Alma Mater Studiorum · Università di Bologna
Campus di Cesena

SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

Individuazione di Nuove Funzioni Biologiche di Geni mediante Transfer Learning Inter-Organismo con Reti Neurali Autoencoder

Relatore:
Prof. Gianluca Moro

Presentata da:
Andrea Giannini

Seconda Sessione di Laurea
Anno Accademico 2017-2018

Sommario

La genomica è la disciplina che si occupa della struttura, sequenza, funzione ed evoluzione del genoma, cioè dell'informazione genetica contenuta nel DNA presente nelle cellule di una particolare specie. Avendo a disposizione enormi quantità di dati per l'uomo diventa difficile poterli analizzare tutti e riuscire fare deduzioni su di essi. Perciò fin dai primi anni di ricerca, in questa branca della biologia, si è resa fondamentale l'informatica per l'elaborazione e la visualizzazione dei dati che produce, tanto da poter affermare che si basa sulla bioinformatica. Con l'accrescere dei dati raccolti dalle ricerche, è nata la necessità di catalogarli in modo ordinato all'interno di enormi database. Sono molteplici i progetti avviati con questo scopo, uno dei principali è Gene Ontology che mantiene e sviluppa un vocabolario controllato, e annota e diffonde dati riguardanti i geni e le loro funzioni.

È qui che entra in gioco il machine learning: grazie ai dati raccolti fino ad ora sulle funzioni biologiche svolte dai prodotti dei geni, e ai metodi introdotti dal machine learning come gli algoritmi supervisionati e le reti neurali, è ormai possibile aiutare la ricerca direzionando gli scienziati nelle analisi e negli esperimenti da provare, utili per scoprire e confermare nuove correlazioni tra geni e funzioni biologiche. Inoltre, essendo tutti gli organismi legati dal punto di vista evolutivo, e quindi con un genoma più o meno simile, è possibile trasferire le conoscenze apprese da uno di essi ad un altro: questo è ciò che si intende per "Transfer Learning Inter-Organismo". In questa tesi verranno sfruttate le reti neurali per trasferire queste conoscenze, proponendo due metodologie differenti: una utilizza la struttura degli autoencoder, l'altra invece crea classificatori differenti per ogni valore da predire usando il deep learning.

Indice

Introduzione

Gli esperimenti per scoprire nuove funzioni biologiche in cui sono coinvolti geni e i loro prodotti sono molto costosi, richiedono una grande quantità di tempo e non sempre sono possibili. Sono necessari lunghi studi e molteplici tipi di esperimenti per dimostrare la veridicità delle ipotesi fatte. Inoltre con l'aumento dei dati disponibili in questo ambito, la ricerca è diventata sempre più dispersiva. Nasce proprio per questo motivo la necessità di catalogare in modo ordinato i dati e con questa, il progetto Gene Ontology. Questo progetto è nato infatti con l'intento di unificare la descrizione delle caratteristiche funzionali dei prodotti dei geni di tutte le specie, quindi di mantenere e sviluppare un vocabolario controllato e di annotare e diffondere dati riguardanti i geni e le loro funzioni.

Da questa catalogazione è possibile notare che le funzioni dei geni sono strettamente legate tra loro: ad esempio, se due geni svolgono entrambi una certa funzione, è probabile che ne abbiano altre in comune. Questa similarità diventa molto utile quando uno dei due geni è stato studiato affondo e l'altro è relativamente sconosciuto. Infatti, in questi casi, i ricercatori possono usare il gene ben studiato come guida per gli esperimenti da svolgere su quello sconosciuto, aumentando così l'efficacia della ricerca. Ma le annotazioni (funzioni dei geni) sono migliaia come anche i geni da analizzare, per cui trovare delle correlazioni all'interno non è così semplice. Ed è proprio questo uno dei casi in cui l'informatica può rappresentare un valido aiuto, grazie al quale è possibile trovare correlazioni in modo più agevole e sfruttarle in molteplici modi. Inoltre siccome tutti gli esseri viventi hanno un genoma molto simile

fra loro, è possibile anche usare geni di organismi diversi da quello che si sta analizzando. Questo metodo viene chiamato "Transfer Learning Inter-Organismo" ed è efficace soprattutto nei casi in cui c'è una grande differenza di conoscenza tra i due organismi.

In un articolo scritto da Giacomo Domeniconi e pubblicato da Elsevier, editore mondiale in ambito medico e scientifico, viene proprio illustrato l'utilizzo di algoritmi supervisionati sui dati di Gene Onthology nel prevedere annotazioni e la loro probabilità di essere confermate. I risultati di quest'ultimo verranno presi come punti di riferimento.

La tesi si suddividerà in queste parti:

- **Genomica**: qui verrà illustrata un'introduzione alla genomica dove si darà una definizione a tale termine, si andrà ad analizzare la sua nascita e ciò di cui si occupa. Inoltre introdurrò Gene Ontology;
- **Analisi dei dati**: questo capitolo si occuperà di illustrare la natura dei dati di cui siamo in possesso e da cui partiranno le ricerche;
- **Le Reti Neurali**: ci sarà un'introduzione alle reti neurali e la spiegazione del loro funzionamento;
- **Progettazione**: verrà fatta l'analisi del "problema" e verranno identificate e spiegate le possibili strade da intraprendere per poterlo risolvere;
- **Implementazione**: questo capitolo mostrerà come è stato svolto il progetto, spiegando e motivando i vari passaggi;
- **Risultati**: verranno esposti tutti i risultati ottenuti rendendoli più chiari possibile facendo uso di tabelle e grafici;
- **Conclusione**

Capitolo 1

Genomica

1.1 Definizione di genomica

Negli anni Settanta del secolo scorso si è sviluppato un filone di ricerca finalizzato a sequenziare il DNA. Grazie al sequenziamento del DNA, cioè a quella tecnica che permette di stabilire la sequenza delle basi azotate presenti in una molecola di DNA, è nata la possibilità di studiare i genomi. Ciò segna l'inizio di quella branca della biologia molecolare che definiamo genomica e che si occupa della struttura, sequenza, funzione ed evoluzione del genoma, cioè di tutta l'informazione genetica contenuta nel DNA (DeoxyriboNucleic Acid) presente nelle cellule di una particolare specie.

Sottoaree della genomica

La genomica comprende principalmente tre sottoaree:

- **Genomica strutturale** si occupa della costruzione di dati di sequenze genomiche, la scoperta di geni, la localizzazione e costruzione di mappe genetiche.

- **Genomica funzionale** mira a comprendere le funzioni biologiche dei geni, la loro regolazione e i loro prodotti.

- **Genomica comparativa** confronta le sequenze dei geni e delle proteine di diversi genomi per mettere in evidenza relazioni funzionali e evolutive. Visto il suo stretto legame con la genomica funzionale spesso viene considerata un suo settore.

1.2 Genomica Strutturale

Sequenziamento del DNA

Il primo passo della genomica è il sequenziamento del DNA, cioè la determinazione dell'ordine delle basi azotate. Questo non corrisponde alla «decifrazione» del DNA, che invece avverrà solo nei passi successivi, all'interno della genomica funzionale.

Anche se adesso il sequenziamento è diventato uno dei passaggi più veloci all'interno della genomica grazie all'introduzione di sistemi automatici, all'inizio era un processo che richiedeva anni di ricerche e la collaborazione di più laboratori. Infatti i primi genomi a essere sequenziati e studiati sono stati quelli di alcuni virus e batteri che, per le loro dimensioni contenute, erano più facili da trattare per i ricercatori. Fin da subito, tuttavia, fu chiaro che l'obiettivo più interessante era giungere a sequenziare il genoma umano; questa impresa prometteva infatti le più importanti ricadute pratiche, soprattutto nel campo della medicina. Fu così che ebbe inizio il progetto di sequenziare il genoma umano, denominato Progetto Genoma Umano (HGP, Human Genome Project) e approvato nel 1989 dal Congresso americano.

Progetto Genoma Umano

Il Progetto Genoma Umano è stato uno dei più grandi progetti di ricerca in campo biologico dell'intero Novecento. Avviato nel 1991 su iniziativa di James Watson con una durata stimata intorno ai 15 anni, il progetto è stato completato ufficialmente nel 2003, anche se una prima bozza era stata pubblicata nel 2001. Sono riusciti nell'impresa di concluderlo con 3 anni di

anticipo grazie all'avanzamento delle tecnologie e alla competizione che si sviluppò con un'impresa privata, la Celera genomics.

Gli obiettivi iniziali dell'HGP furono i seguenti:

- Determinare la sequenza dei 3 miliardi di bp (Base Pair, coppie di basi azotate) che costituiscono il DNA umano
- Identificarne i geni
- Conservare le informazioni ottenute in banche dati pubbliche
- Sviluppare strumenti informatici per l'analisi dei dati
- Trasferire le tecnologie derivate al settore pubblico
- Affrontare gli aspetti di ordine etico, legale e sociale che sarebbero sorti con la realizzazione del progetto

Oltre al genoma umano, l'HGP finanziò il sequenziamento dei genomi di altri organismi, comunemente usati come modello negli studi genetici, al fine di rendere possibile lo studio comparativo dei genomi: il batterio Escherichia coli, il lievito Saccharomyces cerevisiae, il moscerino della frutta Drosophila melanogaster, il nematode Caenorhabditis elegans e il topo Mus musculus.

Organismo	bp stimati	Numero cromosomi	Numero di geni stimato
Escherichia Coli	4.6 milioni	1	3,200
Saccharomyces cerevisiae	12 milioni	32	6,000
Drosophila melanogaster	165 milioni	8	13,000
Caenorhabditis elegans	97 milioni	12	19,000
Mus Musculus	2.9 miliardi	40	∼25,000
Homo Sapiens	3 miliardi	46	∼25,000

Tabella 1.1: Comparazione grandezza del genoma degli organismi del HGP

1.3 Genomica Funzionale

Come detto in precedenza il sequenziamento del DNA non consiste in una "decifrazione", quindi per conoscere un gene non è sufficiente identificarne la sequenza: bisogna anche determinarne l'esatta funzione e il modo in cui interagisce con altri geni, ovvero è necessario attribuirgli una connotazione funzionale (o annotazioni). La genomica funzionale tratta proprio tale argomento e quindi consiste nell'assegnazione dei ruoli funzionali ai prodotti dei geni individuati.

Con il sequenziamento di un intero genoma si trovano, infatti, molte sequenze geniche sconosciute e non associabili ad alcuna funzione. Spesso tale funzione può essere ricavata da ricerche successive, comparando, per esempio, la struttura del gene scoperto a quella di altri geni.

Per tali ricerche si possono usare due tipi di approcci:

- **Approccio riduzionista**, lo studio dei geni e dei loro prodotti, uno per volta
- **Approccio olistico**, in cui molti o addirittura tutti i prodotti genici sono studiati simultaneamente

Attraverso questi, la genomica funzionale mira a comprendere le modalità con le quali i geni che compongono il nostro genoma dirigono lo sviluppo e il funzionamento del nostro organismo e come il loro malfunzionamento induca uno stato patologico.

Contributo della Bioinformatica

La genomica funzionale, nata dal contributo di diverse discipline, si basa sulla bioinformatica per l'analisi computazionale della grande quantità di dati prodotta dai laboratori. Il principale scopo degli approcci computazionali è innanzitutto quello di fornire le corrette annotazioni funzionali di una data sequenza genica. Tali annotazioni prevedono l'identificazione di geni potenzialmente codificanti una o più proteine. Questo processo viene effettuato

grazie all'ausilio di algoritmi di ricerca in grado di identificare le cosiddette ORF (Open Reading Frame), cioè le sequenze di codoni che codificano le proteine. In genere l'identificazione di una sequenza ORF richiede il confronto con le informazioni presenti all'interno di banche dati alle quali è possibile accedere attraverso Internet.

I programmi preposti a tali confronti si basano su algoritmi di allineamento di sequenze e implicano la comparazione della sequenza di interesse con quelle presenti all'interno dei database, di cui si conosce la funzione. Fra i vari algoritmi di allineamento spicca in modo particolare il programma BLAST (Basic Local Alignment Search Tool), prodotto dal National center for biotechnology information (NCBI). I problemi di annotazione di una determinata sequenza possono nascere laddove nessun gene simile a quello che stiamo studiando sia stato ancora caratterizzato. Questi geni vengono spesso definiti geni FUN (Function UNknown), meglio conosciuti come geni orfani. Tuttavia, le annotazioni di un genoma non si limitano solo all'identificazione di sequenze ORF: esistono sequenze regolatrici la cui funzione è quella di riunire i vari esoni e introni di un gene. Per l'identificazione e l'annotazione di queste particolari sequenze si richiede l'ausilio di algoritmi più sofisticati.

Quindi da una sequenza genomica è possibile ricavare principalmente tre tipologie di informazioni:

- Le finestre di lettura (ORF, Open Reading Frames), ossia i tratti codificanti dei geni. Per i geni che codificano proteine, queste regioni sono riconoscibili in base ai codoni di inizio e di stop della traduzione.

- Le sequenze amminoacidiche delle proteine, deducibili dalle corrispondenti sequenze di DNA delle finestre di lettura, applicando le regole del codice genetico.

- Le sequenze regolatrici, come ad esempio i promotori e i terminatori della trascrizione.

Per superare alcuni dei limiti sino a qui descritti, le annotazioni vengono spesso eseguite sulla base della cosiddetta genomica comparativa.

1.4 Genomica Comparativa

La genomica comparativa è quella branca della genomica in grado di confrontare interi genomi di diverse specie con lo scopo di aumentare le conoscenze sulle funzioni dei singoli geni e quindi dell'intero genoma. L'approccio comparativo si basa sull'assunto che tutti gli attuali genomi si siano evoluti partendo da genomi ancestrali. In questo modo, la conoscenza di un gene in un dato organismo può fornire informazioni sul gene omologo di un altro organismo. L'uso di informazioni su altri genomi, diversi da quello umano, è in grado di rimediare alla mancanza di conoscenza spesso causata da problemi etici inerenti, talora, l'impossibilità di una diretta sperimentazione sui geni umani.

1.5 Altri Progetti

Con il passare degli anni i costi e i tempi di sequenziamento del genoma sono diminuiti sempre di più, permettendo così la creazione di molti progetti portati avanti anche da singoli laboratori e di velocizzare l'avanzamento della ricerca nella genomica.

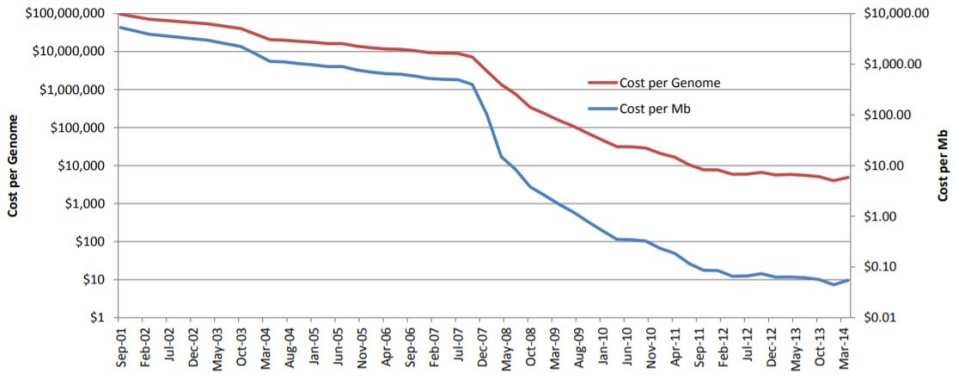


Figura 1.1: Grafico dei costi di sequenziamento per genoma e per Mb

International HapMap project

Il progetto internazionale HapMap è stato avviato nell'ottobre del 2002 grazie a un consorzio di scienziati di sei nazioni con lo scopo di ottenere una mappa del genoma umano e identificare i geni correlati a malattie come l'asma, il cancro, il diabete e le patologie cardiologiche. Inoltre, esso dovrebbe permettere lo studio dei fattori genetici che contribuiscono alla variazione individuale ai rischi ambientali, alla suscettibilità alle infezioni e all'efficacia di farmaci e vaccini.

ENCODE

Il progetto ENCODE (ENCyclopedia Of Dna Elements) ha avuto come finalità la creazione di un'enciclopedia degli elementi funzionali codificati nel DNA, e quindi di identificare e di localizzare con precisione tutti i geni codificanti o meno e tutti gli altri elementi funzionali contenuti nella sequenza del DNA (The ENCODE project consortium, 2007). Questo catalogo monumentale può essere utile agli scienziati per scrutare e utilizzare in maniera più efficace la sequenza del genoma umano, al fine di sviluppare nuove strategie per la prevenzione e il trattamento delle malattie. Anche se solo l'1% codifica per proteine il progetto ha messo in evidenza che più dell'80% del genoma è funzionalmente "attivo".

Questi sono due dei tanti progetti avviati negli anni in questo ambito, ma per questa ricerca si useranno i dati di Gene Ontology.

1.6 Gene Ontology

Uno dei più importanti progressi della genomica si è avuto nella descrizione dell'ontologia dei geni. La mancanza di una terminologia standard universale nel campo della biologia e domini correlati, rendeva la comunicazione e la condivisione dei dati sempre più difficile con l'aumentare della grandezza dei

database. Ciò ha fatto nascere la necessità di creare una standardizzazione e quindi l'avviamento dell'iniziativa Gene Ontology.

Gene Ontology (GO) ha permesso di sviluppare un sistema di nomenclatura comune per definire un concetto di funzionalità genica applicabile a tutti gli organismi (Ashburner, Ball, Blake et al. 2000). Il progetto è nato nel 1998 dalla collaborazione tra gli archivi elettronici dedicati alla conservazione dei dati genomici di vari organismi, con lo scopo di sviluppare una descrizione coerente dei geni e dei loro prodotti fra i diversi archivi. In particolare, il progetto si propone di:

- Mantenere e sviluppare un vocabolario controllato atto a descrivere i geni e i prodotti genici per ogni organismo vivente;
- Annotare i geni e i prodotti genici, e diffondere tali dati;
- Fornire strumenti per un facile accesso ai dati forniti dal progetto.

Inizialmente, al progetto Gene ontology hanno partecipato gli archivi dedicati di tre importanti organismi modello per la ricerca biologica: il moscerino della frutta Drosophila (Flybase, `http://flybase.bio.indiana.edu/`), il lievito (Saccharomyces genome database, `http://www.yeastgenome.org/`) e il topo (Mouse genome database, `http://www.informatics.jax.org/`). In seguito, al consorzio GO hanno aderito tutti i maggiori centri per la conservazione e la divulgazione di dati genomici.

Il progetto ha sviluppato tre vocabolari strutturati e controllati (ontologie), organizzati gerarchicamente in una struttura a grafo orientato aciclico dove ogni nodo è un termine GO (GO term):

- Processi Biologici (BP, Biological process): le operazioni o complessi di eventi molecolari con un inizio e una fine definiti, pertinenti al funzionamento di unità viventi integrate: cellule, tessuti, organi e organismi;
- Componenti Cellulari (CC, Cellular component): le parti di una cellula o del suo ambiente extracellulare;

- Funzioni Molecolari (MF, Molecular function): le attività elementari di un prodotto genico a livello molecolare, come legante o catalisi.

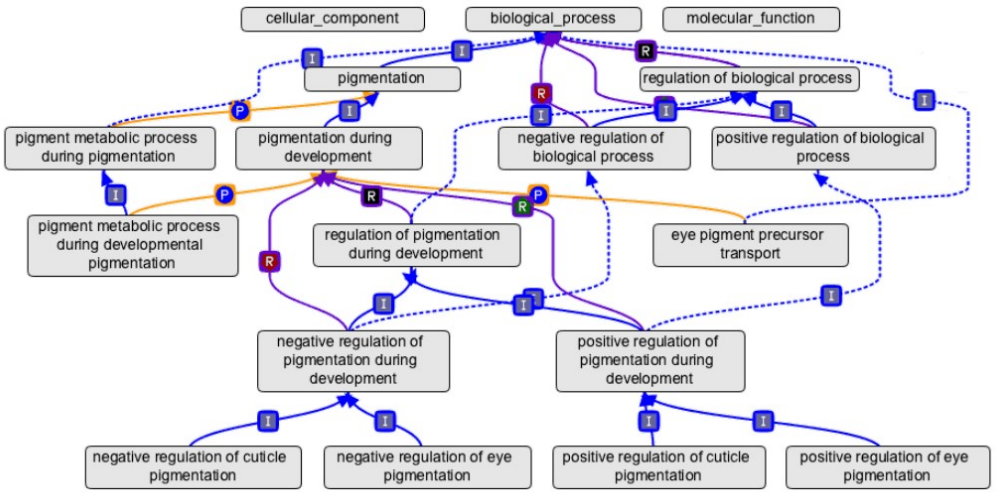


Figura 1.2: Sezione del grafo di Gene Ontology

Come si può vedere nel grafo dell'immagine 1.2 ci sono differenti tipologie di legami tra i termini:

- **is a (I)**, quando un "termine A" è un "termine B"
- **part of (P)**, quando un "termine A" fa parte del "termine B"
- **has part (hP)**, il complementare di "part of"
- **regulates (R)**, quando "termine A" regola "termine B"

Un prodotto genico può essere associato o localizzato in uno o più componenti cellulari, essere attivo in uno o più processi biologici, nel corso dei quali esegue una o più funzioni molecolari. Per esempio, il prodotto del gene MYC è associato per funzioni molecolari alle locuzioni 'fattore di trascrizione' e

'interazione con proteine'; per processo biologico alle locuzioni 'regolazione positiva della proliferazione cellulare' e 'regolazione della trascrizione dai promotori della RNA polimerasi II' e, infine, per componente cellulare al termine 'nucleo'. Inoltre, dato che i vocabolari sono strutturati gerarchicamente, l'annotazione di un gene o di un prodotto genico può essere specificata a differenti livelli, in funzione dell'accuratezza e dell'approfondimento delle conoscenze che di esso sono disponibili.

Il database del consorzio ospita sia le tre ontologie sia le annotazioni dei geni e dei loro prodotti fatte in accordo alla terminologia GO. Le annotazioni sono in pratica delle associazioni che si creano tra i geni e le funzioni biologiche. In tal modo è possibile compiere ricerche complesse sulle annotazioni usando le ontologie. Utilizzando una interfaccia apposita, AmiGO[7], è possibile sfogliare e interrogare l'archivio. La diffusione dell'uso dei termini GO da parte degli archivi genomici che partecipano al progetto permette una maggiore uniformità del linguaggio di interrogazione facilitando le ricerche trasversali. Data la struttura ad albero dei vocabolari GO, le interrogazioni possono raggiungere differenti livelli; per esempi, è possibile ricercare tutti i prodotti genici del topo implicati nella trasduzione del segnale.

All'interno del database di Gene Ontology, ogni annotazione, oltre ad avere informazioni riguardo al prodotto del gene e al suo termine GO, possiede i seguenti dati:

- Reference, la fonte utilizzata (ad esempio, un articolo di una rivista scientifica)
- Evidence code denota il tipo di prova su cui si basa
- La data e il creatore

Capitolo 2

Analisi dei dati

2.1 Scelta dei dati

Online ormai è possibile trovare moltissimi dati riguardanti i genomi e gli esseri viventi analizzati sono sempre più numerosi. Per cui sorge naturale una domanda: da che database prendere i dati?

Il principale problema con cui ci si scontra osservando i database online è la non standardizzazione dei dati: ogni laboratorio pubblica i dati ottenuti a modo suo. Proprio per questo motivo si è deciso di usare i dati di Gene Ontology. Come già spiegato Gene Ontology non è il nome di un laboratorio di ricerca che pubblica i propri dati, ma di un progetto che propone uno standard a cui hanno aderito vari laboratori e deciso di pubblicare i propri dati seguendo il "vocabolario" creato da loro.

Altra questione su cui prendere una decisione è la scelta degli organismi da esaminare. Per il tipo di ricerca che dobbiamo fare sono stati scelti degli esseri viventi sui quali si ha un livello di conoscenza differente. Inoltre la scelta è stata condizionata da un articolo di due anni fa con cui vogliamo confrontare i dati trovati. Per l'applicazione del machine learning e per migliorare le capacità predittive, si usano di solito dati presi in momenti temporali differenti. Per queste ragioni sono stati scelti i genomi degli stessi anni dell'articolo, cioè del 2009 e del 2013.

I genomi scelti quindi sono quelli di questi organismi:

- Bos Taurus (Bovino)
- Mus Musculus (Topo comune)
- Rattus Norvegicus (Ratto di fogna)
- Homo Sapiens (Uomo)

2.2 Primo sguardo

Come vengono rappresentate le annotazioni del genoma di un organismo? Come viene mostrato nella figura 2.1, vengono rappresentate facendo uso di enormi tabelle, dove ogni riga rappresenta un gene diverso con il proprio identificativo, mentre ogni colonna rappresenta un termine GO (GO Term), anche questo con il suo codice.

gene	GO:0019538	GO:0003824	GO:0009987	GO:0042221	GO:0010556	GO:0008233	GO:0016197	GO:0016638	GO:0031226	GO:0006886	...
DUSP10	1	0	1	0	0	0	0	0	0	0	...
PPP1R3D	0	1	0	0	0	0	0	0	0	0	...
DPH3	1	0	1	0	0	0	0	0	0	0	...
GPR81	0	0	0	1	0	0	0	0	0	0	...
CSDA	0	0	0	0	1	0	0	0	0	0	...
CFD	1	1	0	0	0	1	0	0	0	0	...
VPS4B	0	1	1	1	0	0	1	0	0	0	...
LOX	1	1	1	0	0	0	0	1	0	0	...
PRRG2	0	0	0	0	0	0	0	0	1	0	...
BCL3	0	0	1	0	1	0	0	0	0	1	...
WARS	1	0	1	0	0	0	0	0	0	0	...
BICD1	0	0	1	0	0	0	0	0	0	0	...
TAF11	0	0	1	0	1	0	0	0	0	0	...
GPR15	0	0	1	0	0	0	0	0	1	0	...
C9orf89	0	0	0	0	0	0	0	0	0	0	...
PGAM1	0	1	0	0	0	0	0	0	0	0	...
RHOJ	0	1	1	0	0	0	0	0	0	0	...
ERBB2IP	0	0	1	0	0	0	0	0	0	0	...
JAG2	0	0	1	0	0	0	0	0	1	0	...
PLG	0	1	0	0	0	1	0	0	0	0	...

Figura 2.1: Sezione della tabella dell'Homo Sapiens

Le annotazioni vengono rappresentate con degli "1" all'interno della tabella. Ciò significa che, se è presente un "1" all'incrocio tra la riga del gene A e la colonna del termine GO:xxxx, il prodotto del gene A svolge la funzione descritta dal termine GO:xxxx.

Per ogni essere vivente la tabella può assumere delle dimensioni differenti, a seconda della quantità di geni codificanti presenti all'interno del DNA dell'organismo e delle funzioni che svolgono i loro prodotti. Quindi più geni si scoprono più righe si aggiungono, mentre se vengono scoperte delle nuove funzioni di prodotti genici che prima non erano presenti nella tabella, questa si allarga perché vengono aggiunti nuovi termini GO e quindi nuove colonne. All'interno della tabella sono presenti solo le colonne strettamente necessarie, cioè che possiedono almeno un'annotazione. Quindi finché una funzione ancora non si sa se viene svolta all'interno di quel determinato organismo o se, pur essendo svolta, non si sa da che gene deriva il prodotto che fa tale funzione, non appare la sua colonna nella tabella. Il numero di colonne volendo potrebbe essere fisso per tutte le tabelle inserendo tutti i termini GO di cui siamo a conoscenza, ma le tabelle diventerebbero ancora più grandi e occuperebbero troppo spazio inutile.

Trattandosi di tabelle possono essere salvate in csv, ma siccome le annotazioni di ogni gene sono sparse e molto meno delle colonne presenti, sono spesso salvate in arff(Attribute Relationship File Format), formato comodo per salvare tabelle con dati sparsi come in questo caso in quanto salva solo le celle con dei valori diversi da zero memorizzandone solo la posizione.

2.3 Analisi

La prima cosa che salta all'occhio come già detto sono le diverse dimensioni delle tabelle. Essendo tutti mammiferi e quindi aventi più o meno un genoma abbastanza simile, si può capire già da qui quali organismi sono stati studiati in modo approfondito e quali meno. Ad esempio, dalla tabella 2.1 e dal grafico 2.2 si può notare che tra gli organismi presi in esame quello meno studiato è il Bos Taurus che all'interno della sua tabella ha molti meno geni degli altri. Ovviamente vale la stessa cosa anche per la quantità di annotazioni, infatti nel grafico 2.2 si può notare proprio questa enorme differenza.

Altra cosa interessante da notare è l'aumento dei dati dal 2009 al 2013 di tutti gli organismi: in solo 4 anni il numero di termini GO all'interno delle tabelle (quindi il numero di colonne) è più o meno raddoppiato in tutti i casi, portandosi dietro con sé anche la quantità di annotazioni, anche esse raddoppiate, se non di più. L'aumento delle annotazioni è dovuto anche all'aumento dei geni che comunque è aumentato in modo meno evidente, tranne nel caso del Bos Taurus, sul quale probabilmente c'erano ancora tanti geni da analizzare.

Genoma	Num. Geni	Num. GO Terms	Num. Annot.	Annot. per gene
Bos Taurus 2009	735	792	24,208	32.94
Bos Taurus 2013	2,242	2,285	112,945	50.38
Mus Musculus 2009	9,818	3,469	342,903	34.93
Mus Musculus 2013	14,503	7,457	985,047	67.92
Rattus Norvegicus 2009	11,018	4,540	490,905	44.55
Rattus Norvegicus 2013	12,078	7,351	892,898	73.93
Homo Sapiens 2009	10,773	3,506	375,674	34.87
Homo Sapiens 2013	13,426	6,345	872,998	65.02

Tabella 2.1: Dati sulle dimensioni dei genomi

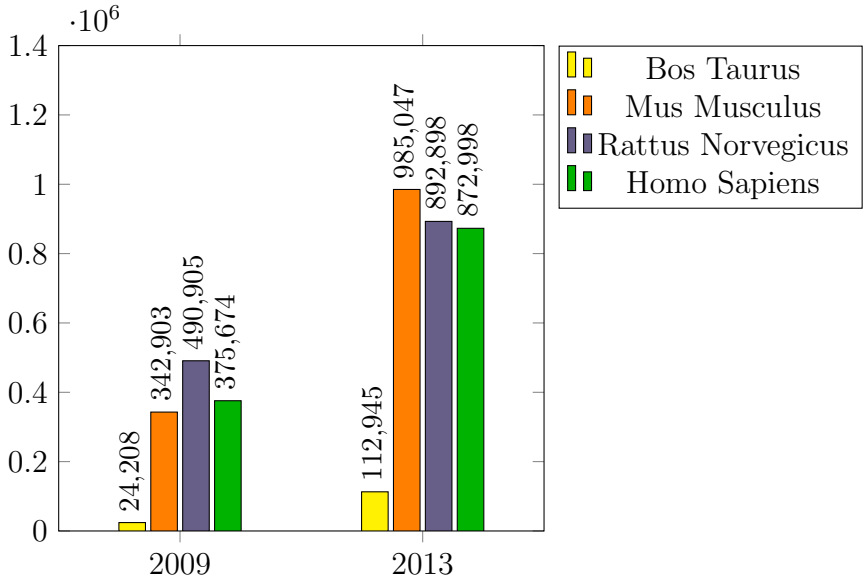


Figura 2.2: Istogramma sulla quantità di annotazioni

Confrontando il numero di annotazioni con quello dei geni, appare evidente che ogni gene può possedere molteplici annotazioni. Questo è dato dal fatto che un gene, essendo l'unione di sequenze genomiche, può codificare molti prodotti funzionali diversi che ovviamente possono avere funzioni completamente differenti.

Altro fattore che va ad aumentare il numero di annotazioni è la struttura del vocabolario di Gene Ontology, perché avendo molti termini GO che sono generalizzazioni di termini più specifici (legame "is a") è ovvio che ci sarà anche l'annotazione nei termini meno specifici. Ad esempio nel gene DUSP10 del genoma del 2009 dell'Homo Sapiens, essendoci l'annotazione nel termine GO:0019538 che corrisponde a "protein metabolic process", vuol dire che avrà sicuramente le annotazioni anche nelle sue generalizzazioni, come GO:0008150 "biological process", GO:0008152 "metabolic process", GO:0006807 "nitrogen compound metabolic" e altri ancora, visibili nella figura 2.3.

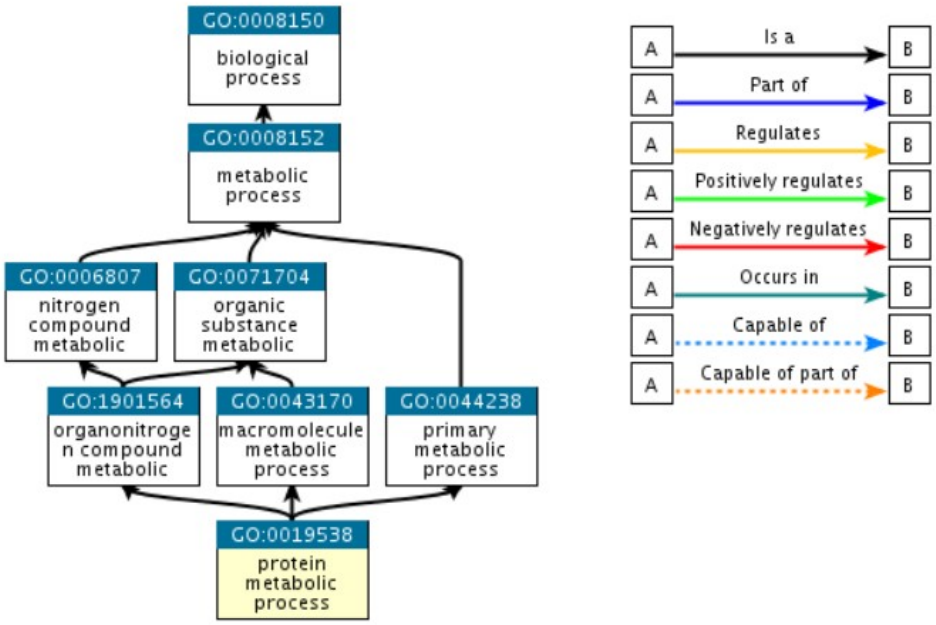


Figura 2.3: Grafo Gene Onthology a cui appartiene il termine GO:0019538

Capitolo 3

Le reti neurali

3.1 Definizione

Le reti neurali artificiali sono modelli matematici ispirati dalla semplificazione delle reti neurali biologiche. Perciò proprio come queste ultime, le reti neurali artificiali sono costituite da un insieme di unità elementari collegate tra loro da connessioni unidirezionali; le unità emulano i neuroni, mentre le connessioni emulano le giunzioni sinaptiche tra i neuroni. Dato il numero elevato di unità e delle loro interconnessioni da cui sono composti questi modelli, si viene a formare una fitta rete ed è da qui che deriva il nome.

Le reti neurali sono composte da gruppi di neuroni artificiali organizzati in livelli: un livello di input, un livello di output, e uno o più livelli intermedi o nascosti (hidden). Durante il funzionamento della rete, l'attivazione si propaga dalle unità di input a quelle di output passando attraverso le unità dei livelli intermedi come mostrato nell'immagine 3.1.

Se immaginiamo che una rete neurale controlli il comportamento di un semplice organismo posto nell'ambiente, le unità di input della rete emulano gli organi sensoriali dell'organismo e le unità di output gli organi motori.

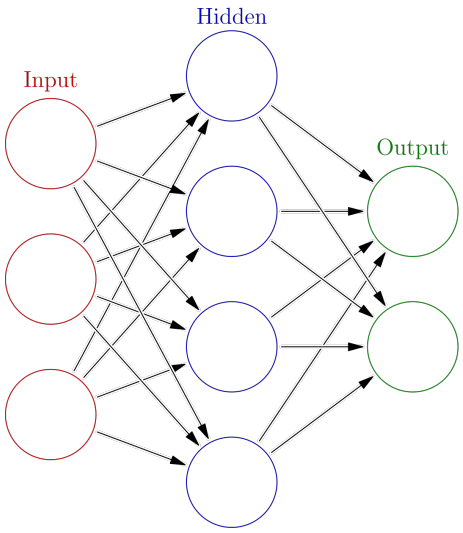


Figura 3.1: Struttura base di una rete neurale

In ogni "ciclo" le unità della rete hanno un livello di attivazione che corrisponde alla frequenza di emissione di impulsi nervosi da parte di un neurone. Nel caso delle unità di input, il livello d'attivazione dipende dall'ambiente esterno, mentre per quanto riguarda le unità interne e quelle di output, dipende dalle eccitazioni o inibizioni che giungono a un'unità dalle altre unità con cui essa è collegata. Una eccitazione tende ad aumentare il livello di attivazione dell'unità, mentre una inibizione tende a diminuirlo. Le connessioni, oltre a poter essere eccitatorie o inibitorie, possono variare nel loro peso, cioè nel contributo che esse danno alla determinazione del livello di attivazione delle unità.

Il peso emula il numero di recettori sinaptici situati sulla membrana esterna del neurone postsinaptico ed è espresso con un numero positivo nel caso in cui la connessione sia eccitatoria, negativo nel caso sia inibitoria. Una unità invia a un'altra unità con cui è collegata una quantità di eccitazione o di inibizione che dipende dal proprio livello di attivazione e dal peso della connessione che collega le due unità.

Le reti neurali sono modelli capaci di riprodurre, rendere più comprensi-

bili e talvolta predire aspetti importanti del comportamento e dell'apprendimento. Esse sono in grado di cogliere le regolarità sottostanti a una varietà di compiti elementari di categorizzazione, riconoscimento di pattern, associazione e memoria, previsione, mappatura sensomotoria, e così via.

Inoltre, le reti neurali dimostrano di avere una serie di proprietà simili a quelle che si osservano negli organismi reali:

- rispondono in modo appropriato a input deteriorati;
- ricostruiscono la parte mancante di input parziali;
- sviluppano forme di rappresentazione interna simili a quelle dei sistemi nervosi reali;
- quando il ricercatore danneggia la rete eliminando unità o connessioni, esibiscono deficit comportamentali simili a quelli di sistemi nervosi che hanno subito lesioni;
- riflettono il modo in cui il comportamento cambia nel corso dell'apprendimento, inclusi i differenti effetti provocati da danneggiamenti apportati in stadi diversi dell'apprendimento.

La spiegazione delle reti neurali può essere semplice facendo riferimento per ogni cosa alle reti neurali biologiche come fatto finora, senza dimenticare però che alla base di quelle artificiali c'è un modello matematico costruito prendendo spunto proprio da questi concetti.

3.2 Modello matematico

Come già detto, durante il funzionamento della rete, l'impulso si propaga dalle unità di input a quelle di output passando attraverso le unità dei livelli intermedi. L'elaborazione del segnale avviene proprio all'interno di queste unità elementari di cui sono composte.

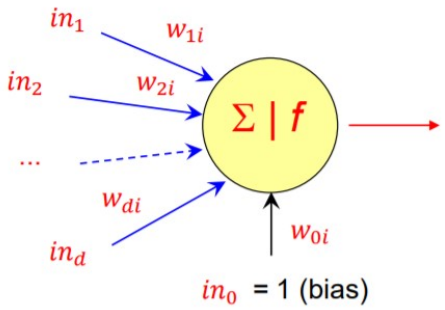


Figura 3.2: Rappresentazione grafica del percettrone

I "neuroni" della rete neurale artificiale sono chiamati percettroni (dall'inglese perceptron) e consistono in pratica in classificatori che mappano i dati ricevuti in input in un valore di output (scalare di tipo reale) e lo trasmette ai percettroni successivi.

La mappatura avviene tramite questa funzione:

$$net_i = \sum_{j=1...d} w_{ji} \cdot in_j + w_{0i}$$

$$out_i = f(net_i)$$

Dove:

- in_1, in_2,... in_d sono i d input che il percettrone i riceve dai percettroni afferenti
- w_{1i}, w_{2i},... w_{di} sono i pesi (weight) che determinano l'influenza che avrà ogni valore in input sul valore di output;
- w_{0i} è un ulteriore peso, chiamato solitamente bias, che si considera collegato a un input fittizio con valore sempre 1; questo peso è utile per "tarare" il punto di lavoro ottimale del percettrone;
- net_i è il livello di "eccitazione" globale del percettrone;

- f è la funzione di attivazione che determina il comportamento del percettrone (ovvero il suo output) in funzione del suo net_i.

Semplificando quindi, i segnali che il percettrone riceve come input vengono moltiplicati per i rispettivi pesi e sommati fra loro, compreso il bias. Infine per ottenere il segnale da inviare ai percettroni successivi al risultato viene applicata la funzione di attivazione.

3.3 Addestramento

Il modo in cui una rete neurale risponde all'input dipende fondamentalmente da due fattori: dalla sua architettura, cioè dallo schema delle interconnessioni, e dal peso di ciascuna connessione. Data una certa architettura e determinati pesi, la rete risponderà a uno stesso input generando sempre il medesimo output.

Fissata l'architettura della rete neurale, il suo addestramento consiste proprio nel determinare il valore dei pesi w che determinano la mappatura desiderata tra input e output, che ovviamente dipenderà dal tipo di problema che vogliamo risolvere:

- se siamo interessati ad addestrare una rete neurale che operi come classificatore, l'output desiderato è l'etichetta corretta della classe dell'input;

- se la rete neurale deve risolvere un problema di regressione, l'output desiderato è il valore corretto della variabile dipendente, in corrispondenza del valore della variabile indipendente fornita in input.

Come in un normale organismo, l'addestramento della rete neurale artificiale, e quindi la modifica dei pesi, è frutto delle "esperienza" fatta da essa. Infatti per questo processo non bisogna fare altro che darle "in pasto" dei dati di input da cui apprendere conoscenza. L'insieme dei dati usati per addestrare la rete viene chiamato training set. Per fare in modo che

nella previsione la rete sbagli meno possibile rispetto ai risultati desiderati vengono usati algoritmi che vanno ad agire su tutti i pesi della rete, che inizialmente hanno valori casuali, rendendoli adatti per lavorare sul training set. L'algoritmo maggiormente usato è la retropropagazione dell'errore (error backpropagation).

Ci sono tre modalità di apprendimento che si differenziano per la presenza o meno dei valori di output desiderati.

3.3.1 Apprendimento supervisionato

In questo caso il training set che viene presentato alla rete neurale contiene sia i dati d'input che l'output desiderato chiamato anche etichetta (label); è per questo che si dice che la rete apprende in presenza di un "maestro".

Usando l'algoritmo di error backpropogation, quando la rete genera un output in risposta a un dato input, questo viene confrontato con l'etichetta e viene calcolata la discrepanza tra i due (l'errore) tramite una loss function scelta in precedenza. Poi l'errore viene usato dall'algoritmo per modificare i pesi e adattare la rete all'output richiesto.

Inoltre, l'algoritmo stima l'errore delle unità interne in modo da poter modificare pure i pesi delle connessioni che hanno determinato l'output delle unità interne. Così si attua una retropropagazione dell'errore partendo dalle unità di output fino a raggiungere le unità di input della rete. Le modifiche dei pesi avvengono in modo tale che, attraverso un certo numero di cicli di apprendimento (detti 'epoche'), l'output, generato dalla rete in risposta a ciascun input, viene gradualmente approssimato all'output desiderato. Dopo un po' di epoche, l'errore tenderà a zero e la rete avrà acquisito la capacità desiderata.

Per testare l'accuratezza della rete la si prova su dati di cui si sa il risultato, ma che non sono stati usati per l'addestramento. Bisogna fare attenzione a questo dato perché, nei casi in cui l'apprendimento è stato effettuato troppo a lungo, o dove c'era uno scarso numero di esempi di allenamento, il modello potrebbe adattarsi a caratteristiche che sono specifiche solo del training set,

ma che non hanno riscontro nel resto dei casi e quindi di ricadere in ciò che viene chiamato overfitting. Come mostrato nella figura 3.3, in questi casi andando avanti con le epoche, l'accuratezza delle previsioni sui dati del Training Set aumenterà, mentre quella delle previsioni sui dati del Validation Set diminuirà.

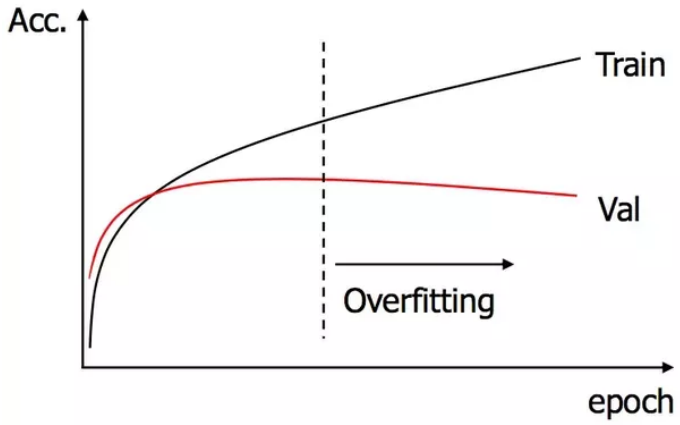


Figura 3.3: Andamento dell'accuratezza di una rete sul training set e il validation set con l'aumentare delle epoche

3.3.2 Apprendimento per rinforzo

Può essere considerato un caso particolare di apprendimento supervisionato. Infatti l'apprendimento con rinforzo differisce da quello supervisionato solo per il fatto che non sono mai presentate delle coppie input-output di esempi già noti in principio.

Per correggere il comportamento della rete e quindi per dare un feedback ad essa è presente un opportuno algoritmo che controlla l'ambiente nel quale l'output della rete agisce e fornisce in risposta un incentivo o un disincentivo, a seconda se si stia avvicinando all'obiettivo o meno.

3.3.3 Apprendimento non supervisionato

È basato su algoritmi d'addestramento che modificano i pesi della rete facendo esclusivamente riferimento ad un insieme di dati che include solo gli input. Tali algoritmi tentano di raggruppare i dati d'input e di individuare pertanto degli opportuni cluster rappresentativi dei dati stessi, facendo uso tipicamente di metodi topologici o probabilistici. L'apprendimento non supervisionato è anche impiegato per sviluppare tecniche di compressione dei dati.

Una di queste è quella che usa la cosiddetta regola di Hebb, in base alla quale il peso di una connessione aumenta di valore se gli output delle due unità connesse sono correlati positivamente e viene diminuito se essi sono correlati negativamente. Con tale procedura di apprendimento la rete impara a riconoscere ed estrarre le regolarità presenti negli input.

3.4 Tipologie

Esistono vari tipi di reti che si differenziano per struttura. Le principali sono le seguenti.

3.4.1 Reti Feed Forward

Le connessioni collegano i neuroni di un livello con i neuroni di un livello successivo. Non sono consentite connessioni all'indietro o connessioni verso lo stesso livello. È di gran lunga il tipo di rete più utilizzata.

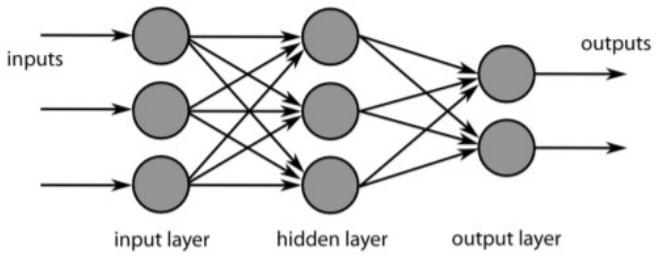


Figura 3.4: Rete Feed Forward

3.4.2 Reti Ricorrenti

A differenza delle architetture feedforward, le architetture ricorrenti presentano connessioni che partono da una data unità e ritornano, direttamente o indirettamente, alla stessa unità. Un tipo di architettura ricorrente può servire a dotare una rete neurale di una memoria a breve termine che conserva una traccia di quanto è avvenuto nei cicli precedenti. In ogni ciclo input-output, il pattern di attivazione delle unità interne viene memorizzato in uno speciale insieme di unità chiamate unità di memoria o 'unità di contesto', collegate alle unità interne tramite normali connessioni. Pertanto, in ogni ciclo, il livello di attivazione delle unità interne, e quindi anche l'output della rete, viene a dipendere non solo dalle unità di input, ma anche dalla traccia dello stato della rete nel ciclo precedente, memorizzata nelle unità

di memoria. Questo permette quindi di avere un comportamento dinamico temporale dipendente dalle informazioni ricevute agli istanti di tempo precedenti, rendendo così la rete ottimale per l'analisi di sequenze di dati, come audio, video e testo.

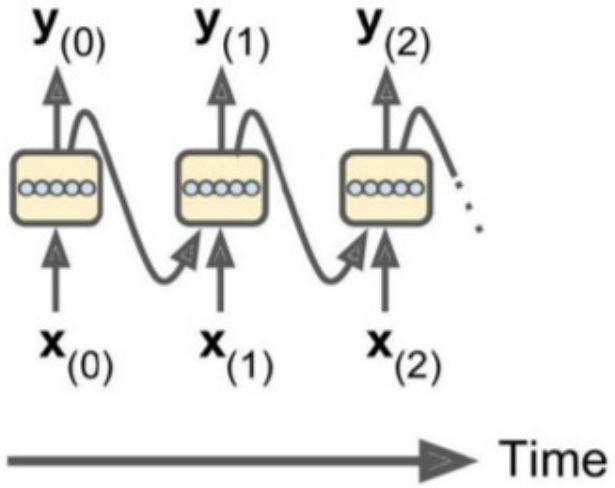


Figura 3.5: Concetto alla base delle reti ricorrenti

3.4.3 Reti Convoluzionali

Le reti Convoluzionali, o in inglese Convolutional Neural Networks (CNN), sono state introdotte da LeCun et al., a partire dal 1998. Le principali differenze rispetto ai precedenti metodi sono:

- **processing locale**: i neuroni sono connessi solo localmente ai neuroni del livello precedente, quindi ogni neurone esegue un'elaborazione locale. In questo modo si ha una forte riduzione del numero di connessioni;

- **pesi condivisi**: i pesi sono condivisi a gruppi, cioè neuroni diversi dello stesso livello eseguono lo stesso tipo di elaborazione su porzioni

diverse dell'input. Grazie a questo si ha una forte riduzione del numero di pesi.

Sono state esplicitamente progettate per processare immagini, per le quali il processing locale e i pesi condivisi non solo semplificano il modello, ma lo rendono più efficace rispetto a modelli fully connected visti in precedenza. Ovviamente possono essere utilizzate anche per altri tipi di pattern.

Al livello architetturale una rete convoluzionale è composta da una gerarchia di livelli. Il livello di input è direttamente collegato ai pixel dell'immagine, mentre nei livelli intermedi si utilizzano connessioni locali e pesi condivisi. Infine gli ultimi livelli sono generalmente fully-connected e operano allo stesso modo delle precedenti reti.

Le connessioni locali e condivise fanno sì che i neuroni processino nello stesso modo porzioni diverse dell'immagine. Si tratta di un comportamento desiderato, in quanto regioni diverse del campo visivo contengono lo stesso tipo di informazioni (bordi, spigoli, porzioni di oggetti, ecc.).

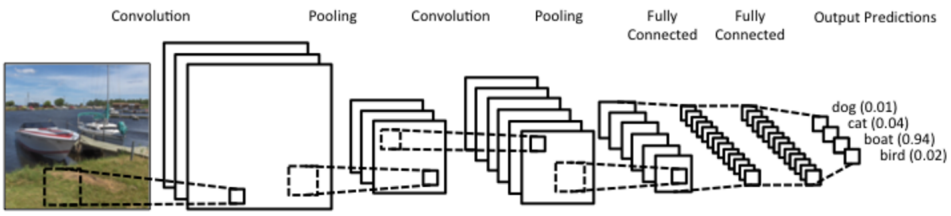


Figura 3.6: Struttura base delle reti convoluzionali

Capitolo 4

Progettazione

4.1 Analisi

Analizzando i dati che abbiamo a disposizione è possibile notare che tra le annotazioni di geni diversi possono esserci delle correlazioni, il che rende i dati di ogni gene confrontabili fra loro. Ad esempio, il più semplice tipo di relazione che ci può essere tra due geni è quello di similarità: se due geni hanno entrambi una certa annotazione in comune, è probabile che ce ne siano delle altre, e questo non vale solo per le annotazioni che generalizzano l'annotazione analizzata (dovuto alla struttura del vocabolario definito da Gene Ontology), ma anche per annotazioni totalmente scollegate dal punto di vista del grafo di Gene Ontology.

Queste correlazioni diventano molto utili quando ci sono geni che sono stati studiati affondo e altri relativamente sconosciuti. Infatti, in questi casi, i ricercatori possono usare i geni ben studiati come punto di riferimento per gli esperimenti da svolgere su quelli sconosciuti, aumentando così la probabilità di scoprire qualche nuova annotazione, velocizzandone la ricerca. Ma le annotazioni sono migliaia come anche i geni da analizzare, per cui trovare queste correlazioni non è così semplice.

Grazie all'informatica, utilizzando sistemi di apprendimento automatico, trovare correlazioni all'interno di grandi quantità di dati è diventato molto

più semplice. Possiamo in questo modo sfruttare le correlazioni scoperte per prevedere direttamente delle nuove annotazioni e la loro probabilità.

4.2 Metodo dell'Articolo di G. Domeniconi

Come punto di partenza è stato analizzato un articolo sull'argomento scritto da Giacomo Domeniconi pubblicato su Elsevier[6] che conferma l'utilità e l'efficacia dell'apprendimento automatico applicato alla genomica, riportando anche degli ottimi risultati che saranno presi come punti di riferimento.

Nel metodo usato nell'articolo si usano due matrici di anni diversi, in modo che in quella più recente ci siano più annotazioni. Questo può essere utile per aiutare il modello nella comprensione delle correlazioni e delle loro evoluzioni con l'avanzamento delle ricerche. Prima di procedere con gli esperimenti c'è stata un'elaborazione dei dati che in questo caso consisteva nell'eliminare tutti i termini GO e geni non in comune tra i genomi dei due anni diversi dello stesso organismo. Inoltre vengono eliminati anche i geni con un numero di annotazioni inferiore a 5 perché questi possono condizionare negativamente l'allenamento del modello.

Infine è stato allenato un modello di classificazione con degli algoritmi supervisionati (IBk nearest neighbors, Random Forest e LibSVM Support Vector Machine) dando come input le righe della matrice vecchia, senza la colonna del termine GO che si vuole predire, mentre come label i valori della colonna rimossa (cioè quella del termine GO da predire) che può essere presa dalla medesima matrice dell'input o da quella più recente, come mostrato nella figura 4.1. In questo modo si ha un modello in grado di prevedere la presenza o meno dell'annotazione in quello specifico termine GO messo come label, perciò ovviamente per provarlo su tutto il genoma è stato addestrato e usato un modello diverso per ogni colonna.

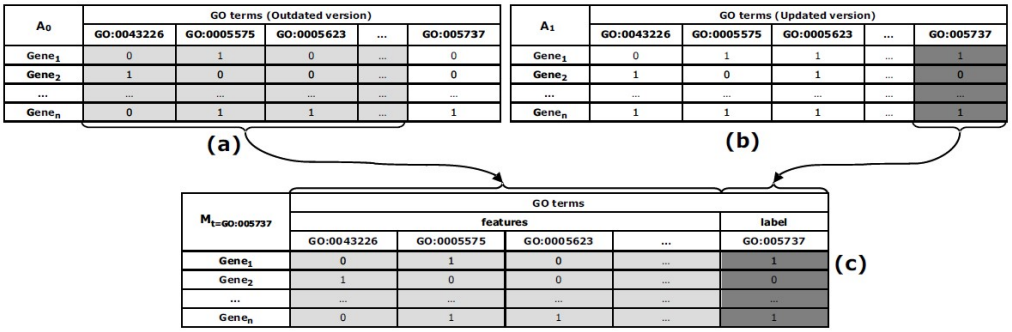


Figura 4.1: Esempio grafico del metodo dell'articolo di riferimento

4.3 Idee di applicazione delle reti neurali

Ciò che si vuole dimostrare in questi esperimenti è l'efficacia dell'uso delle reti neurali, al posto degli algoritmi supervisionati usati nell'articolo di Domeniconi, e la possibilità di avere dei risultati migliori o almeno alla pari di quelli dell'articolo.

Per non complicare la vita al modello in fase d'allenamento e per rimanere coerenti all'articolo, si rimuovono tutti i geni con un numero di annotazioni inferiore ad un certo limite, nel nostro caso fissato a 5.

Perché rimuovere questi geni? Perché non sono necessari all'allenamento della rete in quanto contengono troppe poche annotazioni per permettere al modello di ottenere informazioni rilevanti, anzi, in alcuni casi ne potrebbe anche peggiorare l'accuratezza.

Dopo aver analizzato i dati di cui siamo in possesso e delle varie tipologie di reti applicabili si è deciso di usare le reti di tipo feed-forward. Non sono state scelte le reti ricorrenti in quanto non sono utili in questa tipologia di esperimenti, perché l'ordine con cui si analizzano i geni è indipendente dalla predizione quindi non è necessaria una rete con memoria.

Per quanto riguarda invece le reti convoluzionali, era stata fatta l'ipotesi di trattare l'intera matrice di annotazioni come un'immagine da correggere, ma così facendo la rete andrebbe alla ricerca di pattern bidimensionali all'in-

terno dell'input, cosa improbabile da trovare perché ogni gene, e quindi ogni riga, è indipendente e non ha necessariamente un legame logico con i geni adiacenti; sono presenti solo pattern unidimensionali (nelle righe). Inoltre è probabile che in un gene ci sia la correlazione tra due termini GO distanti nella matrice, ad esempio tra una delle prime colonne e una delle ultime. Perciò il processing locale in questo caso non è efficace, anzi, rischia di essere controproducente.

Quindi sono state usate due metodologie diverse, ma che usano sempre le reti feed forward.

4.3.1 Reti neurali per colonna

In questa tipologia di esperimenti si è seguito quasi del tutto l'esempio dell'articolo, analizzando colonna per colonna, usandole una ad una come label, cioè come è stato illustrato nella figura 4.1. L'unica differenza è che nell'allenamento del modello, invece di usare gli algoritmi supervisionati, è stata usata una rete neurale.

Anche nella preparazione dei dati è stato usato lo stesso procedimento: per ogni esperimento sono stati selezionati i termini GO in comune tra i due organismi che si stanno analizzando, eliminando gli altri, inoltre sono stati rimossi anche i geni con meno di 5 annotazioni.

Con questo metodo è difficile avere la certezza che la struttura della rete neurale usata sia quella ottimale perché andrebbero fatti dei test per ogni singola colonna, in quanto ognuna di esse può avere un proprio modello ottimo con iperparametri diversi.

4.3.2 Autoencoder

Questo metodo l'abbiamo chiamato autoencoder perché la struttura della rete neurale è quella di un autoencoder, cioè una rete avente un collo di bottiglia nei livelli intermedi, quindi con un numero minore di percettroni rispetto all'input e all'output. In un autoencoder l'importante è la compressione dei

dati e la riduzione di dimensionalità che avviene nella parte centrale della rete, infatti poi, se sfruttata con questa finalità, la rete viene divisa a metà:

- **Encoder**: è la prima parte della rete che prende come input il dato nella sua interezza e lo restringe fino ad arrivare al punto in cui la rete ha meno percettroni. Quindi l'output consiste nella compressione del dato inserito come input.
- **Decoder**: è la seconda parte della rete che invece fa il contrario dell'encoder: riceve come input il dato compresso e lo decomprime facendolo tornare alla sua forma originale.

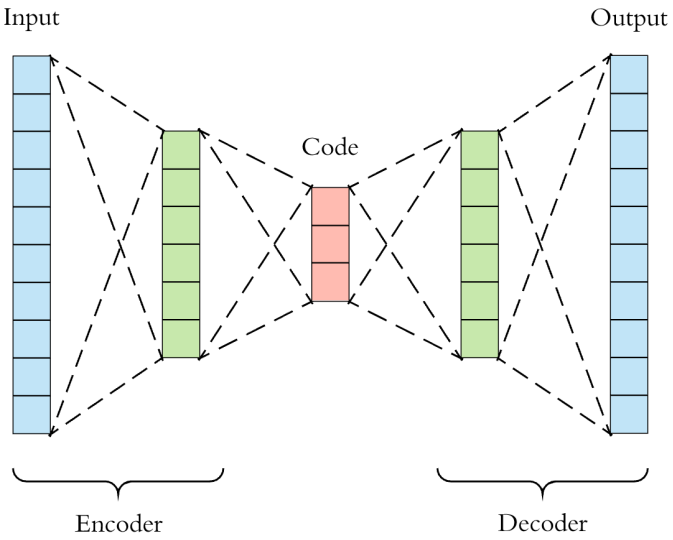


Figura 4.2: Struttura di un Autoencoder

In questo caso invece si sfrutta il restringimento di dimensioni per permettere alla rete di individuare le correlazioni tra le varie annotazioni e quindi di aggiungere delle annotazioni in fase di decode sfruttando queste ultime.

Come si può capire dalla struttura della rete, a differenza dell'articolo, vengono passati come input gli interi vettori dei geni con tutte le annotazioni, senza toglierne nessuna, e come label per l'allenamento i rispettivi vettori dei geni. Nell'autoencoder l'input e l'output per l'addestramento dovrebbero

essere identici, ma questa rete non ha la finalità di un normale autoencoder, quindi in alcuni degli esperimenti è stata messa come output una matrice più recente rispetto l'input.

In questo caso, per rendere più generico possibile il modello e quindi permettere che questo possa essere allenato e testato con i geni di tutti gli organismi, sono state aggiunte in tutte le matrici le colonne (ovviamente azzerate) per i termini GO presenti anche in una sola delle matrici degli organismi presi in considerazione per gli esperimenti. Questo è possibile perché di fatto non si aggiungono dei dati alla matrice in quanto le colonne inserite sono completamente a zero e servono per lo più a rispettare la dimensione dell'input della rete. Questo rende possibile il fatto di prevedere annotazioni che nelle matrici iniziali degli organismi "Target" non venivano nemmeno considerate, ammesso che si sfrutti il Transfer Learning Inter-Organismo (vedi il capitolo successivo) e che le matrici dell'organismo Source abbiano termini GO in più rispetto ai Target.

4.4 Transfer Learning Inter-Organismo

Fino ad ora è stato detto in generale di ciò che è possibile applicare al tipo di dati che possediamo, senza specificare niente su di quale organismo fossero i dati. Durante l'addestramento delle reti è ovvio che vengano usati dati dello stesso essere vivente. Ma una volta allenata la rete, su quali dati può essere sfruttata?

Grazie al fatto che gli organismi condividono la maggior parte del DNA e al vocabolario comune costruito da Gene Ontology, è possibile sfruttare la rete sui dati di qualsiasi organismo, consapevoli però che più ci si allontana dal punto di vista evolutivo dall'organismo usato per l'allenamento della rete, maggiori saranno le differenze e la rete sarà meno efficace. In questo modo si sta mettendo in atto un trasferimento di conoscenza tra organismi differenti, metodo chiamato "Transfered Learning Inter-Organismo".

4.5 Perturbazione dei dati

Spesso per allenare i modelli vengono usati i dati presi da due genomi di anni differenti, in modo che quello più recente abbia un numero maggiore di annotazioni dovuto all'avanzamento della ricerca. Questo però spesso non è possibile per l'indisponibilità di dati meno recenti di alcuni organismi. Per cercare di risolvere questo problema si è deciso di ricreare una finta matrice meno recente perturbando quella di cui siamo in possesso, cioè andando a rimuovere alcune annotazioni in modo casuale per una certa percentuale.

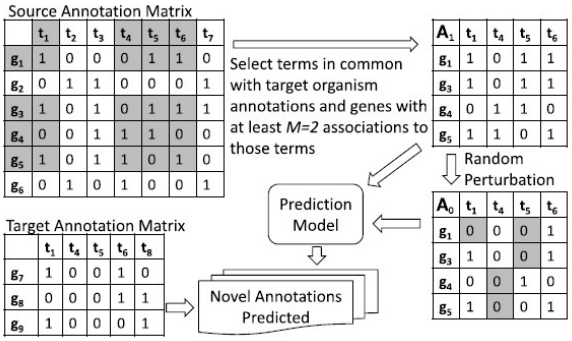


Figura 4.3: Schema riassuntivo di esperimenti dell'articolo con dati perturbati

Capitolo 5

Implementazione

5.1 Linguaggio e librerie

Come linguaggio è stato usato Python, uno dei linguaggi più utilizzati nell'ambito dell'analisi dei dati (insieme ad R) per la sua dinamicità, semplicità e flessibilità.

Per quanto riguarda le librerie che sono state usate sono:

- **Arff**, modulo per la lettura dei file .arff;
- **NumPy**, estensione che aggiunge funzioni matematiche a quelle che ha già di base di Python;
- **Pandas**, libreria che aggiunge funzioni utili per la manipolazione e analisi dei dati, resa possibile grazie alle strutture dati e alle operazioni applicabili su di esse che mette a disposizione;
- **Tensorflow**, libreria che fornisce moduli testati ed ottimizzati per la realizzazione di algoritmi nell'ambito dell'apprendimento automatico
- **Keras**, libreria utile per la costruzione di reti neurali che funziona usando come base un'altra libreria, Tensorflow o Theano, rendendo l'uso di queste ultime più user-friendly.

5.2 Autoencoder

La prima idea è stata quella di implementare una rete neurale che ricevesse come input l'intero vettore del gene e che prevedesse direttamente tutte le sue annotazioni. Inoltre sono stati elaborati i dati in modo che la struttura della rete fosse uguale per tutti gli organismi a prescindere da quale sia l'organismo Source, usato per l'addestramento della rete, o l'organismo Target, su cui si vuole usare la rete per prevedere le sue annotazioni.

5.2.1 Elaborazione dati

Per rendere possibile una cosa del genere è stato fatto uno script che prendesse tutti i termini GO di ogni organismo dai rispettivi file arff. Una volta composte le due liste dei termini GO necessari (una per quelli delle tabelle del 2009 e l'altra per quelli del 2013), vengono usate per aggiungere i termini GO mancanti nelle tabelle sotto forma di colonne completamente azzerate. Siccome le due tabelle di anni diversi hanno un numero di righe diverso, vuol dire che ci sono geni che compaiono solo in una delle due. Per questo motivo sono stati eliminati tutti i geni che le due tabelle non avevano in comune, in modo che poi si potessero usare per l'addestramento come input e label.

Di seguito c'è il codice per estrarre le tabelle dai file arff e fare ciò che è stato detto fino ad ora per la realizzazione della tabella dell'Homo Sapiens del 2009 (in tutto il progetto ho preferito chiamare input le tabelle del 2009 e output le tabelle del 2013):

```
print("Caricamento arff 2009")
dataBTinput = arff.load(open('bt_2009_unfolded.arff'))
dataHSinput = arff.load(open('hs_2009_unfolded.arff'))
dataMMinput = arff.load(open('mm_2009_unfolded.arff'))
dataRNinput = arff.load(open('rn_2009_unfolded.arff'))
BTinput = []
HSinput = []
MMinput = []
```

```
RNinput = []
for k, v in dataBTinput["attributes"]:
    BTinput.append(k)
for k, v in dataHSinput["attributes"]:
    HSinput.append(k)
for k, v in dataMMinput["attributes"]:
    MMinput.append(k)
for k, v in dataRNinput["attributes"]:
    RNinput.append(k)
print("Creazione dell'array con i GO del 2009")
inputMix = list(set(BTinput).union(set(RNinput)).union(set(
    HSinput)).union(set(MMinput)))
print("Creazione dataframe per gli esseri umani con i dati del
    2009")
HSdfin =  pd.DataFrame(dataHSinput["data"])
HSdfin.set_axis(HSinput, axis='columns', inplace=True)
HSdfin.set_index('gene', inplace=True)
print("Aggiunta delle colonne mancanti nel dataframe di input")
for i in inputMix:
    if i not in HSinput:
        HSdfin[i] = 0
print("Eliminazione dei geni non presenti nell'output dall'input
    ")
for i in HSdfin.index.values:
    if i not in HSdfout.index.values:
        HSdfin.drop(i, inplace=True)
```

La stessa cosa è stata ripetuta anche per le altre tabelle, di qualsiasi organismo o anno siano.

Oltre a questo sono stati rimossi i geni con meno di 5 annotazioni (soglia M) dalle matrici d'allenamento sia del 2009 che quelle del 2013 come mostrato nell'articolo preso d'esempio, in quanto rischiano di peggiorare le capacità predittive della rete. Poi verranno mostrati anche i risultati di un esperimento che evidenzia la differenza tra le reti allenate con matrici da cui sono stati rimossi i geni con un numero differente di annotazioni.

Questo è il codice usato per rimuovere i geni con meno di M annotazioni

dalle matrici del Homo Sapiens:

```
M=5
gene=[]
for i in HSdfinput.index.values:
    n=0
    for y in HSdfinput.loc[i]:
        n+=y
    if n >= M:
        gene.append(i)
HSdfinput=HSdfinput.loc[gene]
HSdfoutput=HSdfoutput.loc[gene]
```

5.2.2 Ottimizzazione dei Parametri

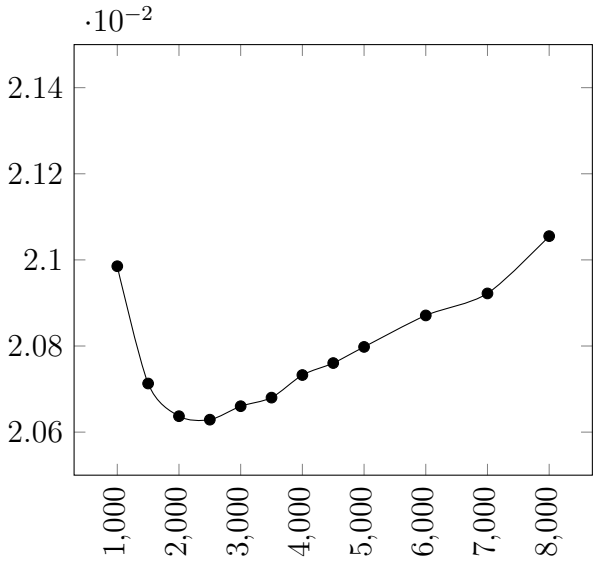
Prima di tutto sono stati fatti dei test per vedere se era fattibile ed efficace l'idea di avere come input e output gli interi vettori di geni e sono subito arrivati risultati piuttosto positivi. Per trovare la configurazione migliore per il nostro problema è stato scritto uno script che riceve come argomenti tutti gli iperparametri della rete neurale, così da poter eseguire di seguito, grazie alla scrittura di un altro script in bash, tutti gli esperimenti di cui si vuole sapere l'efficacia.

Di seguito riporto alcune tabelle con i dati dei vari esperimenti fatti per la ricerca del modello migliore al variare di ogni parametro, eseguiti usando come input la tabella dell'Homo Sapiens del 2009 e come label quella del 2013. Tutti questi esperimenti sono stati fatti con un solo strato intermedio perché facendo alcune prove aggiungendo degli stati sono stati ottenuti dei risultati peggiori, ma è probabile che con alcune configurazioni che non ho provato si potessero ottenere dei migliori.

Numero di percettroni

Num. di percettroni	Binary Crossentropy
1000	0.0209853
1500	0.0207129
2000	0.0206370
2500	0.0206289
3000	0.0206602
3500	0.0206801
4000	0.0207326
4500	0.0207604
5000	0.0207979
6000	0.0208710
7000	0.0209221
8000	0.0210552

Tabella 5.1: Andamento Binary Crossentropy al variare del numero di percettroni



Come si può vedere dalla tabella i risultati ottenuti dimostrano che la configurazione migliore deve avere tra i 2000 e i 2500 percettroni nel suo livello intermedio. Quindi la migliore struttura per la rete risulta essere quella di una sottospecie di autoencoder, con un input di 4922 valori, un livello intermedio di circa 2500 percettroni e un output di 8020 valori.

Funzione d'attivazione

Funz. d'attivazione	Binary Crossentropy
Sigmoid	0.0206289
Relu	0.0211596
Tanh	0.0215862

Tabella 5.2: Andamento Binary Crossentropy al variare della funzione d'attivazione del livello intermedio

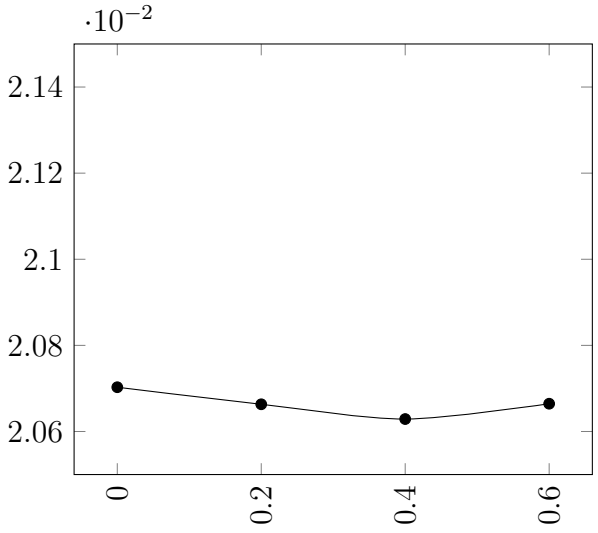
Per la funzione d'attivazione del livello intermedio la migliore è la sigmoid. Si è dimostrata migliore anche per la funzione d'attivazione dell'output. In questo caso sono state messe a confronto sigmoid e softmax per la natura dell'output che si desidera, cioè un valore da 0 a 1.

Dropout

Il dropout è una tecnica per ridurre l'overfitting, in cui ad ogni passo dell'addestramento un certo numero di neuroni scelti casualmente nei livelli nascosti viene "spento", impedendo che il suo output si propaghi ai neuroni successivi.

Dropout	Binary Crossentropy
0	0.0207028
0.2	0.0206632
0.4	0.0206289
0.6	0.0206645

Tabella 5.3: Andamento Binary Crossentropy al variare del dropout



Per quanto riguarda il dropout pur variandolo non cambia molto la precisione della rete, ma settandolo a 0.40 si ottengono i risultati migliori.

Funzione d'ottimizzazione

Gli ottimizzatori definiscono il modo in cui una rete neurale è allenata, ossia sono gli algoritmi con cui si allena una rete neurale. Nei vari esperimenti, testando Adam, Adagrad, Adadelta, Nadam, Adamax, si è notato che il migliore per il nostro problema è Adam.

5.2.3 Previsioni

Una volta trovata una buona configurazione della rete si passa ai veri e propri esperimenti con tutti gli organismi.

Quindi si procede alla creazione della rete e il suo allenamento, facendo uso dei dati scelti: tra i vari esperimenti oltre ad allenare la rete usando matrici di anni diversi abbiamo provato ad esempio anche ad usare la stessa matrice sia come input che output.

```
model = Sequential()
model.add(Dense(2500, input_shape=(XHS_train.columns.size,)))
model.add(Activation('sigmoid'))
model.add(Dropout(0.4))
model.add(Dense(YHS_train.columns.size))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy', optimizer="Adam",
    metrics=['binary_accuracy'])
history = model.fit(XHS_train, YHS_train, batch_size=64, epochs
    =21, verbose=1, validation_split=0.3)
```

Infine si passa all'uso della rete e di conseguenza alla previsione di nuove annotazioni usandola su tutti gli organismi: Bos Taurus, Mus Musculus e Rattus Norvegicus.

```
BTpredict = model.predict(BTdfinput)
MMpredict = model.predict(MMdfinput)
RNpredict = model.predict(RNdfinput)
```

Per concludere, ovviamente, si va a misurare il numero di annotazioni aggiunte alla matrice e quante di esse sono confermate, confrontandola con la matrice più recente, andando così a calcolare la cosiddetta accuratezza della previsione (precision). Questo valore corrisponde al rapporto tra il numero di annotazioni predette in più e confermate dalla matrice più recente, e la totalità delle annotazioni aggiunte. Siccome la rete prevede la probabilità che un'annotazione ci sia nella matrice predetta o meno, si avranno valori che saranno nell'intervallo tra 0 e 1 e non valori discreti. Per cui si decide una soglia oltre la quale le annotazioni vengono considerate a 1. Per rimanere fedeli all'articolo abbiamo deciso di mettere come soglia 0.8, ma sono stati fatti anche degli esperimenti che mostrano l'andamento dell'accuratezza della previsione al variare di questa soglia.

```
matrix=BTdfoutput.values
matrixOr=BTdfinput.values
colOut=np.array(MMdfoutput.columns.values)
colIn=np.array(MMdfinput.columns.values)
c=0
count=0
countP=0
for g in predict:
    d=0
    for i in g:
        if i>0.8 and colOut[d] in colIn and matrixOr[c][np.where
    (colIn == colOut[d])]==0:
            if matrix[c][d]==1:
                countP+=1
            count+=1
        d+=1
    c+=1
print(countP,"/",count,"=",countP/count)
```

5.3 Rete Neurale per termine GO

Tutto ciò che è stato fatto per l'autoencoder poi è stato ripetuto con quest'altro metodo. È stato chiamato "Rete neurale per termine GO" perché si fa uso delle reti neurali per prevedere la presenza di annotazioni per un singolo termine GO, per cui deve essere creata una rete neurale per ognuno di essi. Quindi è lo stesso metodo usato nell'articolo di riferimento, ma con l'uso delle reti al posto degli algoritmi supervisionati.

5.3.1 Elaborazione dati

A differenza dell'autoencoder in questo caso si sono dovute eliminare le colonne dei termini GO non condivisi tra le matrici dei due anni. Inoltre, partendo dai dati dell'autoencoder, è stato necessario eliminare anche i termini GO senza annotazioni che erano stati aggiunti. Per ogni esperimento

sono stati tenuti solo i termini GO che condividono i due organismi che si stanno esaminando come Source e Target. Questo è il codice che è stato usato per questa elaborazione, partendo dai dati elaborati per l'autoencoder:

```
annotComuniHS=[]
HSina=HSdfinput.columns.values
HSouta=HSdfoutput.columns.values
for a in HSina:
    if a in HSouta:
        annotComuniHS.append(a)
annHS0=[]
a = HSdfinput.loc[:, (HSdfinput != 0).any(axis=0)].columns.
    values
b = HSdfoutput.loc[:, (HSdfoutput != 0).any(axis=0)].columns.
    values
for i in a:
    if i in annotComuniHS and i not in annHS0:
        annHS0.append(i)
for i in b:
    if i in annotComuniHS and i not in annHS0:
        annHS0.append(i)
annBT0=[]
a = BTdfinput.loc[:, (BTdfinput != 0).any(axis=0)].columns.
    values
b = BTdfoutput.loc[:, (BTdfoutput != 0).any(axis=0)].columns.
    values
for i in a:
    if i in annotComuniHS and i not in annBT0:
        annBT0.append(i)
for i in b:
    if i in annotComuniHS and i not in annBT0:
        annBT0.append(i)
AnnComHSBT=[]
for i in annHS0:
    if i in annBT0:
        AnnComHSBT.append(i)
AnnComHSBT=np.array(AnnComHSBT)
AnnComHSBT.sort()
```

```
31 HSdfinput=HSdfinput[AnnComHSBT]
32 HSdfoutput=HSdfoutput[AnnComHSBT]
33 BTdfinput=BTdfinput[AnnComHSBT]
34 BTdfoutput=BTdfoutput[AnnComHSBT]
```

Come per l'autoencoder ovviamente anche in questo caso sono stati rimossi i geni con meno di 5 annotazioni.

5.3.2 Ottimizzazione dei Parametri

Siccome va creata una rete neurale per ogni termine GO, per l'ottimizzazione della configurazione in questo caso sarebbe necessario fare gli stessi test fatti per l'autoencoder, ma per ogni termine GO, perché ognuno di essi avrà probabilmente una configurazione ottima diversa. Dato che i termini GO sono migliaia e la stessa cosa si deve fare per ogni coppia di organismi su cui vogliamo fare gli esperimenti, è stato possibile solo trovare un buon compromesso facendo i test su alcuni termini GO.

Sono state provate le reti neurali con massimo 2 layer intermedi, perché con l'aumentare dei layer aumenta anche il numero di parametri rendendo l'allenamento della rete molto più pesante e lento. In questo caso, sembra che i risultati migliori siano stati ottenuti dalle reti con 2 strati intermedi, ma è sempre possibile che ci siano delle configurazioni che non sono state provate. Usando reti neurali con 2 layer intermedi si può già dire che si sta facendo uso del Deep Learning.

Con il fatto che si è aggiunto uno strato, quindi altri parametri, e che per ogni termine GO andrebbe ricercata la configurazione ottimale della rete neurale, è diventato complesso poter fare la stessa cosa dell'autoencoder, cioè rappresentare l'andamento dell'accuratezza espressa dalla rete al variare di un parametro. Ciò è stato dimostrato dai vari test, in cui si è visto come, variando un parametro, si poteva peggiorare l'accuratezza della previsione, ma modificandone un altro oltre a quello, era possibile ottenere dei risultati migliori.

In linea di massima si sono dimostrate migliori le reti aventi: come funzione di ottimizzazione Adadelta, un numero di percettroni che si riduce livello per livello e come ultima funzione di attivazione la sigmoid.

5.3.3 Previsioni

Anche questo processo è differente rispetto all'autoencoder, soprattutto perché, per poter fare tutto in modo automatico, è stato creato uno script che faccia la creazione, l'allenamento, la previsione della rete e il salvataggio delle previsioni in un'apposita cartella per ogni termine GO, autonomamente. Inoltre per rendere più semplice il settaggio dei parametri della rete neurale si è fatto in modo che venissero passati tramite gli argomenti dello script. Siccome per l'esecuzione di questo script è necessario un giorno o più, è possibile che il processo si interrompa per qualche motivo. È per questo che ad ogni colonna lo script salva la loro previsione, avendo così la possibilità di ripartire da dove era arrivato.

```
c=0
for GO in HSdfinput.columns.values:
    if not os.path.isfile("BTpredict/"+repr(c)+".csv"):
        predict=pd.DataFrame(index=BTdfinput.index)
        X2009 = HSdfinput.copy()
        Y2009 = X2009.pop(GO)
        print()
        print(GO)
        print(repr(c)+"/"+repr(HSdfinput.columns.size - 1))
        XHS_train, XHS_test, YHS_train, YHS_test =
    train_test_split(X2009, Y2009, test_size=0, random_state=42)
        model = Sequential()
        model.add(Dense(int(sys.argv[6]), input_shape=(XHS_train
    .columns.size,)))
        model.add(Activation(sys.argv[7]))
        model.add(Dropout(float(sys.argv[8])))
        if int(sys.argv[5])>=1:
            model.add(Dense(int(sys.argv[9])))
            model.add(Activation(sys.argv[10]))
```

```
18              model.add(Dropout(float(sys.argv[11])))
19          model.add(Dense(1))
20          model.add(Activation(sys.argv[4]))
21          model.summary()
22          model.compile(loss=sys.argv[2], optimizer=sys.argv[3],
    metrics=['binary_accuracy'])
23          history = model.fit(XHS_train, YHS_train, batch_size
    =300, epochs=25, verbose=0, validation_split=0.3)
24
25          X2009=BTdfinput.copy()
26          Y2009 = X2009.pop(GO)
27          predict.insert(loc=0, column=GO, value=model.predict(
    X2009))
28          predict=predict.dropna()
29          predict.to_csv("BTpredict/"+repr(c)+".csv",sep=',')
```

Una volta finita l'esecuzione dello script è possibile unire tutte le colonne per creare la matrice predetta, su cui verrà poi calcolata l'accuratezza e misurato il numero di annotazioni aggiunte, con il codice già usato per l'autoencoder.

Questo è il codice usato per unire le colonne:

```
predict=pd.DataFrame(index=BTdfinput.index)
c=0
for GO in BTdfinput.columns.values:
    a=pd.read_csv("BTpredict/"+repr(c)+".csv")
    predict[GO]=a[GO].values
    predict=predict.dropna()
    c+=1
predict.to_csv("BTpredict.csv",sep=',')
```

Capitolo 6

Risultati

In questo capitolo vengono mostrati i risultati dei migliori esperimenti confrontandoli con i dati dell'articolo di riferimento, così da avere un raffronto tra i risultati delle reti neurali e quelli degli algoritmi supervisionati.

Come si può notare dai titoli delle prossime sezioni, gli esperimenti si differenziano per i dati usati per l'addestramento delle reti. Per quanto riguarda invece l'uso delle reti, sono state applicate sempre sulle matrici del 2009 dell'organismo Target, mentre, per capire quante annotazioni aggiunte sono confermate dai studi più recenti, la matrice predetta è stata confrontata con quella del 2013.

6.1 Matrice d'input uguale a label

In questa prima tabella si mostrano i risultati ottenuti usando i modelli addestrati con input e label derivanti dalla stessa matrice (nella colonna dell'organismo Source è specificato l'anno). Ovviamente i dati sono stati ricavati confrontando la matrice predetta con quella più recente, cioè quella del 2013. Sono presenti caselle bianche nella colonna "Articolo" perché il dato non viene scritto nell'articolo di riferimento e quindi non è disponibile.

Prima di mostrare la tabella, si illustrano le strutture delle reti utilizzate:

- Autoencoder: siccome ciò che viene messo nel label deve derivare dalla stessa matrice che si mette in input, i modelli di tutti gli esperimenti fatti con l'autoencoder hanno come input e label vettori della stessa dimensione, quindi con lo stesso numero di colonne (4746).

  Struttura delle reti usate:

  1. Input layer, dimensione 4746
  2. Layer intermedio, dimensione 2500, funzione d'attivazione Sigmoid, dropout 0.4
  3. Output layer, dimensione 4746, funzione d'attivazione Sigmoid

  Ottimizzatore: Adam; Batch-size:64; Epochs:30(tranne per quelli che hanno come organismo Source Bos Taurus che avendo pochi geni ha bisogno di più epochs, circa 70); Validation Split: 0.3

- Rete neurale per Termine GO: per ogni esperimento sono state selezionate per l'input le colonne in comune tra l'organismo Source e quello Target con almeno una annotazione, quindi ogni modello ha un input di dimensione differente. Il label invece, come già detto in precedenza, corrisponde a un valore. Però costruendo una rete neurale per prevedere ogni termine GO è possibile ricostruire la matrice.

  Struttura delle reti usate:

  – Source Homo Sapiens, Target Bos Taurus:

    1. Input Layer, dimensione 1854
    2. Layer Intermedio, dimensione 700, funzione d'attivazione Tanh
    3. Layer Intermedio, dimensione 50, funzione d'attivazione Sigmoid, dropout 0.6
    4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

    Ottimizzatore: Adadelta; Batch-size:300; Epochs:25; Validation Split: 0.3

- Source Mus Musculus, Target Bos Taurus:
  1. Input Layer, dimensione 1854
  2. Layer Intermedio, dimensione 1000, funzione d'attivazione Tanh, dropout 0.2
  3. Layer Intermedio, dimensione 500, funzione d'attivazione Sigmoid, dropout 0.4
  4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

  Ottimizzatore: Adadelta; Batch-size:300; Epochs:30; Validation Split: 0.3
- Source Bos Taurus, Target Bos Taurus:
  1. Input Layer, dimensione 1854
  2. Layer Intermedio, dimensione 1000, funzione d'attivazione Tanh, dropout 0.2
  3. Layer Intermedio, dimensione 500, funzione d'attivazione Sigmoid, dropout 0.4
  4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

  Ottimizzatore: Adadelta; Batch-size:300; Epochs:30; Validation Split: 0.3
- Source Homo Sapiens, Target Mus Musculus:
  1. Input Layer, dimensione 4092
  2. Layer Intermedio, dimensione 2200, funzione d'attivazione Tanh, dropout 0.2
  3. Layer Intermedio, dimensione 450, funzione d'attivazione Sigmoid, dropout 0.6
  4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

  Ottimizzatore: Adadelta; Batch-size:300; Epochs:20; Validation Split: 0.3
- Source Mus Musculus, Target Mus Musculus:

1. Input Layer, dimensione 4604
2. Layer Intermedio, dimensione 2000, funzione d'attivazione Tanh, dropout 0.2
3. Layer Intermedio, dimensione 400, funzione d'attivazione Sigmoid, dropout 0.4
4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

Ottimizzatore: Adadelta; Batch-size:300; Epochs:15; Validation Split: 0.3

– Source Bos Taurus, Target Mus Musculus:

1. Input Layer, dimensione 1854
2. Layer Intermedio, dimensione 1000, funzione d'attivazione Tanh, dropout 0.2
3. Layer Intermedio, dimensione 500, funzione d'attivazione Sigmoid, dropout 0.4
4. Output Layer, dimensione 1, funzione d'attivazione Sigmoid

Ottimizzatore: Adadelta; Batch-size:300; Epochs:60; Validation Split: 0.3

Legenda:

- N = numero di annotazioni trasformate da 0 in 1, in quanto con una probabilità superiore a 0.8, come nell'articolo di riferimento.

- Acc. = Accuratezza delle annotazioni predette, quindi il numero di annotazioni che sono state aggiunte nella previsione e sono presenti anche nella matrice più recente (le predizioni confermate in pratica), diviso N.

Target	Source	Articolo		Autoencoder		NN x Term	
		Acc.	N	Acc.	N	Acc.	N
Bos T.	Homo S. 2009	0.538	234	0.715	200	0.349	272
	Homo S. 2013			0.875	376	0.668	937
	Mus M. 2009	0.137	388	0.660	162	0.541	244
	Mus M. 2013			0.870	407	0.725	895
	Bos T. 2009	0.250	4	0.244	45	0.244	598
	Bos T. 2013			0.817	131	0.903	380
Mus M.	Homo S. 2009	0.573	6048	0.430	1340	0.324	1834
	Homo S. 2013			0.893	7005	0.740	11702
	Mus M. 2009	0.407	205	0.238	307	0.295	871
	Mus M. 2013			0.954	6590	0.867	13177
	Bos T. 2009	0.312	2184	0.430	989	0.354	1895
	Bos T. 2013			0.713	1186	0.534	4821
Rattus N.	Homo S. 2009	-		0.286	2141	-	
	Homo S. 2013			0.799	7929		
	Mus M. 2009			0.266	1237		
	Mus M. 2013			0.886	8651		
	Bos T. 2009			0.327	1354		
	Bos T. 2013			0.660	2418		

Dalla tabella si può notare che l'autoencoder a volte dà un'accuratezza migliore rispetto agli algoritmi supervisionati, ma la maggior parte delle volte prevede un numero inferiore di annotazioni. Dalla colonna denominata "NN x Term", abbreviazione di "Rete neurale per Termine GO", si può capire che grazie a questo metodo si prevede un numero maggiore di annotazioni, ma generalmente si è dimostrato inferiore dal punto di vista dell'accuratezza rispetto all'autoencoder.

Facendo altri esperimenti con l'autoencoder, restringendo il primo strato fino a 200 percettroni, si riesce ad aumentare l'accuratezza nei casi in cui gli organismi Mus Musculus e Rattus Norvegicus sono i target. Ad esempio in quello che ha come Source e come Target Mus Musculus 2009 si riesce ad

arrivare anche a 0.49 di accuratezza mantenendo comunque 300 annotazioni predette. Nel caso invece ci sia come target il Bos Taurus il restringimento del primo strato peggiora l'accuratezza.

6.2 Matrice d'input del 2009, Label del 2013

In questa seconda tabella rispetto alla prima cambia solo una cosa, cioè che come label, invece di usare la stessa matrice, è stata usata quella più recente, cioè quella del 2013 (per questo non è specificato l'anno nella colonna dell'organismo Source come nella tabella precedente).

I modelli usati per gli esperimenti di entrambi le metodologie sono gli stessi esposti prima.

Target	Source	Articolo		Autoencoder		NN x Term	
		Acc.	N	Acc.	N	Acc.	N
Bos T.	Homo Sapiens	0.682	292	0.638	810	0.615	1114
	Mus Musculus	-		0.443	2058	0.431	2372
	Bos Taurus			0.509	161	-	
Mus M.	Homo Sapiens	0.742	5799	0.825	15599	0.753	27603
	Mus Musculus	-		0.756	13473	-	
	Bos Taurus			0.638	3959	0.679	11589
Rattus N.	Homo Sapiens	-		0.552	25695	-	
	Mus Musculus			0.584	49163		
	Bos Taurus			0.536	7691		

A differenza dell'articolo, entrambi i metodi qui proposti dimostrano di prevedere un numero superiore di annotazioni pur rimanendo più o meno in linea con l'accuratezza degli esperimenti dell'articolo.

6.3 Matrici perturbate

In questa tabella invece si vuole mostrare l'uso della perturbazione delle matrici. Infatti in questo caso per allenare il modello viene usata come input

la matrice del 2009 perturbata, cioè con qualche annotazione originariamente a 1 messa a 0 in modo casuale (in questo esperimento con una probabilità del 0.10), e come label la matrice originale. Le strutture dei modelli sono le stesse esposte nella prima tabella.

Target	Source	AutoEncoder		Mat. Perturbata	
		Acc.	N	Acc.	N
Bos Taurus	Homo Sapiens	0.715	200	0.620	274
	Mus Musculus	0.660	162	0.572	327
	Bos Taurus	0.244	45	0.221	104
Mus Musculus	Homo Sapiens	0.430	1340	0.401	1994
	Mus Musculus	0.234	307	0.218	775
	Bos Taurus	0.430	989	0.397	2487
Rattus Norvegicus	Homo Sapiens	0.286	2141	0.268	2782
	Mus Musculus	0.266	1237	0.246	2383
	Bos Taurus	0.327	1354	0.300	3454

A differenza dell'articolo, con l'autoencoder la perturbazione non riesce a migliorare l'accuratezza, anzi si abbassa leggermente, ma allo stesso tempo permette di prevedere più annotazioni.

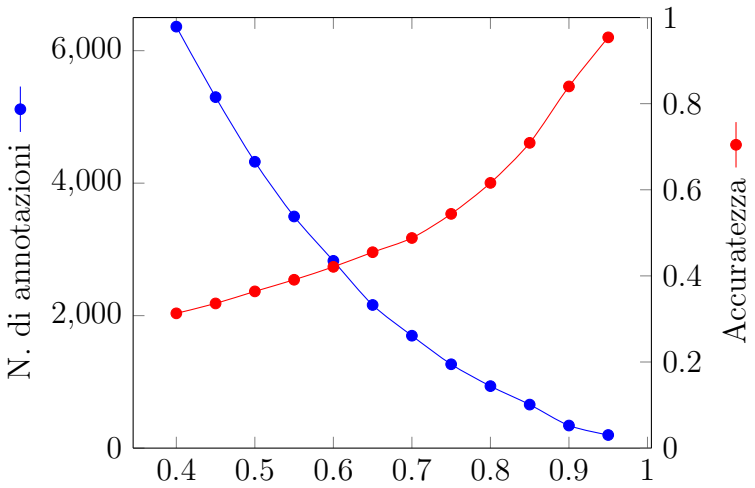
Ovviamente anche qui nella colonna dell'autoencoder senza perturbazione, essendo gli stessi dati, con il restringimento del primo strato si ha un miglioramento in alcuni casi. Provando lo stesso esperimento anche con le matrici perturbate si ha comunque un miglioramento, ma a parità di modello l'accuratezza degli esperimenti con la matrice perturbata è sempre leggermente inferiore.

6.4 Approfondimento sul cambio di soglia

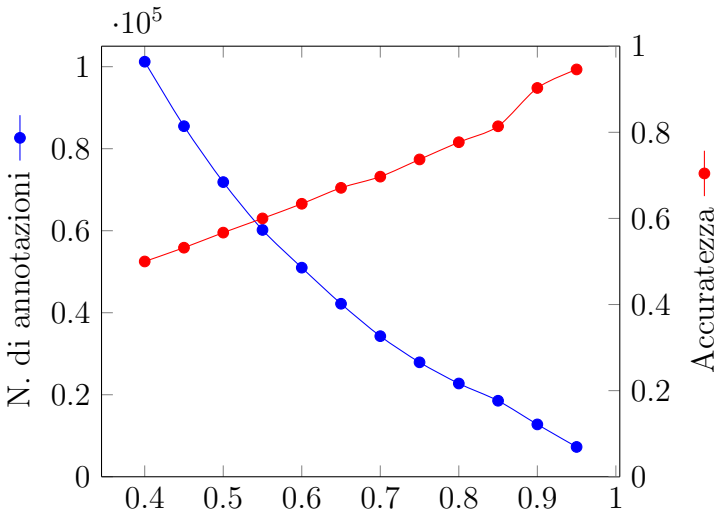
Per evidenziare come cambiano l'accuratezza della previsione e la quantità di annotazioni predette a seconda della soglia scelta per considerare una annotazione a 1 o meno, sono stati fatti dei grafici. Si può notare come all'aumentare della soglia la previsione diventa più accurata, ma diminuiscono

le annotazioni aggiunte. Per questo tipo di approfondimento si è scelto di usare le previsioni eseguite con l'autoencoder usando come Source la matrice dell'Homo Sapiens, mettendo come input la matrice del 2009 e come label quella del 2013. L'organismo Target invece cambia in ogni grafico.

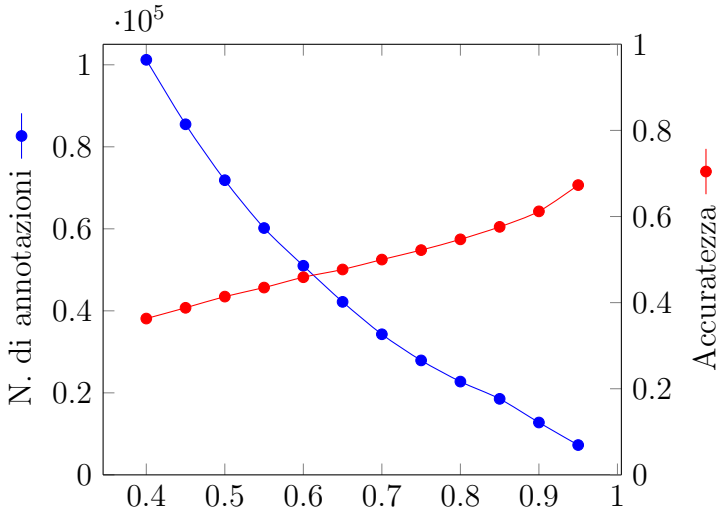
Target: Bos Taurus



Target: Mus Musculus



Target: Rattus Norvegicus



6.5 Approfondimento variazione M

La variabile M è stata usata come soglia per eliminare i geni con poche annotazioni. Generalmente è stata settata a 5, ma per evidenziare come cambiano i risultati al variare di questa variabile, sono stati fatti esperimenti sostituendola con altri valori. Come per l'approfondimento sulla soglia di probabilità sono stati presi come riferimento gli esperimenti eseguiti con l'autoencoder che usano come Source l'Homo Sapiens. Ecco la tabella con i risultati.

Target	M=1		M=5		M=10		M=25	
	Acc.	N	Acc.	N	Acc.	N	Acc.	N
Bos T.	0.572	1064	0.616	936	0.577	1045	0.451	1527
Mus M.	0.760	24189	0.777	22744	0.771	19873	0.644	25970
Rattus N.	0.541	31141	0.547	29060	0.535	30892	0.504	35511

Conclusioni

Sicuramente ci sono ancora molte cose su cui poter approfondire le ricerche e fare esperimenti, riuscendo forse anche ad ottenere dei risultati migliori di quelli ottenuti in questa tesi e nell'articolo usato come riferimento[6]. Questo è dovuto soprattutto alla grande quantità di valori che possono assumere gli iperparametri di una rete neurale e alle sue numerose strutture possibili.

Ciò non toglie che il Transfer Learning Inter-Organismo usato mediante i due metodi esposti in questa tesi è risultato comunque efficace e soddisfacente e conferma di essere uno dei percorsi da approfondire.

Ovviamente con l'avanzare degli anni e l'accrescere delle innovazioni sia in ambito informatico che in quello genomico, si potranno fare sempre più scoperte genomiche grazie all'aiuto di questi sistemi di previsione che potranno supportare gli scienziati negli esperimenti biologici offrendo loro una specie di guida. Questo è ciò che viene dimostrato da questa tesi e da tutti gli altri articoli che parlano di questo argomento.

Appendice A

Directory del progetto

All'interno del progetto su Overleaf è stato caricato tutto il necessario per ripetere gli esperimenti eseguiti ed ottenere gli stessi risultati esposti in questa tesi, sia codice che dati. Non sono stati inseriti tutti gli esperimenti, ma solo quelli che hanno ottenuto i risultati migliori.

In questa appendice spiegherò come sono organizzati all'interno del directory project e l'utilità di ogni file. Prima di tutto si divide in due cartelle:

- **Autoencoder**
- **NN x GO Term**

A.1 Autoencoder

All'interno di questa cartella, come si può immaginare, ci sono gli esperimenti eseguiti con l'autoencoder. Qui ci ritroveremo davanti ad altre 4 cartelle e a un file compresso in formato zip. Il file zip, chiamato DataAllTerm.zip, contiene all'interno tutte le matrici di ogni organismo preso in esame nella tesi. Queste matrici sono già state elaborate come spiegato nel capitolo 5.2.1 in modo da essere già utilizzabili dalle reti neurali.

Invece all'interno delle cartelle ci sono gli esperimenti, che sono suddivisi da esse secondo l'anno delle matrici usate nell'allenamento delle reti neurali.

Nel nome di ogni cartella ci sono due anni, suddivisi da un trattino: il primo anno è quello delle matrici usate come input per la rete neurale, mentre il secondo è quello delle matrici usate come label.

Una di queste cartelle nella prima parte del nome, oltre ad avere l'anno, c'è scritto anche "Pert.": questa è l'abbreviazione di "perturbate", quindi gli esperimenti all'interno useranno come input le matrici del 2009 perturbate.

All'interno di ogni cartella ci sono i notebook jupyter in cui sono state costruite, allenate e usate le reti neurali. Ogni notebook ha come nome le iniziali dell'organismo che usa per l'addestramento della sua rete neurale che poi verrà usata per predire le annotazioni di ogni organismo preso in esame. Le iniziali sono:

- **HS**, corrisponde a Homo Sapiens;
- **MM**, corrisponde a Mus Musculus;
- **BT**, corrisponde a Bos Taurus;

All'interno della cartella "2009 - 2013" sono presenti anche gli esperimenti per evidenziare le differenze nei risultati al variare della M, cioè di quella soglia posta per eliminare dai dati per l'addestramento della rete i geni con un numero di annotazioni inferiore ad essa, esposta nel capitolo 6.5. I notebook che fanno questo tipo di esperimenti sono quelli con "MX", dove X corrisponde al numero messo come soglia, nella parte finale del nome.

Sempre all'interno della stessa cartella, nel notebook "HS-all", è stata evidenziata la differenza dei risultati al variare dell'altra soglia, cioè quella della probabilità mostrata nel capitolo 6.4.

A.2 NN x GO Term

Invece questa cartella contiene tutti gli esperimenti eseguiti usando l'altro metodo, cioè una rete neurale (Neural Network, deriva da qui la prima parte del nome NN) per ogni termine GO.

All'interno di questa cartella ci si ritrova davanti ad uno scenario simile a quello dell'autoencoder, cioè a tre cartelle contenenti nel nome gli anni delle matrici usate per l'addestramento. Ovviamente hanno anche la stessa logica espressa nel capitolo precedente.

Le tre cartelle però hanno un contenuto diverso rispetto a quelle dell'autoencoder: contengono delle ulteriori cartelle che dividono gli esperimenti a seconda di quale sia l'organismo Source e quale quello Target. I nomi sfruttano le iniziali espresse precedentemente: le prime iniziali sono quelle degli organismi usati come Source, cioè usati per l'addestramento della rete; le iniziali che terminano il nome invece sono quelle dell'organismo Target, cioè quello su cui è stata usata la rete neurale per prevedere le annotazioni.

All'interno di queste ultime cartelle ci possono essere 6 tipi di file:

- **DataXY.zip**, dove al posto di X ci sono le iniziali dell'organismo Source, mentre al posto della Y quelle dell'organismo Target. Contiene le matrici usate per gli esperimenti della propria cartella;

- **createData.ipynb**, notebook Jupyter usato per creare i dati presenti all'interno dello zip partendo dalle matrici usate per l'autoencoder. Non sempre sono presenti, perché i dati necessari possono essere già stati creati in altre cartelle e quindi in questi casi è bastato copiare le matrici;

- **paperAll.py**, script in python per la creazione, addestramento e uso delle reti neurali per poi salvare all'interno di una cartella appositamente creata la colonna predetta. All'interno è presente un ciclo che svolge la stessa cosa per ogni termine GO (colonna) della matrice. Salva ogni volta la colonna predetta in modo che se per qualche motivo si interrompe la sua esecuzione se si riesegue lo script riesce a riprendere dalla colonna a cui era arrivato;

- **all.sh**, script in bash per eseguire paperAll.py passandogli i parametri di cui ha bisogno la rete neurale, per cui la struttura di quest'ultima la si può vedere da questo file. La riga che esegue paperAll.py è ripetuta più volte in modo che, se per qualche motivo si interrompe l'esecuzione dello

script paperAll.py, questo script lo possa far ripartire senza l'intervento manuale di qualcuno;

- **mux.py**, script in python per raccogliere e unire in una unica matrice tutte le colonne predette create dallo script paperAll.py. In questo modo si crea la matrice con tutti i valori predetti;
- **predictResult.ipynb**, notebook Jupyter usato per analizzare la matrice predetta creata dall'esecuzione di paperAll.py e di mux.py.

Bibliografia

[1] *Libreria Keras*. URL: `https://keras.io/`

[2] A. Gulli, S. Pal. *Deep Learning with Keras*. Packt Publishing, April 2017

[3] *Enciclopedia Treccani*. URL: `http://www.treccani.it/`

[4] *Gene Ontology Consortium*. URL: `http://www.geneontology.org/`

[5] *EMBL-EBI Nucleotide Archive*. URL: `https://www.ebi.ac.uk/ena`

[6] G. Domeniconi, M. Masseroli, G. Moro, P. Pinoli. *Cross-organism learning method to discover new gene functionalities*, in "Elsevier", 2016. URL: `https://www.sciencedirect.com/science/article/pii/S0169260715003259?via%3Dihub`

[7] *AmiGO 2* URL: `http://amigo.geneontology.org/amigo`

[8] D. Sadava et al. *L'ereditarietà e l'evoluzione*. Zanichelli, 2010