. Il grande vantaggio di utilizzare il codice ATC (il codice che indica la classificazione di un farmaco secondo ogni livello ATC) è che dà una guida riguardo l'uso clinico del farmaco e aiuta la ricerca farmacologica nell'identificazione delle indicazioni terapeutiche e degli effetti collaterali.

2.4.3 Primo livello

Il primo livello contiene il gruppo anatomico principale ed è il riferimento per le classi di output del classificatore che si vuole realizzare. Le classi di primo livello sono riportate nella Tabella 1.

2.4.4 Secondo, terzo, quarto e quinto livello

Il secondo livello contiene il gruppo terapeutico principale (contraddistinto da un numero di due cifre). Il terzo livello contiene il sottogruppo terapeutico farmacologico (contraddistinto da una lettera dell'alfabeto). Il quarto livello contiene il sottogruppo chimico-terapeutico farmacologico (contraddistinto da una lettera dell'alfabeto). Il quinto livello contiene il sottogruppo chimico (contraddistinto da un numero di due cifre) ed è specifico per ogni singola sostanza chimica. Questi livelli non vengono presi in considerazione dal classificatore presentato.

Tabella 1: Classi ATC di primo livello

A	apparato gastrointestinale e metabolismo
B	sangue e sistema emopoietico
C	apparato cardiovascolare
D	apparato tegumentario e pelle
G	apparato genito-urinario e ormone sessuale
H	sistema endocrino, esclusi ormoni sessuali e insulina
J	anti-infettivi per uso sistemico
L	antineoplastici e immunomodulatori
M	sistema muscolare - sistema scheletrico e articolazioni
N	sistema nervoso
P	prodotti antiparassitari, insetticidi e repellenti
R	apparato respiratorio
S	organi di senso
V	vari

2.4.5 Esempio

Nella classificazione ATC il Prednisolone (un ormone steroideo con potenti proprietà anti-infiammatorie) in prodotti a singolo ingrediente è indicato con più codici ATC, in base all'uso terapeutico diverso o diversa formulazione. Nella Tabella 2 si nota che il farmaco appartiene a 6 classi ATC (di primo livello) contemporaneamente.

Tabella 2: Codici ATC del Prednisolone

A07EA01	Agenti anti-infiammatori intestinali (clisteri e schiume)
C05AA04	Anti-emorroidi per uso topico (supposte)
D07AA03	Preparati dermatologici (creme, unguenti e lozioni)
H02AB06	Corticosteroidi per uso sistemico (pastiglie, iniezioni)
R01AD02	Decongestione nasale (spray/gocce nasali)
S01BA04	Oftamolgici (colliri)
S02BA03	Otologici (gocce per le orecchie)

3 Metodo proposto

Il metodo proposto per risolvere questo problema di classificazione è un *metodo ensemble*, composto da un estrattore di caratteristiche e da un classificatore. Il metodo ensemble è un metodo di apprendimento di insieme che utilizza algoritmi di apprendimento multipli per ottenere la miglior prestazione predittiva rispetto agli algoritmi presi singolarmente da cui è costituito [17].

3.1 Dataset

Il dataset di medicinali utilizzato in questa tesi è stato ottenuto da [2]. In totale contiene 3883 medicinali, numero denotato con N, divisi in 14 classi non esclusive (3295 appartenenti a una classe, 370 a due classi, 110 a tre classi, 37 a quattro classi, 27 a cinque classi e 44 a sei classi; nessun dato appartiene a più di sei classi). La somma di campioni tra le classi, denotato con N(vir) in [2] (i.e. numero di campioni virtuali di medicinali), è uguale a 4912 da cui si deduce che il numero medio di classi ATC per campione è $4912/3883 = 1.27$.

Tabella 3: Riassunto numero classi nel dataset

Classe ATC	**Numero di medicinali**
A	540
B	133
C	591
D	421
G	248
H	126
J	521
L	232
M	208
N	737
P	127
R	427
S	390
V	211
N(vir)	**4912**
N	**3883**

I descrittori utilizzati per rappresentare i medicinali sono basati sulle interazioni farmaco-farmaco e le correlazioni con le classi target che devono

essere predette. Dato un insieme di 14 classi ATC di primo livello, ogni campione può essere rappresentato con tre espressioni matematiche che riflettono la sua correlazione intrinseca con ogni classe, portando a un descrittore finale di $14 \times 3 = 42$ caratteristiche (o feature, in inglese). Il descrittore è costituito da un vettore di 42 numeri reali. Le tre differenti proprietà considerate sono: la massima interazione con i farmaci in ognuna delle 14 classi, la massima similarità strutturale con i farmaci in ognuna delle 14 classi e la similitudine dell'impronta molecolare nei 14 sottoinsiemi. Questi descrittori si possono scaricare dal materiale supplementare di [14].

3.2 Estrattore di caratteristiche

L'*estrazione di caratteristiche* (o feature extraction, in inglese) è una forma speciale di riduzione della dimensionalità dei descrittori utilizzati per rappresentare gli elementi del training set. Quando la dimensione del vettore che rappresenta i dati in ingresso (il vettore di rappresentazione grezzo o raw pattern vector, in inglese) è troppo elevata per l'esecuzione di un algoritmo o è presente ridondanza, i dati vengono convertiti in una rappresentazione di dimensione minore (il vettore delle caratteristiche o feature vector, in inglese). Il processo di trasformazione dei dati in ingresso in un insieme di caratteristiche è chiamato estrazione di caratteristiche. Lo scopo principale di questo processo è velocizzare i tempi di allenamento del classificatore e permettere una migliore generalizzazione. Il vettore delle caratteristiche viene poi utilizzato dal classificatore per predire le classi di appartenenza del vettore di input. In figura 6 è rappresentato uno schema del processo di trasformazione e previsione.

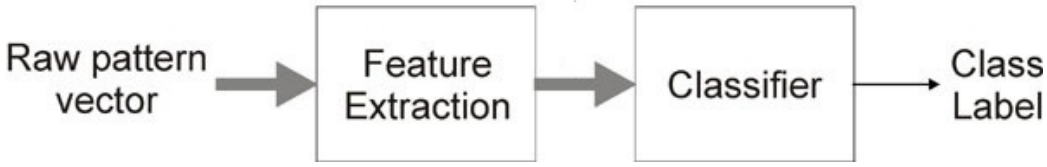


Figura 6: Processo di estrazione di caratteristiche e classificazione di un vettore di input.

Come estrattore di caratteristiche si è scelto di implementare una rete LSTM nell'ambiente di sviluppo MatLab. MatLab fornisce un architettura standard della rete già orientata ai problemi di classificazione, lasciando arbitrio nella

modifica mantenendo tuttavia una struttura di base. I componenti principali della rete LSTM sono il *sequence input layer* e il *layer LSTM*. Il sequence input layer inserisce sequenze di dati all'interno della rete. Il layer LSTM impara dipendenze a lungo termine tra le sequenze. Il diagramma in Figura 7 illustra l'architettura di una semplice rete LSTM per classificazione. Per predire le etichette delle classi, la rete termina con un *fully connected layer*, un *softmax layer* e un *classification output layer*.

Figura 7: Diagramma dell'architettura di una rete LSTM

Il fully connected layer moltiplica l'input per una matrice di pesi e di seguito aggiunge un vettore di bias. Il softmax layer applica la *funzione softmax* all'output del layer precedente. La funzione softmax normalizza il vettore v di dimensione n in input in un vettore u della stessa dimensione n di output, i cui n valori variano nell'intervallo $(0; 1)$ e la cui somma vale 1, i.e. $\sum_{i=1}^{n} u_i = 1$. La sequenza di layer termina con il classification output layer, il quale restituisce un vettore i cui n valori rappresentano n distribuzioni di probabilità. Il numero di features da estrarre è stato scelto uguale al numero di classi di output del classificatore. Questa scelta permette di poter utilizzare direttamente la rete LSTM per classificare.

3.3 Classificatore

Una volta ridotta la dimensionalità del training set si passa alla classificazione vera e propria. Il classificatore è una funzione che mappa un vettore di caratteristiche in un vettore di etichette. Il vettore di etichette indica a quali classi è stato assegnato il composto rappresentato dal vettore delle caratteristiche. Essendo il problema MC-ML, si deve utilizzare un classificatore ad hoc. In questa tesi vengono utilizzati e confrontati diversi classificatori realizzati da esperti nel campo del machine learning, implementati in MatLab. Elenco dei classificatori presi in esame:

- **Classificatore MLkNN**: il classificatore *Multi-Label k-Nearest Neighbor* (abbreviato in MLkNN) è un classificatore realizzato da [22] basato sull'algoritmo k-neareast neighbor (abbreviato in kNN). Come accennato in precedenza è un algoritmo LL; per determinare le etichette dell'input considera i k campioni nel training set più simili al campione

da classificare, detti vicini, e tramite un processo di voto fra i k vicini vengono scelte come etichette le più comuni.

- **Classificatore MLC**: Multi-Label-Classification (abbreviato in MLC) è una libreria MatLab/Octave realizzata da [10]. Questa libreria offre un ambiente per valutare, comparare e visualizzare i risultati di problemi MC-ML. Il pacchetto comprende molti strumenti utili nel campo del machine learning, tra cui reti neurali, riduttori di dimensionalità, regressori e molto altro. In questo lavoro viene utilizzato solo il classificatore MC-ML.

- **Classificatore MatLearn**: MatLearn è una libreria di strumenti di machine learning implementata in MatLab da [6]. Contiene una vasta gamma di implementazioni degli algoritmi più utilizzati nel campo del machine learning per affrontare problemi di classificazione, regressione, clustering e molto altro. In questo lavoro viene utilizzato il classificatore MC-ML.

- **LSTM come classificatore**: Utilizzando il vettore v di uscita del classification output layer della rete LSTM come vettore di probabilità, dove l'elemento v_i rappresenta la probabilità che il composto in input appartenga alla classe i-esima, confrontandolo con una soglia τ di attivazione decisa arbitrariamente, si può trasformare il vettore di probabilità in un vettore u di etichette, dove l'etichetta i-esima indica, nel caso la probabilità sia maggiore della soglia, l'appartenenza alla classe corrispondente (indicato con un $+1$) oppure la non appartenenza (indicato con un -1).

$$u_i = \begin{cases} +1 & \text{se } v_i \geq \tau \\ -1 & \text{altrimenti} \end{cases} \quad \forall i = 1...N \tag{4}$$

In equazione 4 è indicata la funzione che mappa probabilità in etichette; N indica il nuemro di classi. La scelta della soglia migliore viene effettuata in modo empirico, facendo un test per ogni soglia e confrontando gli indicatori di prestazione.

4 Risultati sperimentali

4.1 Protocollo di testing

Gli esperimenti sul dataset sono stati condotti seguendo il *metodo jackknife*. È un metodo di validazione di modelli statistici; consiste in una procedura di ricampionamento utilizzata per stimare l'errore standard di una grandezza, ed è considerato il meno arbitrario tra i metodi di validazione usati in statistica [5]. Dato un training set di dimensione n, la stima jackknife di un parametro è data dall'aggregazione delle stime dei sotto insiemi di taglia $(n-1)$. La procedura viene ripetuta in modo da escludere una volta tutti gli elementi dall'insieme, infine i risultati vengono mediati considerando tutte le simulazioni. Il metodo permette di valutare come i risultati di un'analisi statistica saranno generalizzati a un insieme di dati indipendente dal trainig set.

4.2 Indicatori di prestazione

Seguendo i recenti lavori pubblicati nel campo [3][15], vengono utilizzati 5 indicatori di prestazione (o performance values, in inglese) per valutare e confrontare i risultati del classificatore presentato. Sia L_k il sotto-insieme di label "effettive" osservate per il k-esimo campione e sia L_k^* il sotto-insieme di label predette per il k-esimo campione:

- In equazione 5 viene definito l'**Aiming** (o precisione), misura l'esattezza o fedeltà, cioè il rapporto di label predette correttamente su tutte le label predette;
- In equazione 6 viene definito il **Coverage** (o richiamo/recupero), misura la completezza, cioè il rapporto di label predette correttamente su tutte le label effettive;
- In equazione 7 viene definita l'**Accuracy** (o accuratezza), misura il rapporto di label predette correttamente su tutte le label, cioè l'unione di quelle predette e di quelle effettive;
- In equazione 8 viene definito l'**Absolute true** (o vero positivo) misura il rapporto tra gli eventi di previsione perfettamente corretti rispetto agli eventi di previsione totali;
- In equazione 9 viene definito l'**Absolute false** (o falso negativo) misura il rapporto tra gli eventi di previsione completamente errati rispetto agli eventi di previsione totali.

Tutti i 5 indicatori variano tra $[0; 1]$, i primi quattro devono essere massimizzati, mentre l'ultimo, rappresentando un errore, va minimizzato. Le seguenti formule sono definite seguendo la formulazione in [5]:

$$Aiming = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\parallel L_k \cap L_k^* \parallel}{\parallel L_k^* \parallel} \right) \tag{5}$$

$$Coverage = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\parallel L_k \cap L_k^* \parallel}{\parallel L_k \parallel} \right) \tag{6}$$

$$Accuracy = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\parallel L_k \cap L_k^* \parallel}{\parallel L_k \cup L_k^* \parallel} \right) \tag{7}$$

$$Absolute\, true = \frac{1}{N} \sum_{k=1}^{N} \Delta(L_k, L_k^*) \tag{8}$$

$$Absolute\, false = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\parallel L_k \cup L_k^* \parallel - \parallel L_k \cap L_k^* \parallel}{M} \right) \tag{9}$$

dove N indica il numero di campioni, M il numero di classi e $\Delta(\cdot, \cdot)$ è un operatore che restituisce 1 se i due insiemi hanno esattamente gli stessi elementi, 0 altrimenti.

4.3 Comparazione delle soluzioni

Tutti i classificatori utilizzano lo stesso estrattore di caratteristiche quindi la comparazione dipende prettamente dalle performance del classificatore. La comparazione è basata sugli indici proposti nella Sezione 4.2. Nella Tabella 4 vengono elencati tutti gli indici di prestazione relativi ai classificatori e sono evidenziati in grassetto i risultati migliori ottenuti. Dalla Tabella 4 si evince

Tabella 4: **Confronto di classificatori**

Classificatore	**Aiming**	**Accuracy**	**Coverage**	**Abs. True**	**Abs. False**
MLC	**84.88%**	70.38%	61.93%	65.90%	2.65%
MLkNN	81.19%	71.31%	70.03%	65.70%	2.47%
MatLearn	83.01%	**73.16%**	68.83%	**67.42%**	**2.36%**
LSTM ($\tau = 0.2$)	71.99%	72.40%	**71.44%**	63.10%	2.49%

che il classificatore MatLearn è il migliore sugli indici Accuracy, Absolute True e Absolute False e non si discosta di molto dai migliori in Aiming e Coverage; può essere considerato il migliore tra le soluzioni proposte. Tuttavia nessuna delle soluzione è significativamente inferiore alle altre dato che i risultati sono simili tra loro.

Riguardo la scelta della soglia migliore per l'uso di LSTM come classificatore, in Tabella 5 vengono riportati tutti gli indicatori di performance relativi alle varie soglie testate. All'aumentare della soglia migliora Aiming e Absolute True, tuttavia calano significativamente Accuracy e Coverage. Ciò indica che è presente un overfitting sui dati di training. Il miglior compromesso si ha per $\tau = 0.20$.

Tabella 5: **Confronto soglie di LSTM**

Soglia	**Aiming**	**Accuracy**	**Coverage**	**Abs. True**	**Abs. False**
$\tau = 0.10$	64.40%	72.26%	80.21%	59.70%	3.52%
$\tau = 0.15$	68.73%	72.64%	75.71%	61.65%	2.82%
$\tau = 0.20$	71.99%	72.40%	71.44%	63.10%	2.49%
$\tau = 0.25$	78.86%	71.30%	63.86%	65.52%	2.25%
$\tau = 0.30$	77.67%	71.88%	66.25%	65.00%	2.23%
$\tau = 0.35$	78.86%	71.30%	63.86%	65.52%	2.25%
$\tau = 0.40$	80.37%	70.38%	61.69%	65.46%	2.33%
$\tau = 0.45$	81.92%	69.38%	59.77%	64.98%	2.48%

4.4 Comparazione con lo stato dell'arte

Tramite gli stessi indicatori di prestazione è possibile confrontare il miglior risultato di questa tesi, l'ensemble LSTM + MatLearn (abbreviato in EnsLSTM_ML), con gli altri lavori sul campo sviluppati da vari esperti. La comparazione viene effettuata con i seguenti classificatori:

- Classificatore iATC-mISF [3]
- Ensemble LIFT (con parametro $\theta = 45$) [15]
- Classificatore ibrido iATC-mHyb [4]
- Ensemble AlexNet + LIFT + RR + DO (con parametro $\tau = 0.25$) [21][11][10][14]

Nella Tabella 6 sono riportati tutti gli indicatori di prestazione dei vari classificatori. Nonostante i buoni risultati ottenuti, la migliore soluzione finora rimane quella proposta in [14].

Tabella 6: Confronto con altri lavori

Nome	**Aiming**	**Accuracy**	**Coverage**	**Abs. True**	**Abs. False**
iATC-mISF	67.83%	67.10%	66.41%	60.98%	5.85%
EnsLIFT	78.19%	75.98%	71.31%	63.40%	2.80%
iATC-mHybe	71.91%	71.46%	71.32%	66.75%	2.43%
EnsANet_LR $\oplus$ DO	79.57%	83.35%	77.78%	70.90%	2.40%
EnsLSTM_ML	83.01%	73.16%	68.83%	67.42%	2.36%

5 Conclusioni

Lo scopo di questo lavoro è trovare un nuovo valido classificatore di composti farmaceutici per determinare a quali classi ATC di primo livello appartengono. Viene proposto un nuovo metodo di riformulazione del vettore descrittore del farmaco di input tramite una rete LSTM adibita all'estrazione di caratteristiche. Vengono proposti e confrontati vari tipi di classificatori, e dalle sperimentazioni effettuate emerge una nuova soluzione accettabile, un ensemble tra LSTM e matLearn. La soluzione viene poi comparata con lo stato dell'arte e si attesta sugli stessi risultati senza portare miglioramenti. Dalle analisi effettuate fra i classificatori proposti e i classificatori disponibili sul campo si nota che gli indicatori di performance hanno raggiunto un plateau. Ciò potrebbe essere dovuto al training dataset, troppo piccolo per poter ottenere risultati migliori. Se in un futuro verrà implementato il dataset, sarà possibile tramite ulteriori ricerche realizzare soluzioni migliori.

Il codice MatLab e il dataset utilizzati sono disponibili per future ricerche all'indirizzo `https://github.com/LorenzoBoccalon/tesi_classificazione_ATC`.

Bibliografia

[1] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[2] Lei Chen, Wei-Ming Zeng, Yu-Dong Cai, Kai-Yan Feng, and Kuo-Chen Chou. Predicting anatomical therapeutic chemical (atc) classification of drugs by integrating chemical-chemical interactions and similarities. *PloS one*, 7(4):e35254, 2012.

[3] Xiang Cheng, Shu-Guang Zhao, Xuan Xiao, and Kuo-Chen Chou. iatc-misf: a multi-label classifier for predicting the classes of anatomical therapeutic chemicals. *Bioinformatics*, 33(3):341–346, 2016.

[4] Xiang Cheng, Shu-Guang Zhao, Xuan Xiao, and Kuo-Chen Chou. iatc-mhyb: a hybrid multi-label classifier for predicting the classification of anatomical therapeutic chemicals. *Oncotarget*, 8(35):58494, 2017.

[5] Kuo-Chen Chou. Some remarks on predicting multi-label attributes in molecular biosystems. *Molecular Biosystems*, 9(6):1092–1100, 2013.

[6] M. Schmidt et al G. Roeder, X. She. matlearn: machine learning algorithm implementations in matlab. 2016. URL `https://www.cs.ubc.ca/~schmidtm/Software/matLearn.html`.

[7] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[9] T Jayalakshmi and A Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):1793–8201, 2011.

[10] Keigo Kimura, Lu Sun, and Mineichi Kudo. Mlc toolbox: A matlab/octave library for multi-label classification. *arXiv preprint arXiv:1704.02592*, 2017.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[13] Xiangang Li and Xihong Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4520–4524. IEEE, 2015.

[14] Alessandra Lumini and Loris Nanni. Convolutional neural networks for atc classification. *Current pharmaceutical design*, 24(34):4007–4012, 2018.

[15] Loris Nanni and Sheryl Brahnam. Multi-label classifier based on histogram of gradients for predicting the anatomical therapeutic chemical class/classes of a given compound. *Bioinformatics*, 33(18):2837–2841, 2017.

[16] Raymond C Pitts. Reconsidering the concept of behavioral mechanisms of drug action. *Journal of the experimental analysis of behavior*, 101(3): 422–441, 2014.

[17] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[18] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[19] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[20] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.

[21] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):107–120, 2015.

[22] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.