UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA MAGISTRALE IN
BIOTECNOLOGIE MOLECOLARI E BIOINFORMATICA

Metodi di Ensemble Gerarchici per Ontologie Biologiche Strutturate secondo Grafi Diretti Aciclici

Relatore: Prof. Giorgio **VALENTINI**
Correlatore: Dr. Matteo **RE**

Tesi di Laurea di:
Marco **NOTARO**
Matricola 825751

Anno Accademico 2013/2014

Ai miei genitori,
perché in silenzio
ve lo avevo promesso

Per Aspera sic itur ad Astra
Seneca, *Hercules furens*, atto II, v.437

Indice

Abbreviazioni

AFP	predizione automatizzata della funzione proteica (Automated Function Prediction)
GFP	predizione della funzione genica (Gene Function Prediction)
PFP	predizione della funzione proteica (Protein Function Prediction)
DAG	grafo diretto aciclico (Directed Acyclic Graph)
GO	Gene Ontology
FunCat	Functional Catalogue
HPO	Human Phenotype Ontology
SVM	Support Vector Machine
HTD	hierarchical top-down
TPR	true path rule
TPR-TF	True Path Rule Threshold Free
TPR-T	True Path Rule with Threshold
TPR-W	Weighted True Path Rule
TPR-WT	Weighted True Path Rule with Threshold
AUC	area sotto la curva ROC (Area Under the ROC Curve)
PxR	precisione a fissati livelli di recall (Precision at Different Levels of Recall)

Riassunto

Le ontologie biologiche come la Gene Ontology (GO) o la Human Phenotype Ontology (HPO) consentono di sistematizzare in modo coerente le relazioni gerarchiche fra le classi in diversi problemi rilevanti nell'ambito della biologia computazionale, come la predizione della funzione delle proteine o delle associazioni fra geni e fenotipi patologici nell'uomo. In tale contesto i metodi "flat", che predicono le classi non tenendo conto delle relazioni gerarchiche dell'ontologia, possono portare a predizioni inconsistenti con la struttura dell'ontologia. Ad esempio, un classificatore "flat" potrebbe associare ad una data proteina la classe/termine GO "omodimerizzazione", ma non la classe "genitore" più generale "dimerizzazione" o ancora potrebbe annotare un gene umano con il termine HPO "difetto interatriale", ma non con la classe "anomalia del setto interatriale". Nel presente lavoro di tesi si sono proposti dei nuovi metodi di ensemble gerarchici in grado di fornire predizioni consistenti con la struttura gerarchica dell'ontologia, che da un lato estendono precedenti metodi originariamente sviluppati per ontologie strutturate ad albero ad ontologie strutturate secondo Grafi Diretti Aciclici (DAG, come ad es: la GO e la HPO) e dall'altro sono in grado di migliorare significativamente l'accuratezza delle predizioni dei metodi flat. I nuovi metodi di ensemble proposti (Hierarchical Top-Down per DAG – HTD-DAG e True Path Rule per DAG – TPR-DAG) sono stati implementati in una libreria software scritta in linguaggio R, ed applicati sia alla predizione della funzione delle proteine in organismi modello che a quella dei fenotipi patologici nell'uomo. Risultati sperimentali su scala genomica relativi a complesse ontologie DAG-strutturate, comprendenti migliaia di classi funzionali e decina di migliaia di esempi (proteine o geni), mostrano come gli algoritmi gerarchici proposti, migliorino significativamente le predizioni rispetto all'approccio *Flat*, soprattutto in termini di precisione/sensibilità, indipendentemente dalla scelta del tipo di classificatore utilizzato. Lo studio sulla predizione delle associazioni gene-fenotipo patologico si colloca nell'ambito di una collaborazione con l'Istituto di Genetica Umana della *Charité* di Berlino, ed ha portato alla pubblicazione di due articoli nell'ambito di conferenze internazionali rispettivamente di ambito bioinformatico (IWBBIO 2015 - 3^{rd} International Work-Conference on Bioinformatics and Biomedical Engineering) e di ambito machine learning (MCS 2015 - Multiple Classifier Systems), e si prevede l'estensione del lavoro con pubblicazione in riviste internazionali.

Introduzione

La predizione delle funzioni delle proteine tramite metodi computazionali e un problema rilevante nell'ambito della biologia computazionale [1]. In tale contesto l'analisi Gene Ontology (GO) e degli alberi del Functional Catalogue (FunCat) del MIPS di Monaco, tramite cui si strutturano le relazioni fra le classi funzionali di geni/proteine, sono di grande rilevanza. Inoltre per la classificazione strutturata delle classi funzionali è essenziale introdurre procedure automatiche per l'associazione dei geni alle classi funzionali e a diverse tipologie di dati biomolecolari. A tal proposito, nel corso degli ultimi anni, si sono sviluppati metodi ed algoritmi per la predizione di classi funzionali di geni/proteine, per effettuare il pre-processing e la normalizzazione di dati biomolecolari complessi e multi-view e per supportare la classificazione gerarchica delle funzioni proteiche basate sulle tassonomie della GO e di FunCat.
Sfortunatamente però la predizione della funzione proteica (PFP) è un problema di classificazione complesso multi-classe (le classi funzionali delle proteine sono dell'ordine delle centinaia o migliaia, a secondo dell'ontologia di riferimento considerata), multi-etichetta (una proteina può appartenere a più classi) e multi-path (le classi sono strutturate secondo alberi o grafi diretti aciclici) [2]. Più in generale i metodi di predizione automatica si presentano come un problema complesso da un punto di vista computazionale per molteplici ragioni. Le principali sono elencate di seguito.

(i) *Ampio numero di classi funzionali*: centinaia per il FunCat [3] e migliaia per la Gene Ontology (GO) [4].

(ii) *Le proteine possono essere annotate con più di una classe funzionale*: dal momento che una proteina può appartenere contemporaneamente a più di una classe funzionale, il problema di classificazione è multi-etichetta [2].

(iii) *Per ogni proteina sono disponibili sorgenti multiple di dati*: la sempre crescente quantità di dati prodotta da biotecnologie high-throughput ha condotto ad un incremento delle sorgenti di dati genomiche e proteomiche. Quindi per sfruttare tutte le informazioni disponibili per ogni proteina è necessario disporre di metodi di apprendimento automatico che sono in grado di integrare differenti sorgenti di dati [5]. Infatti poiché ogni sorgente di dati cattura solo alcune delle caratteristiche funzionali dei geni e dei prodotti genici, singole sorgenti di dati biomolecolari sono in genere predittive solo per alcune classi funzionali, mentre possono risultare totalmente non informative per altre. Per tale ragione l'integrazione di diverse sorgenti di dati è uno dei problemi più rilevanti in ambito bioinformatico.

(iv) *Le classi funzionali sono tra di loro correlate*: poiché le classi funzionali sono organizzate secondo una predefinita gerarchia, le annotazioni non possono essere indipendenti. In generale, le relazioni funzionali note (come ad esempio le tassonomie) sono utilizzate per aggiungere informazioni a priori agli algoritmi di apprendimento o per introdurre vincoli espliciti tra le classi [2].

(v) *Poche annotazioni "positive" per ogni classe*: generalmente le classi funzionali presentano un forte sbilanciamento tra esempi positivi e negativi, con il numero di annotazioni positive in netta minoranza rispetto a quelle negative [2].

(vi) *Non univocità nella definizione di esempi negativi*: dal momento che si dispongono solo di annotazioni positive, non è possibile dare una definizione in maniera univoca di esempi negativi (considerando tutte le specie presenti nella GO, il numero complessivo di annotazioni negative ammonta a circa 2500). Le annotazioni negative possono essere selezionate utilizzando diverse strategie [6].

(vii) *Diverso grado di affidabilità delle classi funzionali*: ogni classe viene associata ad un gene o ad una proteina in base ad uno determinato livello di affidabilità [2].

(viii) *Complessità e rumore di fondo*: solitamente i dati biomolecolari presentano un'elevata dimensionalità, possono essere strutturati (ad esempio secondo grafi) ed in genere sono soggetti a rumore [2].

Molti approcci computazionali, ed in particolare metodi di apprendimento automatico, sono stati proposti in letteratura per risolvere i problemi sopra menzionati, come ad esempio metodi basati sulla sequenza [7], metodi basati sulle reti [8], algoritmi output strutturati basati su kernel [9] e metodi di ensemble gerarchici [10]. Altri approcci invece si sono focalizzati principalmente sull'integrazione di sorgenti multiple di dati, dal momento che ogni tipo di dato genomico cattura solamente alcune caratteristiche dei geni da classificare e una specifica sorgente di dati può essere utile per apprendere una specifica classe funzionale ed essere totalmente irrilevante per altre [2]. In letteratura sono stati proposti molti tipi di approcci per affrontare questo argomento, come ad esempio metodi basati sull'integrazione di reti funzionali [11], metodi basati sulla fusione di kernel [12], metodi basati sull'integrazione di spazi vettoriali [13] e metodi di ensemble [14].
La maggior parte dei metodi computazionali proposti sono stati applicati ad organismi unicellulari modello come il lievito *S. cerevisiae* [15, 16, 12], ma recentemente molti di questi sono stati estesi anche ad organismi pluricellulari come ad esempio *M. musculus* o la pianta modello *A. thaliana* [17, 18, 19, 20, 21, 22].
Il presente lavoro di tesi è così strutturato. Nella sezione 2 si descrivono i metodi allo stato

dell'arte per la predizione della funzione delle proteine ponendo una particolare enfasi sui metodi di ensemble gerarchici. Nella sezione 3 si presentano gli algoritmi di ensemble gerarchici appositamente progettati per tassonomie strutturate secondo grafi diretti aciclici. Nella sezione 4 si illustra la fase sperimentale in cui i metodi di ensemble gerarchici sono stati applicati ad ontologie biologiche ed infine nella sezione 5 si discutono i risultati. Nell'appendice A si mostra il codice ad alto livello della funzione che implementa l'algoritmo HTD-DAG, nell'appendice B il codice ad alto livello delle varianti dell'algoritmo TPR-DAG e nell'appendice C alcune funzioni di utilità per processare ed analizzare grafi.

Metodi Computazionali per la Predizione della Funzione Proteica

In letteratura sono stati proposti numerosi metodi computazionali per risolvere i problemi legati alla predizione automatizzata della funzione proteica (AFP) [2]. Alcuni di questi prendono in considerazione le predizioni di un insieme relativamente ristretto di classi funzionali [12, 23, 24], mentre altri estendono le predizioni ad un insieme più ampio di classi funzionali, utilizzando le Support Vector Machine (SVM) e la programmazione semi-definita [12], le reti neurali artificiali [25], le reti funzionali [11, 26] o le reti Bayesiane [23]. Altri metodi, servendosi del modello della regressione logistica [18] o più semplicemente di operatori algebrici [19], combinano le reti a connettività funzionale con metodi di apprendimento automatico. Altri migliorano le annotazioni della GO estraendo le relazioni semantiche esistenti tra geni e funzioni [27]. Infine altri metodi ancora adottano un approccio ensemble [28] che considera l'intrinseca natura gerarchica delle classi funzionali [29, 30, 31, 32].
I metodi computazioni che risolvono i problemi legati a AFP, principalmente basati su metodi di apprendimento automatico supervisionati o semi-supervisionati, possono essere schematicamente raggruppati nelle seguenti quattro famiglie:

1. metodi basati sulla sequenza;
2. metodi basati sulle reti;
3. metodi kernel per spazi di output strutturati;
4. metodi di ensemble gerarchici.

Tuttavia questo raggruppamento non è né esauriente e nemmeno esatto, nel senso che alcuni metodi possono anche non appartenere a nessuno dei gruppi sopra citati, mentre altri possono appartenere a più di un gruppo [2].
Di seguito verranno discusse le caratteristiche salienti di ognuno dei quattro gruppi, soffermandosi dettagliatamente sui metodi di ensemble gerarchici dal momento che rappresentano l'argomento principale di questo lavoro di tesi.

2.1 Metodi Basati sulla Sequenza

Gli algoritmi basati sull'allineamento di sequenza rappresentano il primo tentativo computazionale per predire la funzione di una proteina [33, 34]. L'assunzione sulla quale si

basano questi algoritmi è la seguente: " *se due sequenze sono molto simili tra di loro, allora molto probabilmente avranno la stessa funzione*". Tuttavia, anche se è ben noto che la conservazione della struttura secondaria e terziaria sono più strettamente collegate con la funzione proteica, gli algoritmi basati sull'allineamento di sequenza sono ancora tutt'oggi utilizzati come metodo standard nell'assegnare funzioni a proteine in organismi il cui genoma è stato appena sequenziato [7, 35]. Ovviamente, in questo contesto, l'integrazione dei metodi di predizione basati sulla sequenza con quelli basati sulla struttura, ha prodotto risultati ancora più incoraggianti [2, 36].
Anche se la maggior parte del lavoro di ricerca per la progettazione e lo sviluppo di approcci per risolvere i problemi legati a AFP si concentra sui metodi di apprendimento automatico, vale la pena evidenziare che nella "CAFA 2011 *challenge*" [22] uno dei metodi con le performance più elevate è proprio un algoritmo basato sull'allineamento di sequenza [37]. Infatti, quando l'unica informazione disponibile è rappresentata da una "grezza" sequenza nucleotidica o aminoacidica, i metodi basati sull'allineamento di sequenza possono essere competitivi con i metodi allo stato dell'arte di apprendimento automatico, semplicemente sfruttando la similarità di sequenza per giustificare l'inferenza di omologia [38].

2.2 Metodi Basati sulle Reti

Generalmente questi metodi rappresentano i dati come un grafo indiretto $G = (V, E)$, dove i nodi $v \in V$ corrispondo ai geni o ai prodotti genici e gli archi $e \in E$ sono pesati in base alle evidenze di co-funzionalità mostrate nei dati [39, 40]. Questi algoritmi trasferiscono le annotazioni da nodi precedentemente annotati a nodi non annotati semplicemente sfruttando le "relazioni di prossimità" tra nodi connessi [2]. Fondamentalmente questi approcci si basano sugli algoritmi di *label propagation* che predicono le classi di nodi non annotati senza utilizzare un modello predittivo globale [11, 20, 23].
Svariate strategie per apprendere i nodi non etichettati sono state esplorate grazie agli algoritmi di *label propagation*, che, esplorando la topologia del grafo, sono in grado di propagare le etichette delle proteine annotate [2]. Ne sono un esempio i metodi basati sulle reti di Hopfield [26, 41, 42], i metodi basati sul campo aleatorio di Markov [43, 44] e di Gauss [20, 24] e il semplice metodo *guilty-by-association* [45, 46], che si basa sull'assunzione che i nodi (proteine) connessi in una rete funzionale condividono le medesime funzioni [2].
Recentemente metodi basati su funzioni di scoring kernelizzate, capaci di utilizzare strategie di apprendimento automatico sia locali che globali, sono state applicate con successo a AFP oltre che al problema della "*Gene Disease Prioritization*" [47] (ordinamento di no-

di/geni non etichettati a partire da un nucleo, anche molto ristretto, di nodi positivi, cioè di geni la cui associazione con la malattia è nota a priori) e al nuovo paradigma in campo farmacologico noto come "*Drug Repositioning*" [48, 49] (riposizionamento di farmaci in classi terapeutiche differenti da quelle per cui erano stati inizialmente sviluppati).

2.3 Metodi Kernel per Spazi di Output Strutturati

Dato uno spazio delle caratteristiche $\mathcal{X}$ ed uno spazio strutturato delle etichette $\mathcal{Y}$, l'obiettivo è apprendere una funzione $f : \mathcal{X} \to \mathcal{Y}$ mediante una funzione kernel k che computa la "compatibilità" di una data coppia input-output (x, y) [2, 50]:

$$k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R} \tag{2.1}$$

I metodi output strutturati ricavano l'etichetta $\hat{y}$ trovando il massimo di una funzione g che usa la funzione di joint kernel definita (2.1):

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} g(x, y) \tag{2.2}$$

Un esempio di metodo kernel output strutturato è rappresentato dal metodo GO*struct* [51]. Tale approccio è stato applicato con successo per predire i termini GO nella specie *M. musculus* ed in altri organismi modello [9]. Gli algoritmi a massimo margine sono stati applicati anche per predire le funzioni enzimatiche in strutture strutturate ad albero [52, 53].

2.4 Metodi di Ensemble Gerarchici

I metodi di ensemble sono una delle principali aree di ricerca dell'apprendimento automatico [28, 54, 55, 56]. Da un punto di vista generale i classificatori ensemble sono un insieme di "*learning machine*" che cooperano per risolvere un problema di classificazione (Fig. 2.1). I metodi di ensemble gerarchici considerano esplicitamente le relazioni gerarchiche tra i termini funzionali [29, 30, 57, 58, 59, 60] modificando le predizioni "flat" (predizioni che non dipendono dalla struttura gerarchica delle classi) e migliorando in modo significativo le annotazioni delle proteine in termini di accuratezza e consistenza [31].
Le predizioni computate con il metodo "flat" sono tra di loro indipendenti e di conseguenza un classificatore potrebbe assegnare ad una singola proteina un insieme di termini tra di loro inconsistenti [2]. Ad esempio un predittore che assegna ad una data proteina la classe "*legame di un nucleotide purinico*", ma non assegna al suo termine genitore la classe "*legame nucleotidico*" è chiaramente inconsistente e un biologo molecolare che cerca

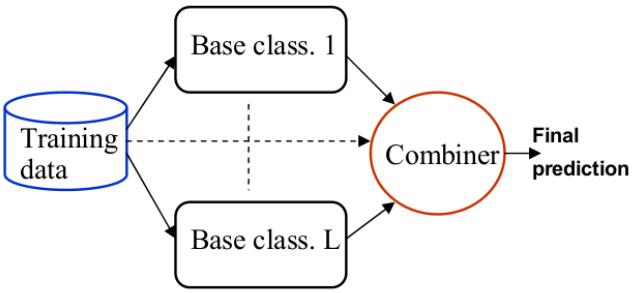
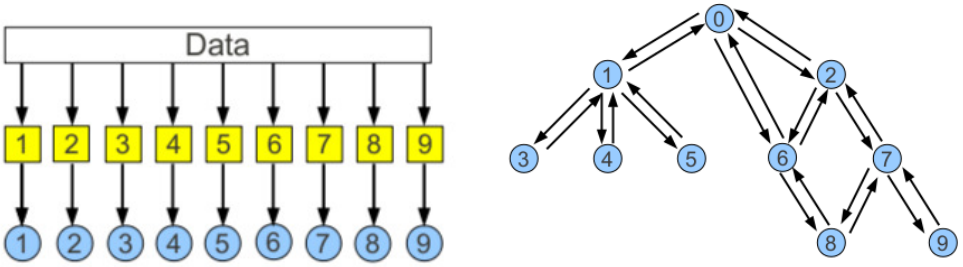


Figura 2.1: Classificatori Ensemble (Figura tratta da [2]).

di interpretare questi risultati probabilmente non avrebbe fiducia in nessuna delle due predizioni [31].
Una possibile soluzione a questo problema potrebbe essere quella di addestrare un classificatore in modo tale da produrre un insieme di predizioni per ogni termine dell'ontologia di riferimento e solo successivamente riconciliare le predizioni tenendo conto anche delle relazioni gerarchiche tra le classi dell'ontologia [2].
In figura 2.2 sono mostrate le due principali strategie di apprendimento di un metodo di ensemble gerarchico.



(a) (b)

Figura 2.2: Rappresentazione schematica dei due principali step dei metodi ensemble gerarchici. (a) Addestramento dei classificatori base (b) Propagazione top-down e/o bottom-up delle predizioni (Figure tratte da [2]).

Nel primo step viene applicato un algoritmo di apprendimento (il quadrato giallo in fig. 2.2 (a)) per addestrare i classificatori associati ad ogni termine (rappresentati dal numero 1 al numero 9). Nella fase successiva le relazioni gerarchiche tra le classi vengono sfruttate per combinare le predizioni ottenute dai classificatori (cerchi blu in fig. 2.2 (b)) [2]. Sempre in figura 2.2 (b) il nodo 0 indica il nodo root (cioè il nodo che si trova al livello più alto della gerarchia) ed il verso delle frecce, dal basso verso l'alto o dall'alto verso il basso, rappresenta la possibilità di combinare le predizioni dei classificatori figli e padri, a seconda che la strategia di apprendimento sia bottom-up o top-down [2]. É importante sottolineare che sono possibili sia combinazioni "locali" (e.g., la predizione del nodo

5 dipende unicamente dalla predizione del nodo 1) che "globali" (e.g., considerando la struttura del grafo le predizione del nodo 9 possono dipendere dalle predizioni di tutti gli altri classificatori, dal 1 al 8).
La tabella 2.1 mette a confronto le principali caratteristiche dei metodi di ensemble gerarchici che verranno argomentati nelle prossime sezioni. Le prime due colonne della tabella 2.1 riportano il nome ed il riferimento bibliografico del metodo, la terza se nella tassonomia sono permessi cammini multipli o singoli e la quarta se vengono considerati i percorsi parziali, cioè percorsi che non terminano con un nodo foglia. Le colonne successive si riferiscono alla struttura della classe (DAG o albero), all'utilizzo o meno di un approccio di classificazione "*cost-sensitive*" (cioè che considera lo sbilanciamento tra esempi positivi e negativi) e all'impiego di strategie per la selezione dei campioni negativi nella fase di addestramento. Infine, le ultime tre colonne, riassumono il tipo di classificatore utilizzato ("*spec*" indica che è permesso solamente uno specifico tipo di classificatore, mentre "*any*" indica che il metodo può utilizzare un qualsiasi tipo di classificatore), se il metodo migliora o meno le predizioni rispetto all'approccio "flat" e se la strategia utilizzata per il processing dei nodi è unicamente top-down ("TD") o se lo step top-down è preceduto da uno bottom-up ("TD&BP").

Tabella 2.1: Caratteristiche dei principali metodi di ensemble gerarchici per AFP (Dati tratti da [2]).

Methods	**Ref.**	**multi path**	**partial path**	**class struct**	**cost sens**	**sel neg**	**base learn**	**improves on flat**	**node process**
HMC-LMLP	[61, 62]	√	√	TREE			ANY	√	TD
HTD-MULTI	[63]		√	TREE			ANY	√	TD
HTD-PERLEV	[64]			TREE			SPEC	√	TD
HTD-NET	[57]	√	√	DAG			ANY	√	TD
TPR	[65, 66]	√	√	TREE		√	ANY	√	TD&BUP
TPR-W	[66]	√	√	TREE	√	√	ANY	√	TD&BUP
TPR-W WEIGHTED	[67]	√	√	TREE	√		ANY	√	TD&BUP
RECONC-HEURISTIC	[31]	√	√	DAG			ANY	√	TD
CASCADED LOG	[31]	√	√	DAG			ANY	√	TD
PROJECTION-BASED	[31]	√	√	DAG			ANY	√	TD&BUP
DECISION-TREE-ENS	[58]	√	√	DAG			SPEC	√	TD&BUP

Tuttavia, prima di incominciare a descrivere in modo dettagliato i vari metodi di ensemble gerarchici per AFP, è necessario introdurre alcune notazioni e definizioni di base.

2.4.1 Notazione

Un gene o un prodotto genico g può essere rappresentato attraverso un vettore $\boldsymbol{x} \in \mathbb{R}^d$ avente d componenti (e.g., diversi livelli di espressione genica in d differenti condizioni, similarità di sequenza con altri geni/proteine, presenza o assenza di interazioni proteiche,

genetiche o fisiche con altre proteine) [2]. D'ora in avanti, per semplicità e anche con una certa approssimazione, ci si riferirà ai geni e alle proteine allo stesso modo, anche se è risaputo che un gene potrebbe codificare più di una proteina.
Un gene g è annotato ad una o a più classi funzionali $C = \{c_1, c_2, \ldots, c_m\}$ organizzate secondo una foresta di alberi T (FunCat) o un grafo aciclico G (GO) [2]. Solitamente, per semplificare il processing dei dati, a T o a G viene aggiunto un nodo root (classe c_0) al quale appartengono tutte le altre classi funzionali [2]. Le annotazioni sono codificate mediante un vettore multi-etichetta $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0,1\}^m$, dove $g \in c_i \iff y_i = 1$ [2].
Sia nella tassonomia GO che in quella di FunCat le classi funzionali possono essere rappresentate mediante un grafo diretto, dove i nodi corrispondono alle classi e gli archi alle relazioni tra le classi [2]. D'ora in avanti il nodo corrispondente alla classe c_i verrà semplicemente denotato come i, l'insieme dei nodi figli di i come $child(i)$ e l'insieme dei nodi genitori di i come $par(i)$ [2].
Mentre nell'ontologia FunCat ogni nodo può avere solamente un genitore, dal momento che le classi sono organizzate secondo una foresta di alberi, nell'ontologia GO un nodo può avere più di un nodo genitore poiché le relazioni tra le classi sono strutturate secondo un grafo diretto aciclico (DAG) [2].
I metodi gerarchici ensemble addestrano un insieme di classificatori per ogni nodo della tassonomia T o G. Questi classificatori sono utilizzati per ricavare le stime $\hat{p}_i(g)$ delle probabilità $p_i(g) = \mathbb{P}\big(V_i = 1 \mid V_{\mathrm{par}(i)} = 1,\ g\big)\ \forall\ g$ e i, dove $(V_1, \ldots, V_m) \in \{0,1\}^m$ è il vettore di variabili aleatorie che modella le etichette non note del gene g e $V_{par(i)}$ denota le variabili casuali associate ai genitori del nodo i [2]. Si noti che $p_i(g)$ è la probabilità condizionata a $V_{par(i)} = 1$, cioè la probabilità che un dato gene è annotato con uno specifico termine i sapendo che il gene è già stato annotato con il termine genitore, rispettando così la true path rule (TPR).

2.4.2 Hierarchical Top-Down Ensemble

I metodi ensemble hierarchical top-down (HTD), sfruttando le sole relazioni gerarchiche top-down tra i termini funzionali (riferendosi alla figura 2.2 (b) le frecce verso il basso), propagano le decisioni dall'alto verso il basso nella struttura gerarchica (l'output di un nodo genitore influenza solamente l'output del nodo figlio) [2].
Il principio su cui si fonda l'algoritmo del metodo di ensemble gerarchico top-down è il seguente: "*per ogni gene g e partendo dai nodi figli della root del grafo G (primo livello), il classificatore associato ad ogni nodo $i \in G$ esegue il processo di classificazione, che si propaga in maniera ricorsiva ai nodi $j \in child(i)$, se e solo se il gene g è associato con la classe c_i; altrimenti, se $g \notin c_i$, il processo di classificazione termina al nodo i e tutti i*

discendenti del nodo i vengono settati a zero" [2].
Un classificatore probabilistico può essere addestrato per predire la classe c_i associata al nodo i della tassonomia gerarchica [2].
Il metodo ensemble HTD usa le probabilità $\hat{p}_i(g)$ di $\mathbb{P}\big(V_i = 1 \mid V_{\mathrm{par}(i)} = 1,\, g\big)$ per classificare un gene g come segue [2]:

$$\hat{y}_i = \begin{cases} \{\hat{p}_i(g) > \frac{1}{2}\} & \text{se } i \in \mathrm{root}(G) \\ \{\hat{p}_i(g) > \frac{1}{2}\} & \text{se } i \notin \mathrm{root}(G) \ \wedge\ \{\hat{p}_{\mathrm{par}(i)}(g) > \frac{1}{2}\} \\ 0 & \text{se } i \notin \mathrm{root}(G) \ \wedge\ \{\hat{p}_{\mathrm{par}(i)}(g) \leq \frac{1}{2}\} \end{cases} \tag{2.3}$$

dove $\{x\} = 1$ se $x > 0$ altrimenti $\{x\} = 0$ e $\hat{p}_{par(i)}$ è la probabilità predetta per il nodo genitore del termine i. É facile verificare che questa procedura garantisce che le multi-etichette predette $\hat{y} = (\hat{y}_i, \ldots, \hat{y}_m)$ sono consistenti con la gerarchia [2]. La medesima procedura top-down può essere applicata sostituendo i classificatori probabilistici con classificatori "*continui*" o "*discreti*", dopo aver leggermente modificato l'equazione 3.6 [2].
Nonostante la semplicità dei metodi HTD, in letteratura esistono molte prove che testimoniano la loro efficienza nel risolvere i problemi legati a AFP [66, 68]. Ad esempio Cerri e De Carvalho hanno sperimentato diverse varianti dei metodi ensemble HTD per AFP [61, 62, 68]. Il metodo HMC-LMLP (Hierarchical Multilabel Classification with Local Multilayer Perceptron), per ogni livello della gerarchia, addestra una rete locale MLP utilizzando il classico algoritmo di retro-propagazione [69]. L'output prodotto da MLP per il primo strato viene successivamente utilizzato come input per addestrare un secondo MLP che apprende le classi del secondo livello della gerarchia e così via finché non si raggiunge l'ultimo livello della gerarchia (Figura 2.3) [2].
Un gene viene annotato con una classe se l'output corrispondente nel MLP è maggiore rispetto ad una predefinita soglia. Nella fase di post-processing (secondo step della classificazione gerarchica) le predizioni inconsistenti, cioè classi predette senza la predizione delle loro super-classi, vengono rimosse [62]. In pratica l'algoritmo HMC-LMLP, al posto di usare un classificatore dicotomico per ogni nodo, utilizza un singolo percettrone multistrato e multi-etichetta ad ogni livello della gerarchia [2].
Un secondo approccio correlato al precedente adotta per ogni nodo del grafo un classificatore multi-classe (HTD-MULTI) al posto di un semplice classificatore binario e prova a trovare il percorso più probabile dal nodo root ai nodi foglia utilizzando semplici tecniche come la moltiplicazione o la somma delle probabilità stimate ad ogni nodo lungo il percorso [63]. Questo metodo è stato applicato al ramo del ciclo cellulare del lievito nella gerarchia di FunCat mostrando dei miglioramenti rispetto ai classici metodi HTD. Tuttavia questo approccio riesce a predire solo le classi che appartengono al "percorso più probabile" senza tenere conto delle annotazioni che sono implicate in cammini multipli o

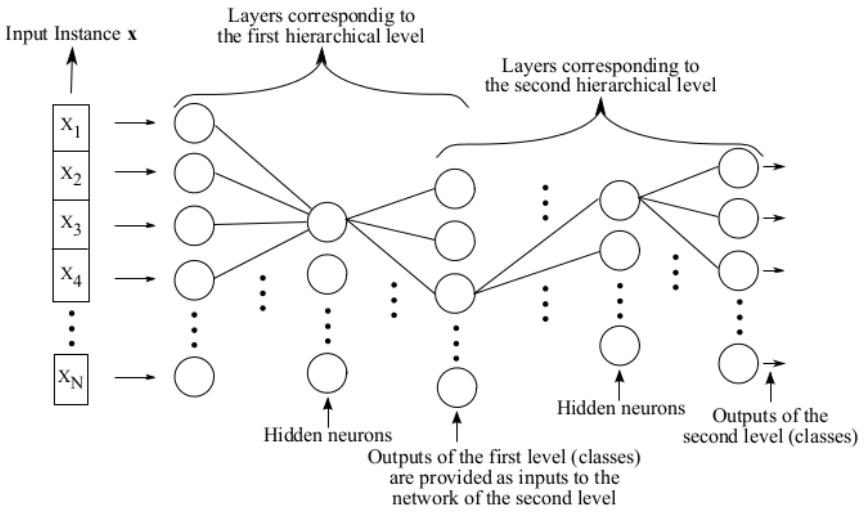


Figura 2.3: HMC-LP: Gli output del percettrone multistrato responsabili delle predizioni del primo livello diventano gli input di un altro MLP che apprende le classi del secondo livello della gerarchia (Figura tratta da [62]).

parziali [2].

Un'altra variante che utilizza un classificatore multiclasse al posto di uno dicotomico è stato proposto da Paes et.all [64]. In questo approccio un classificatore locale multiclasse per livelli (HTD-PERLEV) viene addestrato per distinguere tra le classi di uno specifico livello della gerarchia e due diverse strategie vengono utilizzate per rimuovere le inconsistenze [2]. Questo metodo è stato applicato per classificare gerarchicamente gli enzimi della tassonomia EC, ma sfortunatamente anche questo algoritmo non riesce a predire le etichette che si trovano lungo cammini parziali o multipli [2]. Un altro interessante approccio gerarchico top-down proposto dallo stesso autore è HMC-LP (Hierarchical Multilabel Classification Label Powerset), una variante gerarchia del metodo multi-etichetta non gerarchico *label-powerset* [70]. Questo metodo, in accordo con l'approccio *label-powerset*, si basa sulla combinazione delle etichette, mediante la quale tutte le classi annottate ad un campione (gene) vengono combinate in un'unica classe [2]. Questo processo viene ripetuto per ogni gene e per ogni livello della gerarchia. In tal modo il problema delle multi-etichette vine trasformato in un problema a singola etichetta [2]. L'approccio top-down viene applicato sia nella fase di addestramento che nella fase di test e alla fine del processo di classificazione le classi originali possono essere facilmente ricostrutite [61]. Questo approccio è stato utilizzato per predire la funzione genica del lievito impiegando dieci diversi dataset e la tassonomia FunCat [61].

Gli algoritmi top-down possono essere applicati anche ai metodi basati sulle reti (HTD-NET). Per esempio Jiang at.all [57] hanno proposto un modello probabilistico che combina i dati di interazione proteina-proteina con la struttura gerarchica della GO per predire le

classi funzionali che obbediscono alla TPR settando i discendenti di un nodo come negativi ogni volta che quel nodo è negativo [2]. Più precisamente gli autori calcolano una probabilità condizionale locale gerarchica, nel senso che, per ogni termine GO tranne il nodo root, solamente i nodi genitori influenzano l'etichettatura del nodo figlio [2]. Questa probabilità è calcolata assumendo che l'etichettatura di un gene è indipendente da ogni altro gene data quella dei suoi nodi vicini (una sorta di proprietà di Markov per le reti funzionali di interazione genica) e assumendo anche una distribuzione binomiale per il numero dei nodi vicini etichettati con i termini figli rispetto a quelli etichettati con le classi genitori [2]. Queste assunzioni sono stringenti ma necessarie per rendere il modello facile da controllare. Segue il calcolo della probabilità condizionale gerarchia globale applicando in modo ricorsivo a tutti i nodi predecessori la probabilità condizionale gerarchica locale precedentemente calcolata [2]. Più precisamente assumendo che $\mathbb{P}(\hat{y}_i = 1 \mid g,\ N(g))$, cioè la probabilità che un gene g è annotato con un nodo i, dato lo stato delle annotazioni dei suoi nodi vicini $N(g)$ nella rete funzionale, la probabilità condizionale gerarchica globale viene fattorizzata in accordo con il grafo GO come segue [2]:

$$Pr(\hat{y}_i = 1 \mid g,\ N(g) = \prod_{j \in anc(i)} \mathbb{P}\big(\hat{y}_j = 1 \mid \hat{y}_{par(i)} = 1,\ N_{loc}(g)\big) \tag{2.4}$$

dove $N_{loc}(g)$ rappresenta l'informazione gerarchica locale dei nodi vicini sulla coppia dei termini GO padre-figlio $par(j)$ e j [57]. Questo approccio garantisce di produrre assegnamenti consistenti con la gerarchia senza avere bisogno dello step di post-processing [2].

2.4.3 True Path Rule Hierarchcial Ensemble

Questi metodi di ensemble considerano sia le relazioni funzionali padre-figlio che quelle figlio-padre, ossia sfruttano allo stesso tempo sia le relazioni che vanno dall'alto verso il basso che quelle che vanno dal basso verso l'alto della struttura gerarchica (Fig. 2.2 (b)). Il TPR ensemble [65, 66] è un algoritmo basato sulla logica di annotazione che governa le principali ontologie pubbliche per la classificazione funzionale dei geni GO e FunCat. La regola fondamentale che garantisce la consistenza delle annotazioni dei geni è nota come *true path rule* e può essere riassunta come segue[71]: "*l'annotazione di un gene per una classe nella gerarchia è automaticamente trasferita a tutte le classi progenitrici mentre un gene non annotato per una determinata classe non può essere annotato in nessuno dei discendenti della classe*".
Se si considerano i nodi genitori di un dato nodo i, un classificatore che rispetta la *true path rule* deve obbedire alle seguenti regole:

$$\begin{cases} y_i = 1 & \Rightarrow \quad y_{\mathrm{par}(i)} = 1 \\ y_i = 0 & \not\Rightarrow \quad y_{\mathrm{par}(i)} = 0. \end{cases} \tag{2.5}$$

Considerando invece i nodi figli di un dato nodo i, un classificatore che rispetta la *true path rule* deve obbedire alle seguenti regole:

$$\begin{cases} y_i = 1 & \not\Rightarrow \quad y_{\text{child}(i)} = 1 \\ y_i = 0 & \Rightarrow \quad y_{\text{child}(i)} = 0. \end{cases} \tag{2.6}$$

Dalle equazioni di cui sopra (eq. 2.5 e 2.6), ne consegue che la struttura dell'ensemble gerarchico è percorsa da due flussi asimmetrici di informazione (fig. 2.4): le predizioni positive influenzano in maniera ricorsiva le predizioni dei classificatori associati ai nodi ancestori (eq. 2.5) mentre le predizioni negative influenzano le predizioni dei classificatori associati ai nodi discendenti della classe considerata (eq. 2.6).

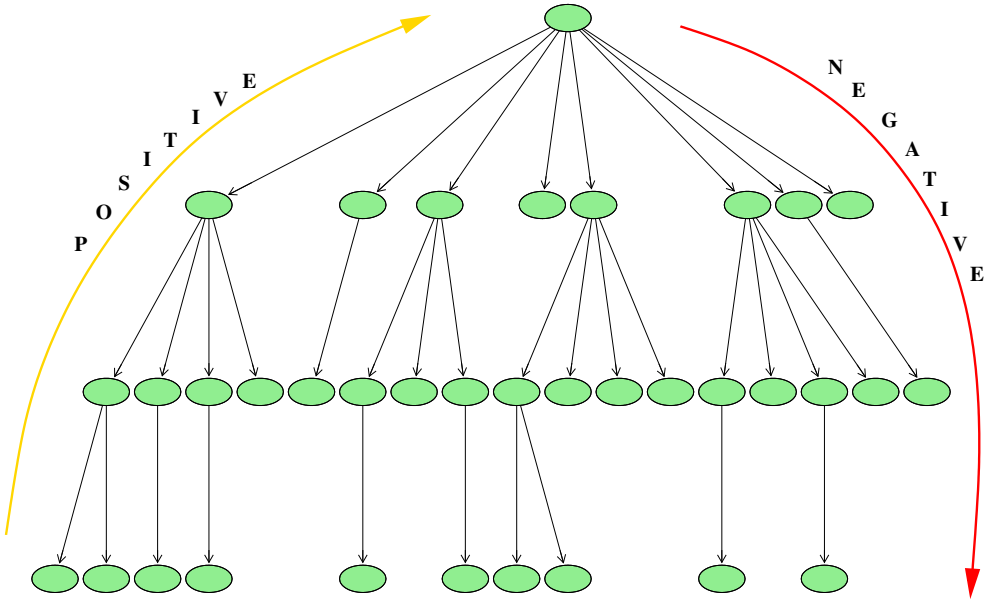


Figura 2.4: I due flussi di informazione asimmetrici che percorrono la struttura di un ensemble gerarchico (Figura tratta da [2]).

L'algoritmo TPR mette insieme le predizioni computate dal classificatore per ogni nodo per realizzare un ensemble gerarchico che obbedisce alla *true path rule*.
I principi su cui si fonda l'algoritmo TPR ensemble possono essere così riassunti:

(1) nella fase di addestramento, per ogni nodo della gerarchia, un algoritmo di apprendimento (e.g. percettrone multistrato o SVM) fornisce un classificatore per le associate classi funzionali [2];

(2) nella fase di valutazione, i classificatori addestrati, associati ad ogni nodo/classe del grafo, forniscono una decisione locale riguardo all'assegnazione di un dato esempio (gene/proteina) ad una data classe [2];

(3) le decisioni "positive", cioè annotazioni ad una specifica classe funzionale, si propagano dal basso verso l'alto della struttura gerarchica, influenzando le decisioni dei nodi genitori ed in modo ricorsivo quelle dei nodi predecessori. Al contrario le decisioni "negative", non propagandosi dal basso verso l'alto, non possono influenzare le decisioni dei nodi genitori [2];

(4) le predizioni "negative" per un dato nodo si propagano ai nodi discendenti per preservare la consistenza della gerarchia in accordo con la *true path rule*. Al contrario le decisioni "positive" non possono influenzare le decisioni dei nodi figli [2].

L'ensemble combina le predizioni locali computate dal classificatore sia con le decisioni positive che vengono dal basso della gerarchia che con le decisioni negative che provengono dai livelli più alti della gerarchia [2]. Più precisamente i classificatori stimano le probabilità locali $\hat{p}_i(g)$ che un dato gene g appartenga alla classe θ_i, ma il "core" dell'algoritmo è rappresentato dalla fase di valutazione, dove l'ensemble fornisce una stima della probabilità globale "consensus" $\overline{p}_i(g)$ [2].
Vale la pena notare che la probabilità $\overline{p}_i(g)$ può essere sostituita da uno score indicante la verosimiglianza che un gene o un prodotto genico g possa appartenere alla classe funzionale i [2].
Sia $\phi_i(g)$ l'insieme dei figli del nodo i per il quale si ha una predizione positiva per una dato gene g:

$$\phi_i(g) = \left\{ j \,:\, j \in \text{child}(i),\, \hat{y}_j = 1 \right\} \tag{2.7}$$

La probabilità globale "consensus" $\overline{p}_i(g)$ di un ensemble dipende sia dalla predizione locale $\hat{p}_i(g)$ che dalla predizione dei nodi appartenenti a $\phi_i(g)$:

$$\overline{p}_i(g) = \frac{1}{1 + |\phi_i(g)|} \left(\hat{p}_i(g) + \sum_{j \in \phi_i(g)} \overline{p}_j(g) \right) \tag{2.8}$$

La decisione $\hat{y}_i(g)$ al nodo/classe i viene settata a 1 se $\overline{p}_i(g) > t$, altrimenti se $\overline{p}_i(g) < t$, la predizione $\hat{y}_i(g)$ viene settata a 0 [2]. Dal momento che $\overline{p}_i(g)$ rappresenta una probabilità, potrebbe essere ragionevole settare la soglia $t = 0.5$. Così un gene g viene annotato ad una classe i se ha una probabilità di appartenenza maggiore del 50%. Si ricorda che solo i nodi figli per i quali si ha una predizione positiva possono influenzare le predizioni dei nodi genitori.
Ai nodi foglia (i.e., nodi che non presentano archi uscenti), la somma nell'equazione 2.8 diventa nulla e l'eq. 2.8 diviene $\overline{p}_i(g) = \hat{p}_i(g)$ [2]. In questo modo le predizioni positive si propagano dal basso verso l'alto, mentre le decisioni negative si propagano verso i nodi discendenti nel momento in cui per un dato nodo $\hat{y}_i(g) = 0$ [2]. Si notino ancora una volta

i due flussi d'informazione asimmetrici che percorrono la struttura dell'ensemble gerarchico (fig. 2.4).

Tuttavia nell'algoritmo TPR ensemble non vi è un modo esplicito per bilanciare le predizioni locali $\hat{p}_i(g)$ al nodo i con le predizioni positive provenienti dalla sua discendenza (eq. 2.8). Per bilanciare le predizioni locali con le predizioni positive provenienti dall'ensemble, è necessario modulare esplicitamente le relazioni tra i predittori locali e globali [2]. A tal scopo è possibile introdurre un peso $w, 0 \leq w \leq 1$, tale che se $w = 1$ la decisione al nodo i dipende solamente dal predittore locale, altrimenti, la predizione è condivisa in modo proporzionato a w e $1 - w$ tra il predittore locale e l'insieme dei suoi nodi figli [2]:

$$\hat{p} = w\,\hat{p}_i + \frac{1-w}{|\phi_i|}\sum_{j\in\phi_i}\overline{p}_j \tag{2.9}$$

Questa variante dell'algoritmo TPR prende il nome di algoritmo ensemble gerarchico Weighted True Path Rule (TPR-W). Uno specifico vantaggio di TPR-W è la capacità di modulare le caratteristiche della precisione e della recall dell'ensemble gerarchico sintonizzando il parametro w (eq. 2.9). Più precisamente per $w \to 0$ il peso del classificatore locale genitore è piccolo e le decisioni dell'ensemble gerarchico dipendono principalmente dalle predizioni positive dei nodi discendenti [2]. Al contrario per alti valori di w ($w \to 1$) il peso del predittore genitore locale è alto e il TPR-W assume un comportamento simile a HTD [2] (fare riferimento alla sezione 2.4.2). Quindi piccoli valori di w incrementano la recall, mentre grandi valori w migliorano la precisione dell'algoritmo ensemble TPR-W [66]. Recentemente, Chen e Hu hanno proposto una nuova strategia (TPR-W *weighted*) per oltrepassare i limiti dell'esplorazione gerarchica bottom-up dell'algoritmo TPR-W "*vanilla*" [2]. Infatti, per le classi che si trovano ai livelli più bassi della gerarchia, le performance del classificatore sono abbastanza scarse a causa sia dei dati rumorosi che del basso numero di annotazioni disponibili [2]. Più precisamente, nella versione "non pesata" dell'algoritmo TPR ensemble le probabilità $\overline{p}_j$ calcolate dai nodi figli del nodo i (eq. 2.9) contribuiscono in ugual modo alla probabilità $\overline{p}_i$ calcolata dall'ensemble al nodo i, indipendentemente dalla accuratezza delle predizioni fatte dai suoi classificatori figli [2]. Questo meccanismo "non pesato" potrebbe provocare una propagazione "a cascata" dell'errore nella gerarchia. Per esempio, una scarsa performance del classificatore figlio potrebbe, con un'alta probabilità, predire un campione negativo come positivo (campione predetto come falso positivo) e questo errore potrebbe propagarsi ai nodi genitori e in modo ricorsivo ai nodi predecessori [2]. Per attenuare questa possibile propagazione bottom-up degli errori, Chen e Hu hanno implementato una variante dell'algoritmo TPR ensemble che prende in considerazione la performance di ogni classificatore figlio semplicemente aggiungendo all'equazione 2.9

un ulteriore peso ν_j:

$$\overline{p}_i = w\,\hat{p}_i + \frac{1-w}{|\phi_i|}\sum_{j\in\phi_i}\nu_j \cdot \overline{p}_j \tag{2.10}$$

dove ν_j viene calcolato sulla base di alcune metriche di accuratezza A_j (e.g., F-score) valutate per i classificatori figli associati al nodo j:

$$\nu_j = \frac{A_j}{\sum_{k\in\phi_i} A_k} \tag{2.11}$$

In questo modo il contributo dei classificatori "mediocri" viene ridotto, mentre i classificatori "buoni" avranno un peso maggiore nel calcolo di $\overline{p}_i$ (eq. 2.10) [2].
Lavori sperimentali che hanno utilizzato il sottoalbero "Protein Fate" della tassonomia FunCat dell'organismo modello lievito, hanno mostrato che l'approccio TPR-W *weighted* ha migliorato le predizioni rispetto alla strategia gerarchica TPR-W "vanilla" [67].

2.4.4 Metodi di Riconciliazione

Come ribadito più volte nei paragrafi precedenti, i metodi di ensemble gerarchici sono principalmente metodi a due step. Il primo fornisce le predizioni per le singole classi, mentre il secondo combina le predizioni prendendo in considerazione le relazioni funzionali tra i vari termini GO [2]. Noble at al. hanno nominato questa procedura generale col termine di *metodi di riconciliazione* [31]. Gli autori hanno quindi proposto dei metodi per calibrare e combinare le predizioni indipendenti al fine di ottenere un insieme di predizioni probabilistiche consistenti, cioè predizioni la cui confidenza (e.g., probabilità a posteriori) aumenta man mano che ci si sposta dai termini più specifici (i.e., foglie) a quelli più generali dell'ontologia GO [31]. I valori di confidenza associati con le predizioni possono essere interpretati come la probabilità che una proteina possegga una certa funzione basandosi sulle informazioni fornite dai dati [2].

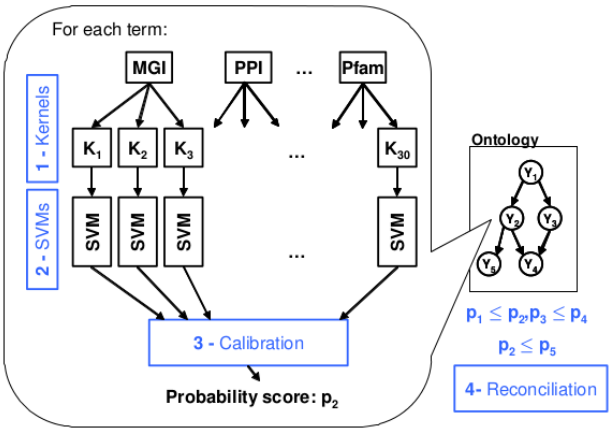


Figura 2.5: Schema generale dei metodi di riconciliazione (Figura tratta da [31]).

La strategia di riconciliazione proposta da Noble et al. può essere complessivamente riassunta nei seguenti quattro step (Figura 2.5):

(1) *Funzioni Kernel*: dai dati disponibili sono stati calcolati un insieme di funzioni kernel. È possibile scegliere una specifica funzione kernel per ogni sorgente di dati (e.g., kernel di diffusione per dati di interazione proteina-proteina [72], kernel lineari o Gaussiani per dati di espressione genica o kernel che operano sulle stringhe per dati di sequenza [73]) oppure è possibile costruire kernel multipli per stessi tipi di dati [31].

(2) *Apprendimento mediante SVM*: utilizzando le funzioni kernel selezionate allo step precedente, come classificatori sono stati utilizzati delle SVMs [31]. È importante sottolineare che mentre gli autori nei loro esperimenti hanno utilizzato delle SVMs, un qualsiasi classificatore potrebbe essere adoperato al loro posto [2].

(3) *Calibrazione*: un approccio di regressione logistica è stato utilizzato per produrre output probabilistici individuali dal set di output prodotti dalle SVM corrispondenti ad un termine GO [31].

(4) *Riconciliazione*: l'obiettivo di questo step è proprio quello di riconciliare gli output prodotti dallo step precedente per generare predizioni consistenti con l'ontologia [31]. In altre parole tutte le probabilità assegnate ai nodi predecessori di un termine GO devono essere più grandi della probabilità assegnata a quel dato termine GO [2].

In pratica, mentre i primi tre step sono fondamentalmente sempre gli stessi per ogni metodo ensemble di riconciliazione, lo step determinante è il quarto (la fase di riconciliazione) e differenti algoritmi ensemble possono essere progettati per implementarlo [2].
Noble et al. hanno proposto ben undici differenti metodi ensemble per combinare le predizioni inconsistenti generate dal classificatore base [31]. Tali metodi possono essere schematicamente raggruppati nelle seguenti quattro classi [2]:

(1) metodi euristici;

(2) regressione logistica a cascata;

(3) metodi basati sulle reti Bayesiane;

(4) metodi basati sulle proiezioni.

Di seguito verranno illustrate le caratteristiche salienti di ognuna delle quattro classi in cui sono stati raggruppati gli undici metodi di riconciliazione.

2.4.4.1 Metodi Euristici

Questi approcci preservano la "*proprietà di riconciliazione*"

$$\forall i, j \in G, \quad (i,j) \in G \Rightarrow \hat{p}_i \geq \hat{p}_j \tag{2.12}$$

mediante una semplice modificazione euristica delle probabilità calcolate allo step tre dello schema generale dei metodi di riconciliazione (fig 2.5). I metodi euristici sono i seguenti:

1. **MAX**: riporta il più grande valore di regressione logistica del nodo i e di tutti i suoi discendenti $desc(i)$.

$$p_i = max_{j \in desc(i)} \hat{p_j} \tag{2.13}$$

2. **AND**: riporta il prodotto dei valori di regressione logistica del nodo i e di tutti i suoi ancestori $anc(i)$. In altre parole questo metodo calcola la probabilità assumendo che tutti i termini GO predecessori di un dato nodo i sono annotati e che tutte le predizioni sono indipendenti.

$$p_i = \prod_{j \in \text{anc}(i)} \hat{p_j} \tag{2.14}$$

3. **OR**: Calcola la probabilità assumendo che almeno uno dei discendenti dei termini GO di un dato nodo i è annotato e che tutte le predizioni sono indipendenti.

$$1 - p_i = \prod_{j \in \text{desc}(i)} (1 - \hat{p_j}) \tag{2.15}$$

2.4.4.2 Regressione Logistica

Al posto di modellare le probabilità condizionate per ogni termine GO, come richiesto dall'approccio Bayesiano, si può utilizzare la regressione logistica per modellare direttamente le probabilità a posteriori. Visto e considerato che modellare la densità di probabilità condizionata è in molti casi difficile, l'utilizzo della regressione logistica potrebbe essere una scelta ragionevole [2]. Noble et. al hanno integrato nelle impostazioni della regressione logistica le dipendenze gerarchiche tra i termini funzionali [31]. Assumendo che una variabile aleatoria $\boldsymbol{X}$, i cui valori rappresentano le caratteristiche del gene g di interesse, è associata al gene g e assumendo che $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x})$ fattorizza in accordo con il grafo GO, si ha:

$$\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{X} = \boldsymbol{x}) = \prod_i \mathbb{P}(Y_i = y_i | \forall j \in \text{par}(i)\ Y_j = y_j, X_i = x_i) \tag{2.16}$$

con $\mathbb{P}(Y_i = 1 \mid \forall_j \in par(i) Y_j = 0, X_i = x_i) = 0$. Gli autori hanno utilizzato la regressione logistica per stimare $\mathbb{P}(Y_i = 1 | \forall j \in \text{par}(i)\ Y_j = 1, X_i = x_i)$.

2.4.4.3 Metodi Basati sulle Reti Bayesiane

Gli autori hanno proposto quattro varianti che possono essere riassunte come segue [31]:

(i) BPAL: belief propagation con una probabilità asimmetrica laplaciana. L'ontologia GO è vista come un modello grafico con gli archi diretti dai termini più generali a quelli più specifici. La distribuzione di ogni SVM è modellata come una distribuzione asimmetrica laplaciana ed un algoritmo inferenziale variazionale che risolve un problema di ottimizzazione viene utilizzato per stimare le probabilità a posteriori dell'ensemble [31];

(ii) BPALf: approccio simile a BPAL ma con il verso degli archi invertiti, dai termini più specifici a quelli più generali [31];

(iii) BPLR: variante euristica di BPAL [31]

(iv) BPLRf: uguale a BPLR ma con il verso degli archi invertiti [31].

2.4.4.4 Metodi Basati sulle Proiezioni

Un diverso approccio è rappresentato dai metodi che usano direttamente i valori calibrati ottenuti dalla regressione logistica (step tre della strategia complessiva dei metodi di riconciliazione, fig. 2.5) per trovare l'insieme più vicino dei valori che sono consistenti con l'ontologia [31]. Questo approccio si riduce a risolvere un problema di ottimizzazione con costrizioni. Il principale contributo nel lavoro di Noble et al. [31] è rappresentato dall'introduzione di tecniche di riconciliazione basate sulla regressione isotonica [74] e sulla divergenza di Kullback-Leibler.
La *regressione isotonica* mira a trovare un insieme di probabilità marginali p_i vicine all'insieme dei valori calibrati $\hat{p}_i$ ottenuti dalla regressione logistica. La distanza euclidea è usata come misura di vicinanza. Quindi considerando che la "*proprietà di riconciliazione*" richiede che $p_i \geq p_j$ dove $(i,j) \in E$, questo approccio si riduce a risolvere il seguente problema di ottimizzazione quadratica:

$$\begin{aligned} &\min_{p_i, i \in I} && \sum_{i \in I} (p_i - \hat{p}_i)^2 \\ &\text{s.t.} && p_j \leq p_i, \ (i,j) \in E \end{aligned} \tag{2.17}$$

Questo è un classico problema di regressione isotonica che può essere risolto utilizzando un risolutore interno o un algoritmo di approssimazione se il numero di archi del grafo è troppo grande [75]. Poiché si sta discutendo di probabilità, una misura della distanza tra

funzioni di densità di probabilità $f(\boldsymbol{x})$ e $g(\boldsymbol{x})$ definite rispetto ad una variabile aleatoria $\boldsymbol{x}$ è rappresentata dalla divergenza di Kullback-Leibler $D_{f_{\boldsymbol{x}}||g_{\boldsymbol{x}}}$ [31]:

$$D_{f_{\boldsymbol{x}}||g_{\boldsymbol{x}}} = \int_{-\infty}^{\infty} f(\boldsymbol{x}) \log\left(\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}\right) d\boldsymbol{x} \tag{2.18}$$

Nel contesto dei metodi di riconciliazione è necessario considerare una versione discreta della divergenza di Kullback-Leibler, ottenendo il seguente problema di ottimizzazione [31]:

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & D_{\hat{\boldsymbol{p}}||\boldsymbol{p}} = \min_{p_i, i \in I} \sum_{i \in I} \hat{p}_i \log\left(\frac{\hat{p}_i}{p_i}\right) \\ \text{s.t.} \quad & p_j \leq p_i, \ (i,j) \in E \end{aligned} \tag{2.19}$$

L'algoritmo, in accordo con la divergenza di Kullback-Leibler e obbedendo alle costrizioni che governano la gerarchia sottostante l'ontologia, trova le probabilità più vicine alle probabilità $\hat{p}$ ottenute dalla regressione logistica e tali probabilità non possono aumentare man mano che si scende ai livelli più specifici dell'ontologia [2].
Tra gli undici metodi di riconciliazione utilizzati nel lavoro sperimentale di Obozinski et al. descritto in [2], il metodo che, tra le diverse metriche di valutazione utilizzate per termine, per ontologia e a diversi livelli di recall, ottiene frequentemente alti livelli di precisione è la regressione isotonica [2]. D'altra parte la regressione isotonica non è sempre il "metodo migliore" e un biologo molecolare, con in mente un preciso obiettivo, potrebbe scegliere di applicare anche un altro metodo di riconciliazione. Per esempio con un piccolo numero di classi generalmente i risultati migliori si ottengono con la proiezione di Kullback-Leibler, mentre se si considerano i risultati medi per ogni termine, i valori di precisione a fissati livelli di recall dei metodi euristici sono comparabili con quelli ottenuti dai metodi basati sulla proiezione e sono addirittura migliori rispetto a quelli ottenuti dai metodi basati sulle reti Bayesiane.

In generale questi metodi ensemble, applicati a circa 3000 termini GO dell'organismo modello *M.musculus*, hanno conseguito ottimi risultati nella predizione della funzione proteica, dimostrando ancora una volta che i metodi gerarchici multi-etichetta giocano un ruolo chiave nel migliorare le performance in PFP [31].

2.4.5 Metodi di Ensemble basati sugli Alberi di Decisione

Un'altra interessante linea di ricerca è rappresentata dai metodi gerarchici basati sugli alberi di decisione induttivi [76]. I primi tentativi per esplorare la struttura gerarchica delle ontologie funzionali per AFP utilizzavano differenti modelli di alberi di decisione per

ogni livello della gerarchia [77], oppure studiavano un modello modificato di albero di decisione in cui l'annotazione di un gene o di una proteina ad un nodo veniva propagato verso i nodi genitori estendendo l'albero di decisione C4.5 a problemi di classificazione multi-classe [2]. In questo quadro di lettura Schietgat et al. hanno mostrato che ensemble di alberi di decisione multi-etichetta gerarchici sono competitivi con i metodi di apprendimento statistici allo stato dell'arte per la predizione delle funzioni proteiche in gerarchie strutturate secondo DAG e utilizzando come organismi modello *S. cerevisiae*, *A. thaliana* e *M. musculus* [58]. Un ulteriore studio ha esaminato la compatibilità di differenti metodi di ensemble basati su alberi di raggruppamento predittivi, che includono sia metodi ensemble globali, che utilizzano insiemi di modelli predittivi ognuno in grado di predire l'intera struttura della gerarchia (cioè tutti i termini GO di un dato gene), che locali, che invece utilizzano un ensemble per ogni ramo della tassonomia [2].
Infine in letteratura sono state proposte anche delle strategie che, ispirandosi ai metodi basati sugli alberi di decisione e utilizzando l'algoritmo di ottimizzazione "colonia di formiche" (ACO, ant colony optimization), hanno portato alla scoperta di nuove regole di classificazione [78].

2.5 La Sola Classificazione Gerarchica non è Sufficiente

Molteplici lavori sperimentali presenti in letteratura hanno mostrato che nei problemi di predizione della funzione proteica è necessario considerare diverse problematiche legate all'apprendimento [1, 8, 22]. Infatti, anche se i metodi ensemble gerarchici sono fondamentali per migliorare l'accuratezza delle predizioni, la loro semplice applicazione non è sufficiente ad assicurare risultati allo stato dell'arte se allo stesso tempo non si considerano anche altre problematiche legate ad AFP [48]. In particolare è stata mostrata una cooperazione significativa tra la classificazione gerarchica, i metodi di integrazione di dati e tecniche "*cost-sensitive*" [79], evidenziando che i metodi di ensemble gerarchici dovrebbero essere pensati prendendo in considerazione anche diverse problematiche legate all'apprendimento, cruciali per risolvere i problemi legati a AFP [2].
I risultati pubblicati in letteratura della CAFA 2011 *challenge* hanno mostrato che l'integrazione dei dati assume un ruolo chiave nel migliorare le predizioni delle funzioni proteiche [22, 80, 81, 82, 83]. Infatti la sempre crescente quantità di dati biomolecolari prodotta da biotecnologie high-throughput ha marcato ancora di più l'importanza dell'integrazione dei dati per migliorare l'accuratezza delle predizioni in PFP [1].
È possibile suddividere i principali approcci per l'integrazione dei dati nei seguenti quattro gruppi [80]:

1. integrazione di spazi vettoriali;
2. integrazioni di reti di associazioni funzionali;
3. fusione di kernel;
4. metodi ensemble.

Di seguito verranno discusse le caratteristiche salienti di ognuno dei quattro gruppi.

2.5.1 Integrazione di Spazi Vettoriali

Questo approccio consiste nel concatenare dati rappresentati mediante stringhe per combinare differenti sorgenti di dati biomolecolari [84]. Per esempio Pavlidis et al. hanno concatenato diversi vettori, ognuno dei quali rappresentate una sorgente di dati genomici differente, al fine di ottenere un vettore più grande da utilizzare per addestrare una classica SVM [13]. Guan et al. invece hanno proposto un approccio simile al precedente, con l'unica differenza che ogni sorgente di dati è stata precedentemente normalizzata per tenere conto della distribuzione assunta dai dati in ogni singolo spazio vettoriale [85].

2.5.2 Integrazione di Reti di Associazioni Funzionali

Nelle reti di associazioni funzionali, si combinano grafi differenti al fine di ottenere una rete composita risultante [26, 11]. L'approccio più semplice adotta tecniche connettive [39], cioè aggiunge un arco quando in tutte le reti due geni sono tra di loro collegati o quando il collegamento tra i due geni è presente in almeno una rete funzionale [23].
Altri metodi invece ponderano in modo diverso ogni sorgente di dati utilizzando tecniche che vanno dai campi aleatori gaussiani [24] all'integrazione Naive-Bayes [86]. Altri metodi ancora fondono i dati prendendo in considerazione la gerarchia GO [87] o applicano tecniche basate sul metalinguaggio XML [88].

2.5.3 Fusioni di Kernel

In questi metodi si costruisce, separatamente per ogni sorgente di dati disponibile, una matrice di Gram utilizzando un apposito kernel, che rappresenta le relazioni di similarità tra i geni/prodotti genici [2]. In un secondo momento, sfruttando la proprietà di chiusura della somma o di altri operatori algebrici, le matrici di Gram vengono combinate per ottenere una matrice globale integrata "consesus" [2]. I metodi di fusione di kernel (e.g., somma delle matrici di Gram), sia con che senza ponderazione delle sorgenti di dati, sono state applicate con successo alla classificazione delle funzioni proteiche [89, 90, 91, 92].

2.5.4 Metodi di Ensemble

La fusione dei dati genomici può essere realizzata mediante un sistema di ensemble. I classificatori vengono dapprima addestrati su differenti "*punti di vista*" dei dati (dati multi-view) e, in un secondo momento, gli output dei singoli classificatori componenti vengono combinati [2]. Così ogni tipo di dato potrebbe catturare caratteristiche differenti e complementari degli oggetti da classificare e l'ensemble risultante potrebbe ottenere migliori capacità di predizione grazie alla diversità e all'anticorrelazione degli output dei classificatori [2].
Alcuni esempi di metodi ensemble per la combinazione di dati includono "*l'integrazione tardiva*" di kernel addestrati su differenti sorgenti di dati [13] e l'integrazione naive-Bayes [93] degli output delle SVMs addestrate con sorgenti multiple di dati [85].
Recentemente è stato mostrato che anche con metodi relativamente semplici, come la votazione a maggioranza [94, 95] o i templati di decisione, è possibile ottenere [96] risultati comparabili con lo stato dell'arte, sfruttando allo stesso tempo sia la modularità che la scalabilità che caratterizzano la maggior parte degli algoritmi di ensemble [14]. Infine, in un altro lavoro, si è mostrato che i metodi di ensemble sono in grado di tollerare anche relativamente elevati livelli di rumore nei dati, senza un significativo deterioramento delle prestazioni [97].

Metodi di Ensemble Gerarchici per Grafi Diretti Aciclici

I problemi di classificazione gerarchica sono caratterizzati da tassonomie strutturate secondo una gerarchia predefinita. La maggior parte dei metodi presenti in letteratura si focalizzano su tassonomie organizzate ad albero e solamente pochi su tassonomie strutturate secondo un DAG [98]. Nel contesto della predizione della funzione genica o proteica le classi funzionali sono connesse sia secondo una struttura ad albero (FunCat, Functional Categories [3]) che secondo un DAG (GO, Gene Ontology [4]).
Molteplici studi sperimentali mostrano che la predizione flat, cioè la predizione che non dipende dalla struttura gerarchica delle classi, introduce significative inconsistenze nella classificazione a causa della violazione della *true path rule*, che governa le relazioni gerarchiche tra le classi [31, 79]. In accordo con questa regola, le predizioni positive per un dato termine GO sono trasferite in modo ricorsivo ai nodi predecessori, mentre le predizioni negative vengono trasferite ai nodi discendenti [2]. In linea generale la strategia di apprendimento dei metodi gerarchici ensemble è costituita da due step: [57, 58, 68, 99]:

1. Nel primo step ogni classificatore, in modo autonomo o interagendo con altri classificatori tra di loro connessi, apprendono la classe funzionale proteica basandosi su un singolo termine. Nella maggior parte dei casi questo si traduce in un insieme di problemi di classificazione indipendenti, dove ogni classificatore viene addestrato per apprendere una specifica classe funzionale.

2. Nel secondo step le predizioni fornite dal classificatore addestrato sono combinate sfruttando le relazioni gerarchiche tra i classificatori modellati in accordo con la gerarchia delle classi funzionali.

Generalmente i metodi gerarchici migliorano in modo significativo le performance delle predizioni rispetto ai metodi "*Flat*" [98].
I metodi ensemble per la classificazione gerarchica sono stati applicati in svariati ambiti di ricerca, quali la predizione della funzione genica [52, 59, 60], la classificazione del genere musicale [100, 101, 102] o delle immagini [103, 104], l'annotazione dei video [105] e la classificazione automatica dei documenti del World Wide Web [106, 107].
Tuttavia la maggior parte dei metodi ensemble presenti in letteratura si focalizzano su tassonomie con una gerarchia strutturata ad albero [61, 62, 63, 66, 67, 79, 99, 108], mentre solamente pochi si concentrano su tassonomie aventi una gerarchia organizzata secondo

un DAG [31, 58, 85]. A tal proposito di seguito verranno presentati e discussi dei nuovi metodi ensemble gerarchici appositamente progettati per tassonomie DAG-strutturate. In particolare si discuteranno gli algoritmi hierarchical top-down (HTD-DAG) e true path rule (TPR-DAG) e alcune loro varianti .

3.1 Notazioni e Definizioni

Sia $G =< V, E >$ un grafo diretto aciclico (DAG) con vertici $V = \{1, 2, \ldots, |V|\}$ e archi $e = (i, j) \in E, i, j \in V$. G rappresenta una tassonomia strutturata come un DAG, i cui nodi $i \in V$ rappresentano le classi della tassonomia e gli archi diretti $(i, j) \in E$ le relazioni gerarchiche tra i e j: i è la classe genitore e j è la classe figlio [98]. Si suppone che nel DAG G esista solamente un nodo $root(G)$. Se nel grafo G dovessero esistere molteplici nodi root, si può aggiungere in G un nodo root semplicemente connettendo con un arco il nodo root aggiunto con i molteplici nodi root di partenza [98].
L'insieme dei nodi figli di un nodo i è indicato come $child(i)$, l'insieme dei suoi nodi genitori come $par(i)$, l'insieme dei suoi nodi ancestori come $anc(i)$ e l'insieme dei suoi nodi discendenti come $desc(i)$ [98].
Un classificatore "flat continuo" $f : X \longrightarrow \mathbb{Y}$ fornisce uno score $\hat{\boldsymbol{y}} \in \mathbb{Y} = [0, 1]^{|V|}$ per un dato campione $x \in X$. In altre parole questo classificatore fornisce uno score $\hat{y}_i \in [0, 1]$ per ogni nodo/classe $i \in V$ del DAG G [98]:

$$\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} > \tag{3.1}$$

È facile verificare come un classificatore "flat discreto" sia un caso particolare di f, dove $\hat{y}_i \in \{0, 1\}$ [98]. Infatti, in questo caso, il classificatore semplicemente assegna ($\hat{y}_i = 1$) o non assegna ($\hat{y}_i = 0$) un campione x ad una classe i. Generalizzando, il classificatore "flat continuo" fornisce uno score $\hat{y}_i \in [0, 1]$ che può essere interpretato come la probabilità del campione x di appartenere ad una data classe i [98], mentre nel caso di un classificatore "flat discreto" l'etichetta $\hat{\mathbf{y}}$ identifica direttamente l'insieme $S \subset V$ delle classi predette [98]:

$$S = \{i | i \in V \wedge \hat{y}_i = 1\} \tag{3.2}$$

Si dice che S è un insieme valido o consistente se possiede la seguente proprietà (*true path rule*) [98]:

$$S \text{ valido} \iff S = \{i | i \in V \wedge i \in S \wedge j \in par(i) \Rightarrow j \in S\} \tag{3.3}$$

Nel caso più generale degli score continui, si dice lo score multi-etichetta $\mathbf{y}$ è valido o consistente se rispetta la *true path rule* [98]:

$$\boldsymbol{y} \text{ valido} \iff \forall i \in V, i \in par(j) \Rightarrow y_i \geq y_j \tag{3.4}$$

Si evidenzia che (3.3) è una caso particolare di (3.4) [98].
Quindi un classificatore flat "continuo" o "discreto" soddisfa la *true path rule*, se ogni campione x possiede rispettivamente la proprietà 3.4 o 3.3 [98].
Nei casi reali è molto improbabile che un classificatore flat "continuo" o "discreto" soddisfi la *true path rule*, dal momento che per definizione le predizioni sono eseguite senza prendere in considerazione la gerarchia delle classi. Tuttavia, considerando la gerarchia delle classi, è possibile modificare le etichette o gli score del classificatore flat al fine di ottenere un classificatore gerarchico che obbedisca alla TPR. Per fare ciò bisogna implementare una funzione $h(f(x)) : X \longrightarrow \mathbb{Y}$ tale che $\forall x \in X$ e dato che $f(x) = \hat{\boldsymbol{y}}$ e $h(f(x)) = \bar{\boldsymbol{y}}$, per ogni $(x, \bar{\boldsymbol{y}})$ si abbia [98]:

$$\forall i \in V, i \in par(j) \Rightarrow \bar{y}_i \geq \bar{y}_j \tag{3.5}$$

A tal proposito di seguito verranno illustrati alcuni algoritmi gerarchici che implementano la funzione h.

3.2 Algoritmo Hierarchical top-down per DAGs (HTD-DAG)

Il più semplice algoritmo per DAG è l'algoritmo hierarchical top-down (HTD). Per il codice ad alto livello si rimanda all'appendice A. Questo algoritmo, esplorando il grafo per livelli dall'alto verso il basso, si basa sulle seguenti semplici regole [98]

$$\bar{y}_i = \begin{cases} \hat{y}_i & \text{se} \quad i \in root(G) \\ \min_{j \in par(i)} \bar{y}_j & \text{se} \quad \min_{j \in par(i)} \bar{y}_j < \hat{y}_i \\ \hat{y}_i & \text{altrimenti} \end{cases} \tag{3.6}$$

Si noti che $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ rappresenta l'insieme delle predizioni gerarchiche ottenute dalle predizioni flat $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$ mediante l'algoritmo HTD [109].
Per livello si intende la massima distanza che un nodo i ha dal nodo root [109]. Più precisamente se un DAG ha $\mathcal{L} = \{0, 1, \ldots, \xi\}$ livelli, $\psi : V \longrightarrow \mathcal{L}$ è la funzione livello che assegna ad ogni nodo $i \in V$ un livello, indicante la massima distanza che il nodo i ha dal nodo root [109]. Per esempio il nodo $\{i|\psi(i) = 0\}$ corrisponde al nodo root, i nodi $\{i|\psi(i) = 1\}$ corrispondono a tutti quei nodi che si trovano alla massima distanza 1 dalla root e i nodi $\{i|\psi(i) = \xi\}$ corrispondono a tutti quei nodi che si trovano alla distanza ξ dal nodo root [109].
La figura 3.1 mostra che il grafo deve essere necessariamente esplorato per livelli nel senso della massima e non della minima distanza dalla root [109]. Questo è necessario per preservare la consistenza delle predizioni. Infatti mettendo a confronto gli score gerarchici

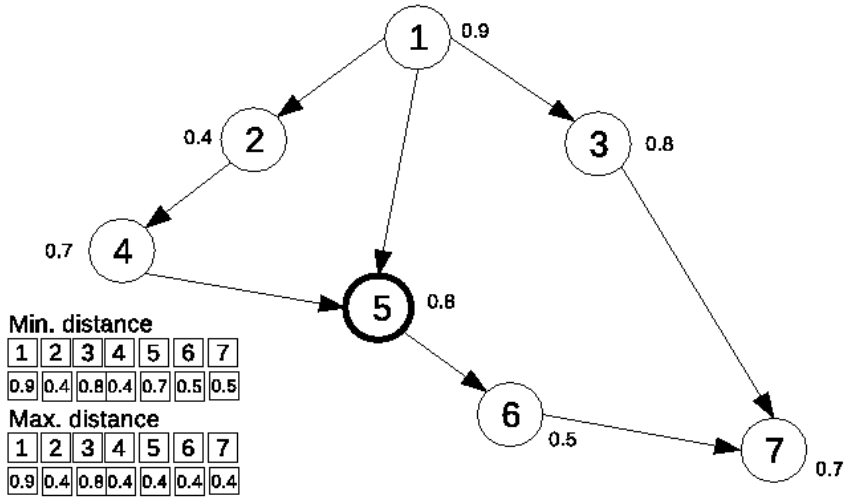


Figura 3.1: I livelli della gerarchia devono essere definiti nel senso della massima distanza dal nodo root (nodo 1). I numeri vicino ad ogni nodo indicano lo score della predizione flat, mentre la tabella in basso a sinistra mette a confronto score gerarchici top-down ottenuti considerando rispettivamente la minima e la massima distanza dal nodo root. Si rimanda al testo per ulteriori dettagli (figura tratta da [109])

top-down ottenuti determinando i livelli del grafo in base alla minima distanza dalla root, con quelli ottenuti determinando i livelli del grafo in base alla massima distanza dalla root (tabella in basso a sinistra della figura 3.1), è possibile osservare che solamente raggruppando i nodi in base alla massima distanza dalla root è possibile preservare la consistenza delle predizioni [109]. Focalizziamo, ad esempio, l'attenzione sul nodo 5. Visitando il DAG per livelli in accordo con la minima distanza dalla root, si ha che il nodo 5 appartiene al livello 1 ($\psi^{min}(5) = 1$) e applicando le regole su cui si basa HTD (eq. 3.6), lo score flat $\hat{y} = 0.8$ viene erroneamente modificato a score HTD ensemble $\bar{y} = 0.7$. Invece visitando il DAG per livelli in accordo con la massima distanza dalla root si ha che il nodo 5 appartiene al livello 3 ($\psi^{max}(5) = 3$) e lo score HTD ensemble viene correttamente settato a $\bar{y} = 0.4$. In altre parole alla fine dell'esplorazione top-down, se i livelli vengono visitati in accordo con la minima distanza, si ha che $\bar{y}_5 = 0.7 > \bar{y}_4 = 0.4$, quindi il nodo figlio ha uno score più alto rispetto a quello del nodo genitore e ciò è in contrasto con la TPR. Al contrario se i livelli vengono attraversati in accordo con la massima distanza dalla root si ha che $\bar{y}_5 \leq \bar{y}_4 = 0.4$ e la consistenza della TPR viene garantita. Questo è dovuto al fatto che, scegliendo di adottare il cammino più breve, quando si visita il nodo 5 il suo nodo genitore (nodo 4) non è ancora stato processato e di conseguenza il valore 0.4 non è stato trasmesso dal nodo 2 al nodo 4. Invece visitando il DAG in accordo con la massima distanza tutti i nodi predecessori del nodo 5 (nodo 4 incluso) sono già stati processati e lo score 0.4 viene correttamente trasmesso al nodo 5 lungo il cammino $2 \rightarrow 4 \rightarrow 5$.

Per l'algoritmo HTD-DAG è valido il seguente teorema [109]:

Teorema 1. *Dato un DAG* $G =< V, E >$*, la funzione di livello* ψ *e un insieme di predizioni flat* $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$*, per ogni classe associata ad ogni nodo* $i \in \{1, \ldots, |V|\}$*, l'algoritmo* HTD-DAG *garantisce che per l'insieme delle predizioni ensemble* $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ *venga rispettata la seguente proprietà:*

$$\forall i \in V,\ j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i \tag{3.7}$$

Infatti applicando la regola 3.6 partendo dall'alto e scendendo verso il basso nella gerarchia è possibile garantire che gli score dei nodi genitori sono più grandi o al più uguali a quelli dei suoi nodi figli [109]. Inoltre la visita per livelli della gerarchia in accordo con la funzione ψ (i livelli sono definiti nel senso della massima distanza), garantisce che ogni nodo genitore viene processato prima dei suoi nodi figli [109]. La proprietà 3.7 viene assicurata anche dal fatto che ogni nodo viene visitato solo una volta e non può essere modificato da step successivi della visita top-down [109].
Esistono molti modi per implementare la funzione di livello ψ [109]. Nel presente lavoro di tesi si è scelto di utilizzare l'algoritmo di Bellman-Ford [110]. Sapendo che l'algoritmo classico di Bellman-Ford calcola i cammini minimi di un'unica sorgente su un grafo diretto pesato (dove i pesi degli archi possono essere anche negativi), per ottenere la massima distanza dalla root è sufficiente invertire il segno dei pesi degli archi [109]. La complessità dell'algoritmo di Bellman-Ford è $\mathcal{O}(|V|^3)$ rispetto al numero dei nodi, ma dal momento che i livelli devono essere calcolati solo una volta mediante la funzione ψ, su un calcolatore moderno non esistono significative differenze in termini di tempo computazionale medio empirico [109, 111]. Altri metodi, come ad esempio le procedure basate sull'ordinamento topologico dei grafi, sono molto più efficienti rispetto all'algoritmo di Bellman-Ford [111]. Infatti, sfruttando l'ordinamento topologico dei nodi, la massima distanza di un nodo dalla root viene calcolata con una complessità computazionale $\mathcal{O}(|V| + |E|)$, che è quadratica per grafi densi e lineare per grafi sparsi rispetto al numero dei nodi da cui un DAG è composto [111].
In fig. 3.2 viene mostrato lo *pseudocodice* dell'algoritmo HTD-DAG [98]. Le righe dalla 1 alla 4 mostrano come raggruppare i nodi in base alla loro massima profondità nel grafo, mentre il blocco B dell'algoritmo esplora il grafo partendo dal livello più alto e scendendo verso il livello più basso (righe 5-16).
Partendo dal livello 1 (nodi figli della root) e per ogni livello del grafo, i nodi vengono processati e, in accordo con l'equazione 3.6, viene eseguita la correzione gerarchica top-down delle predizioni flat $\hat{y}_i$, $i \in \{1, \ldots, |V|\}$ a predizione ensemble [98]. In questo modo l'algoritmo HTD garantisce che lo score di un nodo figlio non possa essere maggiore dello score del suo nodo genitore o dei suoi nodi genitori.

Figura 3.2: **Algoritmo Hierarchical Top-Down per DAGs (HTD-DAG)**

`Input`:
- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$

`begin algorithm`

01: A. Compute $\forall i \in V$ the max distance from $root(G)$:
02: $E' := \{e' | e \in E,\ e' = -e\}$
03: $G' :=< V, E' >$
04: $dist :=$ Bellman.Ford$(G', root(G'))$
05: B. Per-level top-down visit of G:
06: $\bar{y}_{root(G)} := \hat{y}_{root(G)}$
07: `for each` d `from` 1 `to` $\max(dist)$ `do`
08: $N_d := \{i | dist(i) = d\}$
09: `for each` $i \in N_d$ `do`
10: $x := \min_{j \in par(i)} \bar{y}_j$
11: `if` $(x < \hat{y}_i)$
12: $\bar{y}_i := x$
13: `else`
14: $\bar{y}_i := \hat{y}_i$
15: `end for`
16: `end for`

`end algorithm`

`Output`:
- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$

In base a quanto discusso finora, il blocco A dell'algoritmo (righe 1-4), dominato dall'algoritmo di Bellman-Ford ha una complessità computazionale $\mathcal{O}(|V|^3)$ per grafi sparsi, mentre la complessità del blocco B (righe 5-16) è lineare nel numero dei nodi del grafo sparso [109].

3.3 Algoritmo Hierarchical True Path Rule per DAGs (TPR-DAG)

L'algoritmo che verrà discusso di seguito non è altro che un'estensione per DAG dell'algoritmo TPR originariamente proposto per tassonomie strutturate ad albero [32, 66]. La

principale differenza rispetto alla precedente versione consiste in un'esplorazione separata per livelli: l'esplorazione top-down viene preceduta da una bottom-up. Mentre nella versione per alberi le esplorazioni bottom-up e top-down si alternano ad ogni livello del grafo [66], nella versione per DAG la separazione degli step bottom-up e top-down è necessaria per garantire la consistenza della TPR delle predizioni [98]. Infatti il primo step (bottom-up) della versione adatta ai DAG, esplorando il DAG dal basso verso l'alto e propagando le "predizioni positive" ai nodi genitori ed in modo ricorsivo ai nodi predecessori, si pone come obiettivo di aumentare la sensibilità delle predizioni. Invece, lo step top-down, partendo dal nodo root ed attraversando il DAG verso il basso, propaga le predizioni negative verso i nodi figli ed in modo ricorsivo verso i nodi discendenti, aumentando così la precisione delle predizioni [111].
Sempre nella versione per DAG,dal momento che da ogni nodo alla root sono possibili cammini multipli, il livello a cui una classe appartiene è definito in termini di massima distanza dalla root mediante la funzione di livello ψ. Quindi la separazione degli step bottom-up e top-down e la funzione di livello ψ garantiscono che lo score di un nodo genitore o di un nodo predecessore sia sempre più grande o al più uguale a quello dei suoi nodi figli o dei suoi nodi discendenti [111].
La versione "*vanilla*" dell'algoritmo TPR-DAG, analogamente alla versione per gli alberi, adotta una visita per livelli dal basso verso l'alto del DAG G, partendo dai nodi più lontani (nel senso della massima distanza) dalla root. Più precisamente se $p(r,i)$ rappresenta un cammino dal nodo root r al nodo $i \in V$, $l(p(r,i))$ la lunghezza del percorso p, $\mathcal{L} = 0, 1, \ldots, \xi$ l'insieme dei livelli osservati, dove ξ rappresenta il livello massimo, allora $\psi : V \longrightarrow \mathcal{L}$ è la funzione di livello che associa ogni nodo $i \in V$ al livello di appartenenza $\psi(i)$:

$$\psi(i) = max_{p(r,i)} l(p(r,i)) \tag{3.8}$$

Ad ogni livello le predizioni flat $\hat{y}_i$ sono modificate prendendo in considerazione le predizioni positive provenienti dai nodi figli nel modo seguente [111]:

$$\bar{y}_i := \frac{1}{1 + |\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \tilde{y}_j) \tag{3.9}$$

dove ϕ_i sono i figli "positivi" di i, cioè i nodi responsabili della propagazione bottom-up delle predizioni positive.
Tuttavia, in questo conteso, la scelta dei nodi figli positivi rappresenta un problema cruciale e per la loro selezione si possono adottare diversi tipi di approcci [111]. Nel presente lavoro di tesi si sono seguite le seguenti strategie:

1. *True Path Rule Threshold Free* (TPR-TF). Una semplice soluzione consiste nel selezionare come nodi positivi tutti i nodi figli che posseggono uno score più alto

rispetto a quello dei loro nodi genitori [111]:

$$\phi_i := \{j \in child(i) | \bar{y}_j > \tilde{y}_i\} \tag{3.10}$$

2. *True Path Rule with Threshold* (TPR-T). In questo caso i nodi positivi vengono scelti fissando una uguale soglia $t_j \; \forall \; i \in V$:

$$\phi_i := \{j \in child(i) | \tilde{y}_j > t_j\} \tag{3.11}$$

   In pratica, per il tuning del parametro t_j, si è lanciato la variante TPR-T dell'algoritmo TPR-DAG (fig. 3.3) facendo variare la soglia t_j tra 0.1 e 0.9 con un passo di 0.1. In base al miglior risultato di performance osservato si è quindi scelto il miglior valore di t_j. Come metriche di performance si sono utilizzate l'area sotto la curva ROC (AUC) e la precisione a fissati livelli di recall (PxR). I livelli di recall scelti per valutare le performance sono stati 10, 20 e 40.

3. *Weighted True Path Rule* (TPR-W). In questo approccio viene aggiunto un peso $w \in [0, 1]$ per bilanciare il contributo tra il nodo i e l'insieme dei suoi figli "positivi" $\phi(i)$ [98]. In questo modo, analogamente alla versione TPR-W per le tassonomie strutturate ad albero [79, 65], si è progettata anche la variante "pesata" per le tassonomie strutturate secondo un DAG, semplicemente sostituendo la riga 10 dell'algoritmo TPR-DAG (fig. 3.3) con la seguente riga di *pseudocodice* [98]:

$$\tilde{y}_i := w\hat{y}_i + \frac{(1-w)}{|\phi_i|} \sum_{j \in \phi_i} \tilde{y}_j \tag{3.12}$$

   In pratica, per tuning del parametro w, si è lanciato la variante TPR-W dell'algoritmo TPR-DAG (fig. 3.3) facendo variare il peso w tra 0.1 e 0.9 con uno step di 0.1. Analogamente a quanto fatto per t_j, il miglior valore di w è stato scelto in base al miglior risultato di AUC, P10R, P20R e P40R ottenuto.

4. *Weighted True Path Rule with Threshold* (TPR-WT). Una procedura del tutto analoga è stata seguita anche per la doppia parametrizzazione di t_j e w. In questo caso la miglior performance di AUC, P10R, P20R e P40R è determinata dalla miglior combinazione dei parametri t_j e w.

Indipendentemente dalla scelta della variante dell'algoritmo TPR-DAG, le predizioni vengono propagate dal basso verso l'alto, dai nodi figli "positivi" verso i nodi genitori ed in modo ricorsivo verso i nodi predecessori [111]. Per il codice ad alto livello delle varianti dell'algoritmo TPR-DAG sopra descritte si rimanda all'appendice B. Inoltre, dato che il contributo dei nodi discendenti di un nodo i decade esponenzialmente con l'accrescere

della loro distanza dal nodo i [66], per incrementare il contributo dei nodi più specifici, si potrebbe considerare un contributo costante dei nodi discendenti "positivi" [98]:

$$\tilde{y}_i := \frac{1}{1 + |\Delta_i|}(\hat{y}_i + \sum_{j \in \Delta_i} \tilde{y}_j) \tag{3.13}$$

dove

$$\Delta_i = \{j \in desc(i) | \tilde{y}_j > t_j\} \tag{3.14}$$

In altre parole si potrebbe implementare un'ulteriore variante dell'algoritmo TPR-DAG in cui, invece di considerare i soli figli "positivi" di un nodo i, si potrebbero considerare tutti i suoi discendenti [98].

Il successivo step top-down, partendo dal livello più alto della gerarchia (nodo root) e muovendosi verso i livelli più specifici, modifica gli score precedentemente computati nello step bottom-up [111]. Infatti il principale obiettivo di questo step consiste nel propagare le decisioni "negative" verso i figli ed in modo ricorsivo verso i discendenti di ogni nodo, per rendere le predizioni consistenti e più precise [111]. Lo step top-down, esplorando il grafo per livelli dall'alto verso il basso, assume la seguente semplice regola [111]:

$$\bar{y}_i := \begin{cases} \tilde{y}_i & \text{se} \quad i \in root(G) \\ \min_{j \in par(i)} \bar{y}_j & \text{se} \quad \tilde{y}_i > \min_{j \in par(i)} \bar{y}_j \\ \tilde{y}_i & \text{altrimenti} \end{cases} \tag{3.15}$$

dove gli score $\tilde{y}_i$ sono quelli computati nello step bottom-up, mentre $\bar{y}_i$ sono gli score ensemble computati dall'algoritmo TPR-DAG [111].

Lo step top-down garantisce la consistenza gerarchia delle predizioni della TPR, come dichiarato dal seguente teorema, la cui dimostrazione è del tutto analoga a quella del teorema 1 [111]:

Teorema 2. *Dato un DAG $G =< V, E >$, una funzione di livello ψ che assegna ad ogni nodo la sua massima distanza dal nodo root, un insieme di predizioni $\tilde{\boldsymbol{y}} =< \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{|V|} >$ generate dallo step bottom-up dell'algoritmo TPR per ogni classe associata al suo nodo corrispondente $i \in 1, \ldots, |V|$, lo step top-down dell'algoritmo TPR garantisce che per l'insieme delle predizioni ensemble $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ venga rispettata la seguente proprietà:*

$$\forall i \in V,\ j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$$

Dal teorema 2 si può dimostrare, procedendo per assurdo, che la consistenza delle predizioni della TPR viene rispettata anche da tutti i predecessori di un nodo $i \in V$:

Corollario 3. *Dato un DAG $G =< V, E >$, una funzione di livello ψ, un insieme di predizioni flat $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$ per ogni classe associata al suo nodo corrispondente*

$i \in 1, \ldots, |V|$, *l'algoritmo* TPR *garantisce che per l'insieme delle predizioni ensemble* $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ *venga rispettata la seguente proprietà:*

$$\forall i \in V,\ j \in anc(i) \Rightarrow \bar{y}_j \geq \bar{y}_i$$

In fig. 3.3 viene mostrato lo *pseudocodice* ad alto livello dell'algoritmo TPR-DAG [111]. Le prime quattro righe mostrano come raggruppare i nodi per livello usando l'algoritmo di

Figura 3.3: **Algoritmo Hierarchical True Path Rule per DAGs (TPR-DAG)**

`Input`:
- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$, 1 is the *root* node
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$

`begin algorithm`

A. Compute $\forall i \in V$ the max distance from $root(G)$:
  $E' := \{e' | e \in E,\ e' = -e\}$
  $G' :=< V, E' >$
  $dist :=$ Bellman.Ford$(G', root(G'))$
B. Per-level bottom-up visit of G:
  `for each` d `from` $\max(dist)$ `to` 0 `do`
    $N_d := \{i | dist(i) = d\}$
    `for each` $i \in N_d$ `do`
      Select ϕ_i according to a *positives selection strategy*
      $\tilde{y}_i := \frac{1}{1+|\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \tilde{y}_j)$
    `end for`
  `end for`
C. Per-level top-down visit of G:
  $\bar{y}_1 := \tilde{y}_1$
  `for each` d `from` 1 `to` $\max(dist)$ `do`
    $N_d := \{i | dist(i) = d\}$
    `for each` $i \in N_d$ `do`
      $x := \min_{j \in par(i)} \bar{y}_j$
      `if` $(x < \tilde{y}_i)$
        $\bar{y}_i := x$
      `else`
        $\bar{y}_i := \tilde{y}_i$
    `end for`
  `end for`

`end algorithm`

`Output`:
- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$

Bellman-Ford. Il blocco B (righe 5-12) esegue una visita bottom-up del grafo e aggiorna le predizioni $\tilde{y}_i$ dell'algoritmo TPR-DAG in accordo con le eq. 3.9 e con una delle strategie per la selezione dei nodi positivi precedentemente descritte (eq. 3.10, 3.11 e 3.12). Si tenga presente che questo step dell'algoritmo serve solo per propagare le predizioni "positive" dal basso verso l'alto ma non garantisce la consistenza della TPR sulle predizioni [111]. La TPR è garantita dal blocco C dell'algoritmo (righe 13-24) che semplicemente implementa lo step gerarchico top-down, in accordo con la procedura descritta nella sezione 3.2.
Il blocco A dell'algoritmo TPR-DAG, dominato dall'algoritmo di Bellman-Ford, ha una complessità computazionale $\mathcal{O}(|V|^3)$ per grafi sparsi, mentre la complessità sia del blocco B che del blocco C è $\mathcal{O}(|V|)$ [111]. Si noti inoltre che, mentre il blocco A deve essere eseguito solamente una volta per tutti i campioni, il blocco B ed il blocco C devono essere iterati per ogni campione per cui si vogliono predire le etichette [111].

L'algoritmo TPR-DAG garantisce una sensibilità maggiore rispetto all'algoritmo HTD-DAG, come enunciato dal teorema 4 e dal corollario 5:

Teorema 4. *Dato un DAG* $G =< V, E >$*, un insieme di predizioni flat* $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >$ *per ogni classe associata ad ogni nodo* $i \in \{1, \ldots, |V|\}$*, un insieme di predizioni ensemble* $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$ *per HTD-DAG e un insieme di predizioni ensemble* $\breve{\boldsymbol{y}} =< \breve{y}_1, \breve{y}_2, \ldots, \breve{y}_{|V|} >$ *per TPR-DAG, dove i figli positivi sono stati scelti in accordo con una delle strategia precedentemente menzionate (eq. 3.10, 3.11 e 3.12), si ha che* $\forall i \in V, \quad \breve{y}_i \geq \bar{y}_i$

La motivazione di ciò si basa sul fatto che l'esplorazione dal basso verso l'alto dell'algoritmo TPR-DAG può solamente incrementare lo score $\breve{y}_i$ rispetto alle predizioni flat $\bar{y}_i$. Ne consegue che la visita dall'alto verso il basso in TPR-DAG è identica a quella di HTD-DAG, ma con l'unica differenza che gli score di partenza sono più elevati [98].

Corollario 5. *L'algoritmo TPR-DAG, dove i figli positivi sono stati scelti in accordo con una delle strategia precedentemente menzionate (eq. 3.10, 3.11 e 3.12), mostra sempre una sensibilità uguale o maggiore rispetto a quella dell'algoritmo HTD-DAG.*

Dal teorema 4 si ha che $\forall i \in \breve{y}_i \geq \bar{y}_i$, per cui TPR-DAG rispetto a HTD-DAG [98]:

(a) incrementa o mantiene costante il numero di veri positivi;

(b) decrementa o mantiene costante il numero di falsi negativi

Poiché la sensibilità è definita come il rapporto tra i veri positivi ed il numero totale di positivi presenti in un test, TPR-DAG avrà sempre una sensibilità uguale o maggiore di HTD-DAG [98].
Un'altra variante ancora dell'algoritmo TPR-DAG è rappresentata dall'algoritmo ISO-TPR, il cui *pseudocodice* è mostrato in fig. 3.4. Rispetto all'algoritmo TPR-DAG, nell'algoritmo

Figura 3.4: **Algoritmo Hierarchical Isotonic True Path Rule per DAGs (ISO-TPR)**

`Input`:
- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$, 1 is the *root* node
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$

`begin algorithm`

01: A. Compute $\forall i \in V$ the max distance from $root(G)$:
02: $E' := \{e' | e \in E, e' = -e\}$
03: $G' :=< V, E' >$
04: $dist :=$ Bellman.Ford$(G', root(G'))$
05: B. Per-level bottom-up visit of G:
06: `for each` d `from` $\max(dist)$ `to` 0 `do`
07: $N_d := \{i | dist(i) = d\}$
08: `for each` $i \in N_d$ `do`
09: Select ϕ_i according to a *positives selection strategy*
10: $\tilde{y}_i := \frac{1}{1+|\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \tilde{y}_j)$
11: `end for`
12: `end for`
13: C. Isotonic correction:
14: $\hat{\boldsymbol{y}} := \tilde{\boldsymbol{y}}$
15: $\bar{\boldsymbol{y}} = \begin{cases} \min_{\bar{\boldsymbol{y}}} \sum_{i \in V} (\hat{y}_i - \tilde{y}_i)^2 \\ \forall i, \quad j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i \end{cases}$

`end algorithm`

`Output`:
- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$

ISO-TPR l'esplorazione top-down è rimpiazzata da un approccio di regressione ad ordine parziale [74]. In altre parole, viene risolto il seguente problema di ottimizzazione:

$$\bar{\boldsymbol{y}} = \begin{cases} \min_{\bar{\boldsymbol{y}}} \sum_{i \in V} (\hat{y}_i - \bar{y}_i)^2 \\ \forall i, \quad j \in par(i) \Rightarrow \bar{y}_j \geq \bar{y}_i \end{cases} \tag{3.16}$$

In questo modo le costrizioni della TPR vengono mantenute dalla costruzione ed è selezionata la soluzione più vicina alle predizioni flat (nel senso dell'errore quadratico) che obbedisce alla TPR.

Mentre i metodi HTD-DAG e TPR-DAG sono stati implementati in modo efficiente nel linguaggio di programmazione R. la variante ISO-TPR è risultata essere onerosa in termini computazionali. Di conseguenza per questo algoritmo non si sono valutate le performance, ma ci si è limitati ad implementare lo *pseudocodice* (fig. 3.4). In appendice B si riporta il codice ad alto livello. Tuttavia, una nuova procedura algoritmica che implementa in modo efficiente ISO-TPR è in fase di studio e sperimentazione.

Parte Sperimentale

La parte sperimentale del presente lavoro di tesi si articola principalmente in due parti. Per prima cosa gli score flat sono stati processati utilizzando due diversi tipi di classificatori, RANKS e LP. Successivamente, i metodi di ensemble gerarchici descritti nella sezione 3, sono stati applicati a due diverse ontologie entrambe strutturate secondo grafi diretti aciclici: GO (sezione 4.3) e HPO (sezione 4.2). La principale differenza tra queste due ontologie risiede nel fatto che mentre la tassonomia HPO è strutturata in un'unica ampia classe, la GO è suddivisa in tre diverse e ampie divisioni (BP, MF e CC).

4.1 Base Learners

Per processare e fornire gli score flat utilizzati negli esperimenti del presente lavoro di tesi, si sono utilizzati due diversi metodi semi-supervisionati basati su reti: RANKS e LP. Di seguito verranno spiegate le caratteristiche salienti di ognuno dei due *base learner*, soffermandosi in modo particolare sul primo, in quanto tra i due, è quello che è stato applicato con più successo in ambito medico-farmacologico.

4.1.1 RANKS

Il classificatore RANKS è un metodo di apprendimento semi-supervisionato basato su reti che è stato recentemente applicato con successo alla *gene disease prioritization* [47], alla predizione della funzione proteica [112] e al riposizionamento di farmaci [113].
Le funzioni di score kernelizzate adottano una strategia sia globale che locale [109]. L'apprendimento locale è raggiunto mediante una generalizzazione del classico approccio di *guilt-by-association* [45], attraverso l'introduzione di diverse funzioni per quantificare la similarità tra un gene e i suoi vicini [109]. La strategia globale invece, è introdotta in forma di un kernel che è in grado di catturare la topologia globale della rete biomolecolare [109]. Più precisamente, con questo approccio è possibile ricavare le funzioni di score $S : V \longrightarrow \mathbb{R}^+$ basate su opportune funzioni kernel definite su grafi, attraverso le quali è possibile direttamente ordinare i vertici v in accordo con il valore di $S(v)$: maggiore è lo score $S(v)$ e più elevata sarà la probabilità che il gene v appartenga ad una data classe C [112]. Le funzioni di score sono costruite sulla base di misure di distanza definite nello spazio di Hilbert $\mathcal{H}$ e ricavate utilizzando il "trucco kernel", attraverso il quale, al posto di calcolare esplicitamente il prodotto interno $< \phi(\cdot), \phi(\cdot) >$ nello spazio di Hilbert con $\phi : V \longrightarrow \mathcal{H}$, si calcola la funzione kernel associata $K : V \times V \longrightarrow \mathbb{R}^+$ nello spazio di

input V [109]. Sia $D(v, V_C)$ una misura di distanza nello spazio di Hilbert tra un dato gene/vertice v e un un insieme di geni V_C appartenenti ad una specifica classe C. Benché sia possibile definire diversi tipi di misure di distanza [112], negli esperimenti svolti nel seguente lavoro di tesi, i classificatori locali associati a ciascun termine di un DAG G si basano sulla distanza media D_{AV}:

$$D_{AV}(v, V_C) = \frac{1}{|V_C|} \sum_{x \in V_C} ||\phi(v) - \phi(x)||^2 \tag{4.1}$$

Basandosi su questa distanza è possibile ricavare la funzione *Average score*. Infatti sviluppando il quadrato in 4.1 si ottiene:

$$D_{AV}(v, V_C) = < \phi(v), \phi(v) > + \frac{1}{|V_C|} \sum_{x \in V_C} < \phi(x), \phi(x) > - \frac{2}{|V_C|} \sum_{x \in V_C} < \phi(v), \phi(x) > \tag{4.2}$$

Ricordando che $< \phi(\cdot), \phi(\cdot) >= K(\cdot, \cdot)$ e che per ottenere una misura di similarità è sufficiente cambiare il segno all'equazione 4.2, l'equazione 4.2 diventa:

$$Sim_{AV}(v, V_C) = -K(v, v) + \frac{2}{|V_C|} \sum_{x \in V_C} K(v, x) - \frac{1}{|V_C|} \sum_{x \in V_C} K(x, x) \tag{4.3}$$

Osservando che il terzo termine dell'equazione 4.3 è identico per ogni $v \in V$, si ricava la seguente funzione di *Average score* S_{AV}:

$$S_{AV}(v, V_C) = -K(v, v) + \frac{2}{|V_C|} \sum_{x \in V_C} K(v, x) \tag{4.4}$$

L'equazione 4.4 può essere ulteriormente semplificata rimuovendo il primo termine, dato che tutte le funzioni $K(v, v)$ sono uguali per ogni v:

$$S_{AV}(v, V_C) = \frac{2}{|V_C|} \sum_{x \in V_C} K(v, x) \tag{4.5}$$

In linea di principio ogni opportuna funzione kernel K può essere applicata per calcolare la sopracitata funzione di score kernelizzata, ma negli esperimenti del presente lavoro di tesi si è utilizzata la funzione *random walk kernel* [72], dal momento che è in grado di catturare la similarità tra ogni coppia di vertici appartenenti ad un medesimo grafo G. Più precisamente è stata utilizzata una funzione *random walk kernel* ad 1-step, al fine di esaminare solamente i nodi vicini di ogni gene nella rete di interazione funzionale [109].
Si sarebbero potute applicare anche funzioni di score con *random walk kernel* a 2 o più step al fine di esplorare meglio la topologia della rete e, almeno potenzialmente, ottenere risultati migliori specialmente per i vertici del grafo con poche annotazioni o per i nodi "positivi" relativamente lontani l'uno dall'altro, ma il principale obiettivo che il presente lavoro di tesi si è prefissato di raggiungere è quello di verificare che i nuovi algoritmi per

DAG descritti nella sezione 3, migliorassero in modo significativo le predizioni rispetto all'approccio *Flat* e non che raggiungessero i migliori risultati in assoluto. Per una motivazione del tutto analoga, non sono state applicate altre tipologie di funzioni kernel, anche se potenzialmente avrebbero potuto condurre a risultati migliori.
Si noti che le funzioni di score kernelizzate non restituiscono direttamente una probabilità, ma uno score. Quindi per rendere gli score delle varie classi confrontabili a livello di un intervallo di valori, gli score flat, generati mediante il classificatore RANKS, devono necessariamente essere normalizzati. Ad esempio, una semplice normalizzazione, potrebbe essere quella di dividere gli score di ogni classe per il suo score massimo, in modo tale da garantire che gli score di tutte le classi siano compresi nell'intervallo di valori $[0, 1]$.

4.1.2 Label Propagation

L'algoritmo di *label propagation* (LP) è un metodo semi-supervisionato basato su campi gaussiani e su funzioni armoniche [114], che è stato utilizzato per processare gli score flat di solamente uno degli esperimenti del presente lavoro di tesi. Molto brevemente, gli algoritmi di *label propagation*, sono in grado di propagare le etichette delle proteine annotate nella rete, sfruttando la topologia del grafo sottostante.
Poiché il *base learner* LP ritorna già la probabilità che un gene v appartenga alla classe C, gli score flat generati mediante l'algoritmo di *label propagation*, non devono subire nessun tipo di normalizzazione, dal momento che gli score sono già compresi nell'intervallo di valori $[0, 1]$.

4.2 Predizione delle Associazioni Geni Umani Fenotipi Patologici

La caratterizzazione delle malattie umane mediante dati fenotipici e l'ammontare sempre crescente dei dati genomici resi disponibili dalle biotecnologie *high-throughput*, hanno condotto ad una migliore comprensione dei meccanismi biomolecolari che stanno alla base delle malattie umane [111]. Infatti l'analisi fenotipica è fondamentale sia nel comprendere la patofisiologia delle rete cellulare che nel mappare i geni implicati nelle malattie [115].
Con il termine *ontologia* generalmente ci si riferisce ad una rappresentazione computazionale ad alto livello dei domini della conoscenza basata su vocabolari controllati [109]. La Human Phenotype Ontology (HPO) mira a fornire una categorizzazione standardizzata delle anomalie fenotipiche associate alle malattie umane, ognuna rappresentata da un termine HPO, e delle loro relazioni semantiche [111]. La HPO è stata sviluppata, ed è tutt'ora in fase di sviluppo, utilizzando dati di letteratura medica e riferimenti incrociati ad altre ontologie biomediche quali OMIM, *Orphanet* e DECIFER [116]. Infatti, una caratteristica chiave di HPO è proprio la sua capacità, basata sul mappaggio delle corrispondenze con altri vocabolari fenotipici pubblici, di consentire l'integrazione e l'interoperabilità con molteplici risorse biomediche [117].
È importante notare che un termine HPO non rappresenta una malattia, ma denota piuttosto le caratteristiche, i sintomi e altre anomalie fenotipiche che caratterizzano la malattia stessa [109]. Quindi una malattia può essere caratterizzata da uno o più termini HPO e i termini HPO possono essere associati con molteplici definite malattie [109].
I termini HPO sono strutturati in accordo con un DAG, dove ogni termine figlio può essere interpretato come una sottoclasse dei loro genitori e le relazioni gerarchiche tra i termini sono definite mediante relazioni is_a [109, 111].

4.2.1 Costruzione ed Integrazione delle Reti Funzionali Proteiche

I dati utilizzati per questo esperimento si basano sulla *release* di Settembre del 2013, in cui l'ontologia HPO contava esattamente $10,099$ termini e $13,382$ relazioni tra le classi [109, 111]. Dalla stessa *release* si sono scaricate anche tutte le annotazioni disponibili (associazioni gene-termine) dei $20,257$ geni umani. Di questi $20,257$ geni solamente 2759 sono annotati con almeno un termine HPO, mentre i restanti $17,498$ non sono annotati con nessun termine HPO (da notare il forte sbilanciamento tra esempi positivi e negativi).
L'insieme dei $20,257$ geni umani considerati sono stati ottenuti dalla recente *challenge* internazionale *valutazione critica della annotazione della funzione proteica* (CAFA2, *critical*

assessment of protein function annotation) [109, 111].
Partendo dai 20,257 geni umani si sono costruiti, per ogni gene, differenti vettori di profili binari, indicanti l'assenza o la presenza di caratteristiche biomolecolari nel prodotto genico codificato dal gene considerato, utilizzando le banche dati elencate nella tabella 4.1 [109, 111]. Questi vettori binari sono stati poi usati per costruire $n = 8$ reti geniche,

Database	**Content**	**Web site**
InterPro	functional family, domains, functional sites	www.ebi.ac.uk/interpro
Pfam	functional family, domains	http://pfam.xfam.org/
PRINTS	protein fingerprints, conserved motifs	www.bioinf.manchester.ac.uk
PROSITE	domains, families, functional sites	http://prosite.expasy.org
SMART	modular architectures	http://smart.embl-heidelberg.de
SUPFAM	structural and functional annotation	http://supfam.cs.bris.ac.uk
Gene Ontology	biological processes, cellular components and molecular functions	http://geneontology.org
OMIM	genetic diseases	http://omim.org
FI net (Wu et al.)	integrated network with expert-curated and non-curated sources of information	
HumanNet (Lee et al.)	integrated network with multi-species data	

Tabella 4.1: Sorgenti di dati utilizzate negli esperimenti.

una per ogni sorgente di dati elencata nella tabella 4.1, semplicemente calcolando la similarità di Jaccard tra ogni possibile coppia di vettori delle caratteristiche associati ai geni [109, 111].
Infine le n reti d'interazione genica funzionale sono state combinate semplicemente mediando i pesi degli archi w_{ij}^d di ogni rete $d \in \{1, n\}$[118]:

$$\bar{w}_{i,j} = \frac{1}{n} \sum_{d=1}^{n} w_{ij}^d \tag{4.6}$$

Al fine di costruire una rete d'interazione genica più informativa, al processo d'integrazione sono state aggiunte due ulteriori reti geniche funzionali (le reti FI e HumanNet della tabella 4.1) prese dalla letteratura [119, 120], integrando così ben 10 reti biomolecolari (Tabella 4.1).

Negli esperimenti del presente lavoro di tesi l'ontologia di partenza HPO (10,099 termini e 13,382 relazioni tra le classi) è stata "sfoltita" al fine di predire solamente:

1. termini HPO con annotazioni ≥ 2
2. termini HPO con annotazioni ≥ 50

Nel primo caso è stato ottenuto un DAG-HPO caratterizzato da 4847 termini e 5925 relazioni tra i termini, e nel secondo un DAG-HPO caratterizzato da 912 termini e da 1095 relazioni tra i termini. Ognuno dei due DAG-HPO è stato processato in maniera differente, utilizzando diversi *base learners* per generare gli score flat (RANKS e LP), applicando

diverse tecniche di validazione statistica (e.g., *leave-one-out* e 5-*fold-cross-validation*) e utilizzando gli algoritmi di correzione gerarchica descritti nella sezione 3.

4.2.2 Risultati

Iniziamo col considerare il DAG-HPO caratterizzato da 4847 termini e 5925 relazioni tra i termini, in cui si è intenzionati a predire solamente i termini HPO con almeno due annotazioni. A tal fine si è comparato l'algoritmo HTD (sottosezione 3.2) con i metodi euristici di Obozinski (MAX, AND, OR) e con le predizioni flat ottenute con una funzione 1-*step random walk kernel* e applicando la classica tecnica statistica di *leave-one-out*.
Nella tabella 4.2 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per il metodo *Flat* e per i tre metodi euristici di Obozinski originariamente proposti per la predizione gerarchia dei termini della GO [31]. Vale la pena sottolineare che nonostante nel lavoro di Obozinski et al. [31], i metodi gerarchici basati sulla regressione isotonica avessero ottenuto risultati migliori, in tutti gli esperimenti del presente lavoro di tesi sono stati ugualmente utilizzati gli algoritmi di ensemble gerarchici euristici. Tale scelta è dovuta semplicemente al fatto che i metodi basati sulla regressione isotonica hanno una complessità computazionale troppo elevata, considerando le dimensioni relativamente grandi delle due tassonomie HPO, di quelle ancora più "pesanti" delle tre reti dell'ontologia GO (vedere la sottosezione 4.3) e delle corrispondenti matrici delle etichette.

	FLAT	**HTD**	**MAX**	**AND**	**OR**
AUC	0.7897	0.7923	0.7879	**0.8151**	0.7880
P10R	0.1620	**0.1957**	0.1315	0.1665	0.1352
P20R	0.1278	**0.1535**	0.1081	0.1283	0.1110
P40R	0.0812	**0.0890**	0.0728	0.0758	0.0741

Tabella 4.2: AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R) utilizzando come classificatore le funzioni di score kernelizzate ad uno step (RANKS). I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto.

Come si può notare dalla tabella 4.2 il metodo HTD ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat*, sia in termini di AUC che di PxR. Anche se la differenza in favore di HTD è molto piccola (0.7923 vs 0.7897), considerando l'AUC medio di ognuno dei 4846 termini HPO (nodo root escluso, in quanto non deve essere predetto), il metodo gerarchico HTD migliora i risultati rispetto all'approccio *Flat* in 3346 termini HPO, ottiene lo stesso risultato in 554 termini HPO e ottiene risultati peggiori in 956 termini HPO. Questo significa che si ottiene un miglioramento per più di 3/4 dei termini HPO e questo spiega anche perché, in accordo con il test di *Wilcoxon-Mann-Whitney*, la differenza tra i due metodi è statisticamente significativa in favore di HTD al livello di

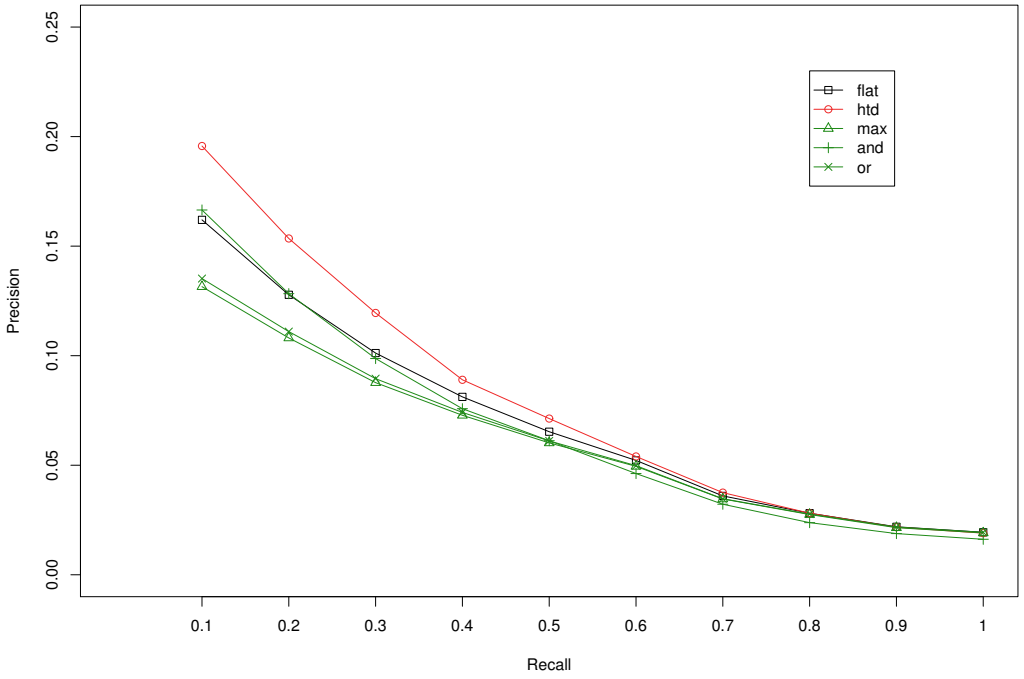


Figura 4.1: Precisione media a fissati livelli di recall tra i termini HPO (classificatore: RANKS).

significatività $\alpha = 10^{-5}$. Risultati migliori si ottengono anche considerando la precisione a fissati livelli di recall, dove il metodo HTD migliora la precisione rispetto all'approccio *Flat* tra il 10% e il 20%, almeno per i livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.2). Il metodo HTD ottiene risultati significativamente migliori anche rispetto agli altri metodi di ensemble gerarchici, fatta eccezione per la sola AUC, in cui il miglior risultato è dato dal metodo gerarchico AND. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.1), dove è facile osservare come la curva del metodo HTD si trova più in alto rispetto a quelle degli altri metodi comparati, evidenziando come, ancora una volta, il metodo HTD ottiene in media risultati migliori rispetto agli altri metodi.
Anche se la precisione media a fissati livelli di recall è relativamente bassa per tutti i metodi comparati (fig 4.1), bisogna tenere conto anche del fatto che si è provato a predire termini aventi solamente due annotazioni positive, un obiettivo molto difficile che probabilmente ha portato ad ottenere in molti casi valori di precisione prossimi allo zero. In ogni caso, l'algoritmo HTD mostra significativi miglioramenti di precisione rispetto all'approccio *Flat* almeno per bassi livelli di recall. Al contrario, i metodi di ensemble gerarchici euristici MAX, AND e OR, non sono in grado di migliorare le performance della predizione *Flat*, confermando i risultati precedentemente ottenuti in letteratura nel contesto della predizione genica [31].

Consideriamo ora il DAG-HPO caratterizzato da 912 termini e 1095 relazioni tra i termini, in cui si è intenzionati a predire le associazioni gene-fenotipo anomalo dei termini HPO

con 50 o più annotazioni. In questo caso gli score flat sono stati processati utilizzando come *base learner* i classificatori RANKS ed LP e applicando in entrambi i casi la tecnica di validazione statistica 5-*fold cross-validation*. Successivamente gli score flat prodotti da ogni classificatore sono stati corretti utilizzando le diverse varianti dell'algoritmo TPR-DAG (sezione 3.3) e i metodi euristici di Obozinski (MAX, AND, OR). Tra le varie varianti dell'algoritmo TPR-DAG implementate è stata scelta quella con le performance migliori. Dopodiché, per ogni classificatore e in base alle performance ottenute, i vari metodi gerarchici sono stati comparati con l'approccio *Flat*. Infine i due classificatori sono stati messi a confronto, al fine di verificare se i metodi gerarchici dipendono o meno dalla scelta del tipo di classificatore.

Nella tabella 4.3 si riportano i risultati medi per termine ottenuti per l'algoritmo gerarchico True Path Rule Threshold Free (TPR-TF), per il metodo *Flat* e per i tre metodi euristici di Obozinski, utilizzando come classificatore le funzioni di score 1-*step random walk kernel* (RANKS).

	FLAT	**TPR-TF**	**MAX**	**AND**	**OR**
AUC	0.8213	**0.8269**	0.8241	**0.8274**	0.8241
P10R	0.2969	**0.3427**	0.2908	0.2815	0.2994
P20R	0.2043	**0.2333**	0.2025	0.1903	0.2081
P40R	0.1054	**0.1225**	0.1071	0.0993	0.1095

Tabella 4.3: AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

Come si può notare dalla tabella 4.3 la variante TPR-TF dei metodi di ensemble gerarchici TPR ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat*, sia in termini di AUC che di PxR. Anche se la differenza in favore di TPR-TF è molto piccola (0.8269 vs 0.8213), considerando l'AUC medio di ognuno dei 911 termini HPO (escludendo il nodo root, che non deve essere predetto), il metodo gerarchico TPR-TF migliora i risultati rispetto all'approccio *Flat* in 830 termini HPO e ottiene risultati peggiori nei rimanenti 81 termini HPO. Questo significa che nel 90% dei termini HPO si ha un miglioramento della performance AUC e questo spiega anche perché, in accordo con il test di *Wilcoxon-Mann-Whitney*, la differenza tra i due metodi è statisticamente significativa in favore di TPR-TF al livello di significatività $\alpha = 10^{-5}$. La variante TPR-TF ottiene risultati migliori anche rispetto ai tre metodi di ensemble gerarchici euristici MAX,AND e OR. Più precisamente la differenza è statisticamente significativa rispetto ai metodi MAX e OR, mentre non si è registrata nessuna differenza statisticamente significativa rispetto al metodo AND.
La variante TPR-TF ottiene risultati migliori anche in termini di precisione a fissati livelli

di recall. Infatti la differenza è statisticamente significativa sia rispetto all'approccio *Flat* che rispetto ai tre metodi di ensemble gerarchici euristici, in tutti e tre i livelli di recall considerati (Tabella 4.3). Questi risultati sono confermati anche dalla curva precisione-

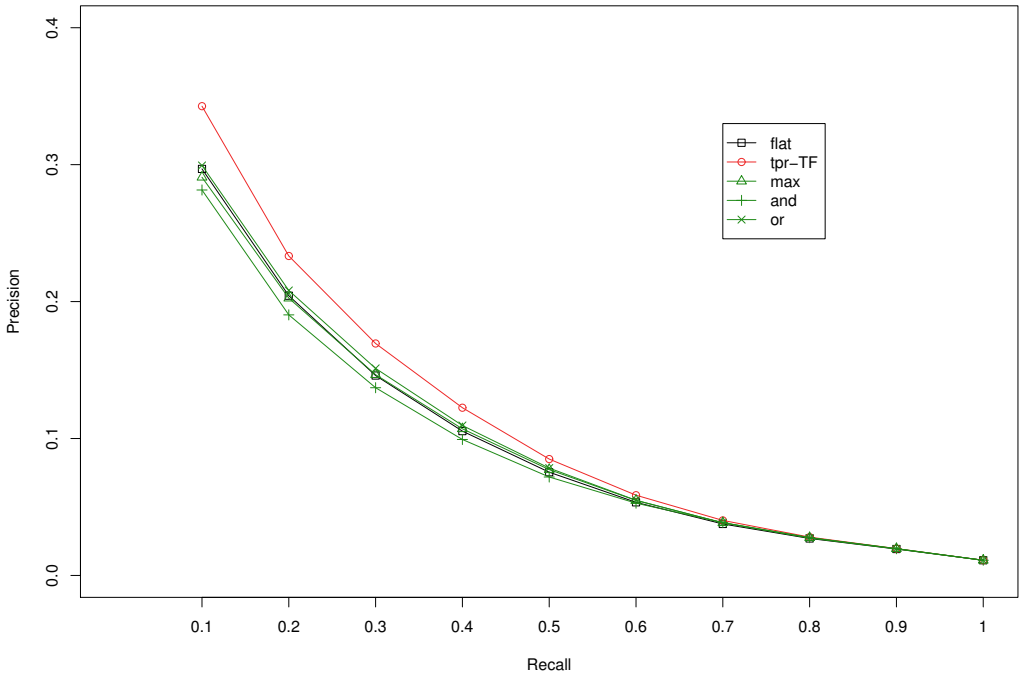


Figura 4.2: Precisione media a fissati livelli di recall tra i termini HPO utilizzando come classificatore le funzioni di score kernelizzate ad uno step (RANKS).

recall (fig. 4.2), in cui la curva del metodo TPR-TF è costantemente sopra tutte le altre curve, mostrando ancora una volta che il metodo di correzione gerarchica TPR-TF ottiene in media risultati migliori rispetto a tutti gli altri metodi comparati. Al contrario, i metodi di ensemble gerarchici euristici MAX, AND e OR, non sono in grado di migliorare le performance della predizione *Flat*, confermando i risultati precedentemente ottenuti in letteratura nel contesto della predizione genica [31].

Gli stessi esperimenti sono stati ripetuti usando il classificatore LP per processare gli score flat. I questo caso, tra i vari metodi di ensemble gerarchici TPR, la variante True Path Rule with Threshold (TPR-T) ha ottenuto risultati migliori rispetto ai metodi *Flat* sia in termini di AUC che in termini di PxR (Tabella 4.4). Il valore della soglia t_j (eq. 3.11) per il quale l'algoritmo TPR-T ha ottenuto le performance migliori, sia in termini di AUC che di P10R, P20R e P40R, è 0.9.
In particolar modo, considerando la precisione a fissati livelli di racall, la variante TPR-T ha ottenuto risultati significativamente migliori sia rispetto all'approccio *Flat* che rispetto ai tre metodi di ensemble gerarchici euristici in accordo con il test di *Wilcoxon-Mann-Whitney* ($\alpha = 10^{-5}$) (Tabella 4.4). Questi risultati sono confermati anche dalla curva precisione-recall (fig. 4.2), in cui la curva del metodo TPR-T è costantemente sopra tutte

	FLAT	TPR-T	MAX	AND	OR
AUC	0.7883	**0.7967**	0.7869	**0.7974**	0.7923
P10R	0.0673	**0.0936**	0.0653	0.0704	0.0730
P20R	0.0568	**0.0709**	0.0549	0.0564	0.0606
P40R	0.0439	**0.0503**	0.0426	0.0444	0.0462

Tabella 4.4: AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore LP. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

le altre curve, mostrando ancora una volta, che il metodo di correzione gerarchica TPR-T ottiene in media risultati migliori rispetto a tutti gli altri metodi comparati (figura 4.3).

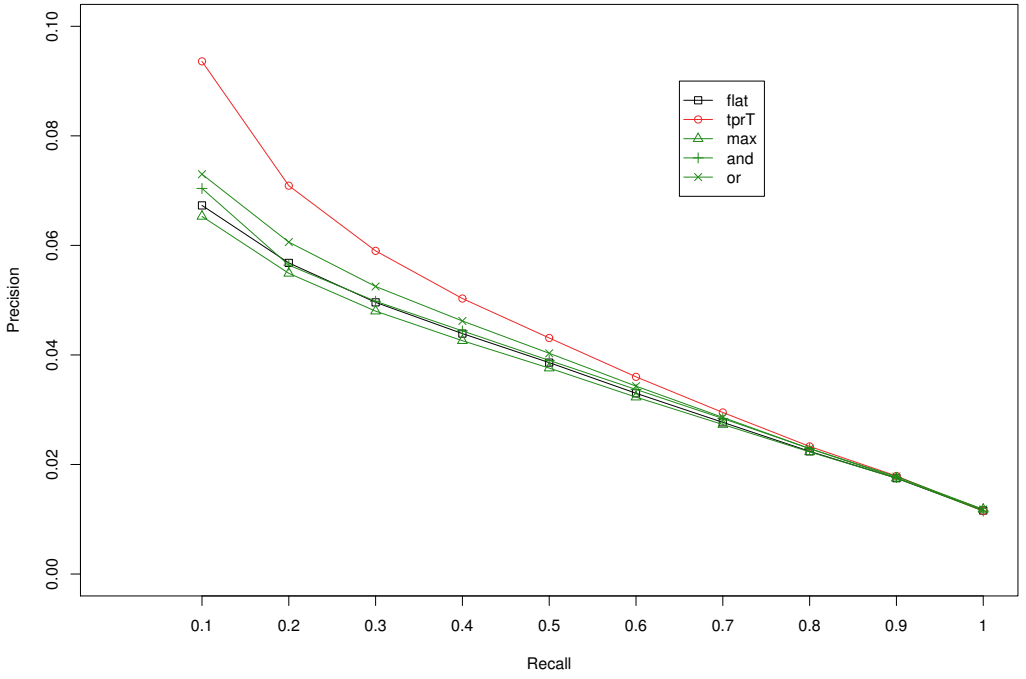


Figura 4.3: Precisione media a fissati livelli di recall tra i termini HPO utilizzando come classificatore l'algoritmo di *label propagation* (LP).

È importante notare come i valori medi assoluti di AUC e PxR ottenuti utilizzando il classificatore LP (Tabella 4.4) sono significativamente più bassi rispetto a quelli che si sono ottenuti utilizzando il classificatore RANKS (Tabella 4.3), mostrando che i risultati dei metodi di ensemble gerarchici TPR ed euristici dipendono anche dalla scelta del classificatore [31, 111]. Probabilmente i risultati poco performanti ottenuti con il *base learner* LP sono dovuti al fatto che l'algoritmo di *label propagation* prosegue fino a convergenza, esplorando troppo la topologia del grafo e considerando anche i contribuiti dei nodi tra loro molto distanti. Nonostante tutto, i metodi di ensemble TPR con il classificatore LP sono in grado di migliorare la precisione rispetto all'approccio *Flat* tra il 15% ed il 40%, almeno per livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.4). Un analogo ragionamento

non può essere applicato ai metodi di ensemble gerarchici euristici (MAX, AND, OR), confermando ancora una volta i risultati precedentemente ottenuti in letteratura nel contesto della predizione genica [31].

4.2.3 Analisi Empirica dei Tempi di Calcolo

Di seguito si riportano i tempi computazionali (in secondi) degli algoritmi gerarchici utilizzati per correggere, i due DAG-HPO presi in esame: il DAG-HPO avente termini con almeno due annotazioni (HPO2) e il DAG-HPO avente termini con almeno 50 annotazioni (HPO50). Come si può notare dalle tabelle 4.5, 4.6 e 4.7, gli algoritmi gerarchici, efficientemente implementati nel linguaggio di programmazione R, sono in grado di computare la correzione gerarchia in pochi minuti sull'intera ontologia, eseguendo le operazioni su un computer avente un processore Intel(R) Xeon(R) CPU $E5-1620$ @ 3.60GHz e una RAM di 32 Gb.

	HPO2			
	HTD	MAX	AND	OR
TEMPO(SEC)	240	296	264	164

Tabella 4.5: Tempo di calcolo degli algoritmi gerarchici. Il tempo computazionale (sec.) si riferisce alla correzione gerarchica eseguita sull'intera ontologia HPO2, DAG-HPO con 4847 termini e $20,257$ proteine per ogni termine.

	HPO50			
	TPR-TF	MAX	AND	OR
TEMPO(SEC)	114	53	48	31

Tabella 4.6: Tempo di calcolo degli algoritmi gerarchici. Il tempo computazionale (sec.) si riferisce alla correzione gerarchica eseguita sull'intera ontologia HPO50, DAG-HPO con 912 termini e $20,257$ proteine per ogni termine.

	HPO50			
	TPR-T	MAX	AND	OR
TEMPO(SEC)	108	57	50	33

Tabella 4.7: Tempo di calcolo algoritmi di ensemble gerarchici. Il tempo computazionale (sec.) si riferisce alla correzione gerarchica eseguita sull'intera ontologia HPO50, DAG-HPO con 912 termini e $20,257$ proteine per ogni termine.

4.3 Predizione Gerarchica della Funzione Proteica

La Gene Ontology (GO) [4] è composta da migliaia di classi funzionali strutturate secondo un DAG e suddivise in tre distinte ontologie che descrivono i prodotti genici in base alla loro associazione con i processi biologici (BP, *biological process*), con le funzioni molecolari (MF, molecular function) e con le componenti cellulari (CC, cellular component) in una maniera specie indipendente [2, 4]. Infatti un gene può partecipare a specifici processi biologici (e.g., ciclo cellulare, metabolismo, biosintesi dei nucleotidi) e allo stesso tempo esercitare specifiche funzioni molecolari (e.g., attività catalitiche o di legame) a livello di specifici componenti cellulari (e.g., compartimento mitocondriale o reticolo endoplasmatico).
L'ontologia BP rappresenta una serie di eventi compiuti da uno o più insiemi di funzioni molecolari che espletano una specifica funzione biologica, come ad esempio il metabolismo dei lipidi o il ciclo degli acidi tricarbossilici [2, 4].
L'ontologia MF descrive le attività che si verificano a livello molecolare, come le attività catalitiche o di legame. Un esempio di funzione molecolare può essere l'attività della glicina deidrogenasi o quella del trasportatore del glucosio [2, 4].
L'ontologia CC rappresenta semplicemente le componenti della cellula in cui è localizzato uno specifico prodotto genico. Ne sono un esempio il reticolo endoplasmatico e il ribosoma [2, 4].
Le relazioni tra i termini GO possono essere classificate nei seguenti tre gruppi:

(i) *is_ a* (relazioni di sottotipo): se il nodo A *is_ a* B, allora il nodo A è un sottotipo del nodo B. Per esempio il ciclo cellulare mitotico è un ciclo cellulare e l'attività liasica è un'attività enzimatica [2].

(i) *part_ of* (relazioni parte ed intero): se il nodo A è *parte del* nodo B, allora ogni volta che A esiste è parte di B e la presenza di A implica anche la presenza B. Per esempio il mitocondrio fa parte del citoplasma [2].

(iii) *regulates* (relazioni di controllo): se il nodo A regola il nodo B, allora A subisce la manifestazione di B, cioè il primo regola il secondo [2].

Mentre le relazioni *is_ a* e *part_ of* godono della proprietà transitiva, la relazione *regulates* non gode di questa proprietà [2]. Inoltre, poiché non è detto che le proteine regolatorie abbiano le stesse proprietà delle proteine che regolano, in alcuni casi, predire una funzione regolatoria potrebbe richiedere dati e assunzioni aggiuntive rispetto alla predizione della funzione basata sulla similarità [2]. Per questo motivo, in AFP, le relazioni *regulates*, che comunque rappresentano la minoranza tra le reazioni esistenti, solitamente non vengono

usate.
Ogni annotazione è etichetta da un *codice di evidenza* che indica come è supportata l'annotazione ad un un particolare termine. I *codici di evidenza* sono classificati in diverse categorie:

(a) *codici di evidenza sperimentali*: annotazioni supportate da prove sperimentali. Ad esempio annotazioni dedotte da interazioni fisiche (IPI, *inferred from physical interaction*), o da un fenotipo mutante (IMP, *inferred from mutant phenotype*), o da interazioni genetiche (IGI, *inferred from genetic interaction*) [2];

(b) *codici dichiarati da autori*: annotazioni effettuate basandosi su dichiarazioni di uno o più autori presenti in un citato riferimento bibliografico. Ne sono un esempio le annotazioni TAS (*traceable author statement*) [2];

(c) *codici di evidenza computazionale*: annotazioni basate su un'analisi *in silico* manualmente ricontrollate. Ne sono un esempio le annotazioni dedotte da similarità di sequenza e di struttura (ISS, *inferred from sequence or structural similarity*) [2].

Per un elenco più completo ed esauriente dei *codici di evidenza* si rimanda al sito della GO (http://geneontology.org).

4.3.1 Costruzione ed Integrazione delle Reti Funzionali Proteiche

Per questo esperimento sono state utilizzate reti di tre differenti specie, le cui caratteristiche sono descritte nella tabella 4.8 e 4.9. Più precisamente, la rete HS si riferisce alla specie *Homo sapiens*, la rete AT si riferisce alla specie *Arabidopsis thaliana* (pianta modello per antonomasia), mentre le specie totali presenti nella rete BACTERIA sono ben 301. Per la rete BACTERIA è stato utilizzato un numero così alto di specie in quanto molte di esse contavano un numero molto basso di annotazioni GO sperimentali. Inoltre, poiché i metodi di apprendimento che sono stati utilizzati sono trasduttivi, sono state considerate molteplici sorgenti di annotazioni anche al fine di aumentare le performance. Infatti più elevata è l'informazione iniziale e maggiore sarà l'efficacia nella propagazione delle annotazioni.
Nella tabella 4.8 per ogni specie considerata e per ognuna delle tre ontologie (BF, MF, CC), si descrivono il numero di termini GO (NGO) ed il numero di proteine (NPR) da cui è composta ogni ontologia di ogni rete. Nella tabella 4.9 invece si descrivono, sempre per ogni specie e ontologia, il numero di termini GO (NGO) ed il numero di relazioni tra i vari termini GO (GOREL).
In linea generale, tutte le tre reti sopracitate sono state costruite, nello stesso modo in cui è stato costruito il DAG-HPO (sottosezione 4.2). Anche in questo caso l'insieme dei

Tabella 4.8: Caratteristiche delle reti HS, AT, BACTERIA. Si rimanda al testo per ulteriori dettagli.

	HS	AT	BACTERIA
BP	NGO:8310 NPR:20257	NGO:3410 NPR:12069	NGO:2462 NPR:17638
MF	NGO:2453 NPR:20257	NGO:1676 NPR:12069	NGO:1921 NPR:17638
CC	NGO:961 NPR:20257	NGO:422 NPR:12069	NGO:210 NPR:17638

Tabella 4.9: Caratteristiche delle reti HS, AT, BACTERIA. Si rimanda al testo per ulteriori dettagli.

	HS	AT	BACTERIA
BP	NGO:8310 GOREL:18846	NGO:3410 GOREL:7294	NGO:2462 GOREL:5163
MF	NGO:2453 GOREL:3086	NGO:1676 GOREL:2068	NGO:1921 GOREL:2455
CC	NGO:961 GOREL:1834	NGO:422 GOREL:834	NGO:210 GOREL:392

geni, che può riguardare una o più specie a seconda della rete considerata, è stato fornito dai CAFA2 *organizers*. Per tutte le proteine appartenenti all'insieme dei geni relativi ad una specifica rete, sono stati costruiti dei vettori binari in cui ogni elemento rappresenta la presenza o l'assenza di una caratteristica biomolecolare estratta da una delle banche dati elencata nella tabella 4.1. Più precisamente, per ogni proteina, è stato costruito un vettore binario contenente le caratteristiche biomolecolari ottenute rispettivamente da tutte le banche dati elencate nella tabella 4.1. Unica eccezione è la GO, dalla quale sono stati ottenuti non uno, ma ben 3 vettori binari (uno per ogni ontologia GO). Questo porta ad ottenere, per ogni proteina, un totale di 10 vettori binari. Questi vettori binari sono stati successivamente utilizzati per costruire $n = 10$ reti di similarità funzionale calcolando il coefficiente di Jaccard tra tutte le possibili coppie di vettori. Si noti che non è garantita l'esistenza del vettore binario $v_{i,j}$ dove i rappresenta la i-esima proteina e j il j-esimo database. Infatti nel caso in cui la proteina i non abbia nessuna annotazione nella banca dati j il vettore non esiste. Lo step successivo consiste nel combinare le n reti d'interazione genica, mediante la tecnica di *Unweighted Average* (UA, eq. 4.6), al fine di ottenere, un'unica rete integrata per ogni insieme di proteine. Si noti che nel processo d'integrazione per costruire, ad esempio, la rete dei termini GO-BP, non è stata integrata la rete di similarità di Jaccard ottenuta dai vettori GO-BP, ma sono state integrate quelle di GO-MF e GO-CC. Questo è stato fatto per non introdurre un ulteriore *bias* nella predizioni delle etichette dell'ontologia GO-BP. I vettori binari GO-BP che non sono stati integrati, sono stati utilizzati per la fase di apprendimento. Un ragionamento del tutto analogo è stato applicato anche alle ontologie MF e CC.
In questo modo per ogni insieme di proteine si sono ottenute non una ma ben 3 reti integrate: una per GO-BP, una per GO-MF ed una per GO-CC. Ovviamente per ogni rete è stata ottenuta anche la corrispondente matrice delle etichette.

In linea generale questo è lo schema di base che è stato seguito per costruire la rete integrata e la matrice delle etichette di ognuna delle reti utilizzate per questo esperimento. Solamente nel processo d'integrazione della rete HS, sono state incluse anche altre due reti funzionali (le reti FI e HumanNet della tabella 4.1).

4.3.2 Risultati

Innanzitutto, per ognuna delle tre reti analizzate (HS, AT, BACTERIA) e per ogni ontologia GO (BP, MF, CC), gli score flat sono stati processati utilizzando funzioni di score kernelizzate ad 1-step (classificatore RANKS) e applicando la tecnica di validazione statistica *leave-one-out*. Successivamente gli score flat, sono stati corretti in accordo con la gerarchia del grafo, applicando l'algoritmo HTD (sottosezione 3.2), le diverse varianti dell'algoritmo TPR-DAG (sottosezione 3.3) e i metodi euristici di Obozinski (MAX, AND, OR). Tra tutte le varianti TPR implementate è stata scelta quella con le performance migliori. Per ogni ontologia di tutte le reti analizzate, la variate gerarchica Weighted True Path Rule (TPR-W) è stata quella che ha ottenuto le performance migliori sia in termini di AUC che di PxR. Infine i vari metodi gerarchici sono stati comparati con l'approccio *Flat* al fine di verificare se ci sono stati miglioramenti nelle performance: più le performance dei metodi gerarchici si discostano da quelle dell'approccio *Flat* e migliore sarà l'accuratezza delle predizioni dei metodi gerarchici rispetto alla strategia *Flat*.

Incominciamo col prendere in esame l'ontologia BP della rete HS. Nella tabella 4.10 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante Weighted True Path Rule (TPR-W), e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare nella tabella 4.10, l'algoritmo HTD ottiene sempre

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.8010	**0.8100**	0.8094	0.7861	**0.8175**	0.7942
P10R	0.0901	**0.1947**	0.1942	0.0384	0.1256	0.0527
P20R	0.0901	**0.1744**	0.1738	0.0390	0.1086	0.0561
P40R	0.0832	**0.1364**	0.1356	0.0399	0.0837	0.0566

Tabella 4.10: Rete: **HS**. Ontologia: **BP**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

risultati significativamente migliori rispetto all'approccio *Flat* sia in termini di AUC che di PxR. Anche se la differenza in favore di HTD è piccola (0.810 vs 0.801), considerando l'AUC medio di ognuno dei 8309 termini, l'algoritmo HTD migliora i risultati rispetto all'approccio *Flat* in 5546, ottiene lo stesso risultato in 2278 termini e ottiene risultati

peggiori in 485 termini. Questo significa che nel 94% dei termini GO-BP si ha un miglioramento delle performance e questo spiega anche perché, in accordo con il test di *Wilcoxon-Mann-Whitney*, la differenza tra i due metodi è statisticamente significativa al livello di significatività $\alpha = 10^{-5}$. Risultati nettamente migliori si ottengono considerando la precisione a fissati livelli di recall. Infatti in questo caso l'algoritmo HTD è in grado di migliorare la precisione rispetto all'approccio *Flat* tra il 64% e il 116%, almeno per i livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.10). Inoltre, il metodo HTD ottiene risultati significativamente migliori anche rispetto agli altri metodi di ensemble gerarchici, fatta eccezione per la sola AUC, in cui il miglior risultato è dato dal metodo gerarchico AND (Tabella 4.10). Tuttavia la differenza è statisticamente significativa rispetto all'approccio flat, mentre non è statisticamente significativa rispetto al metodo HTD.

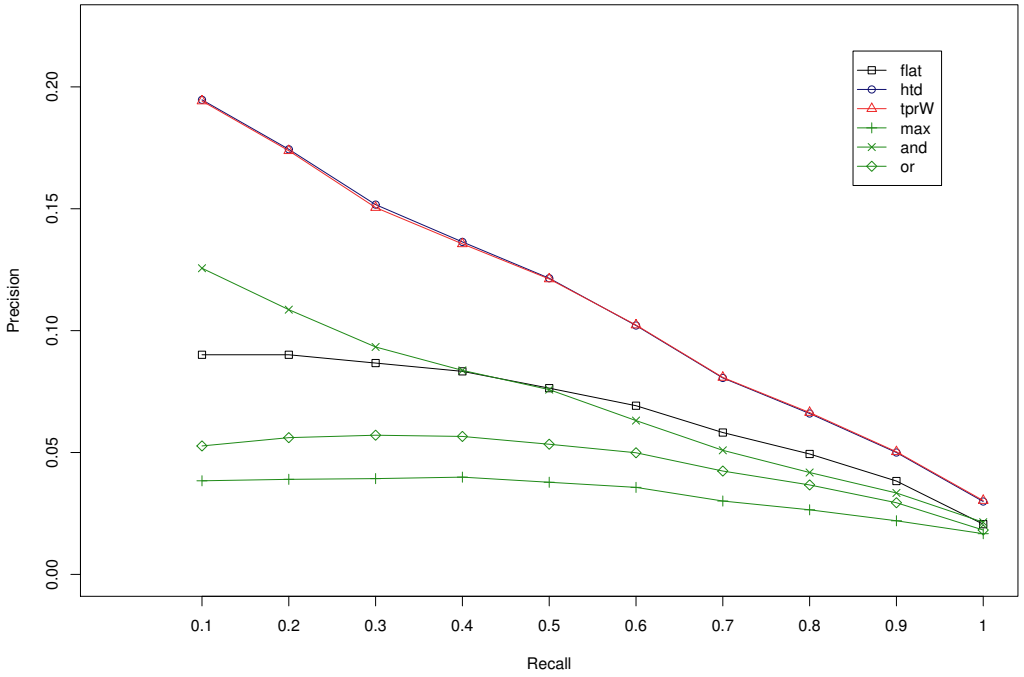


Figura 4.4: Rete: HS. Ontologia: BP. Precisione media a fissati livelli di recall.

Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.4), in cui è facile osservare come la curva del metodo HTD e quella della variante gerarchica TPR-W (che assume un andamento praticamente equivalente a quella di HTD) si trovino abbondantemente più in alto rispetto alla curva del metodo *Flat* e a quelle dei metodi gerarchici euristici MAX, AND e OR, mostrando che mediamente ottengono risultati migliori rispetto a tutti gli altri metodi comparati. Invece i metodi di ensemble gerarchici euristici MAX e OR non sono i grado di migliorare l'accuratezza delle predizioni rispetto all'approccio *Flat* a nessun livello di recall, confermando i risultati ottenuti in letteratura nel contesto della predizione genica [31]. Al contrario, il metodo gerarchico euristico AND riesce a migliorare la precisione rispetto al metodo *Flat* fino al livello di recall di 0.4 e più precisamente riesce

a incrementare la precisione tra il 29% ed il 39% per valori di recall compresi tra 0.1 e 0.2 (Tabella 4.10).

Consideriamo ora l'ontologia MF della rete HS. Nella tabella 4.11 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.7469	**0.7502**	0.7500	0.7423	**0.7543**	0.7444
P10R	0.0969	**0.2080**	**0.2093**	0.0707	0.1542	0.0779
P20R	0.0966	**0.1981**	**0.1980**	0.0712	0.1397	0.0816
P40R	0.0987	**0.1719**	0.1708	0.0738	0.1188	0.0872

Tabella 4.11: Rete: **HS**. Ontologia: **MF**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

tabella 4.11, il metodo HTD ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat* e ai metodi gerarchici euristici, sia in termini di AUC che di PxR, fatta eccezione per la sola AUC, in cui il migliore risultato è dato dal metodo euristico AND. Tuttavia la differenza è statisticamente significativa rispetto all'approccio *Flat*, mentre non è statisticamente significativa rispetto al metodo HTD. Per le metriche PxR, tra il metodo

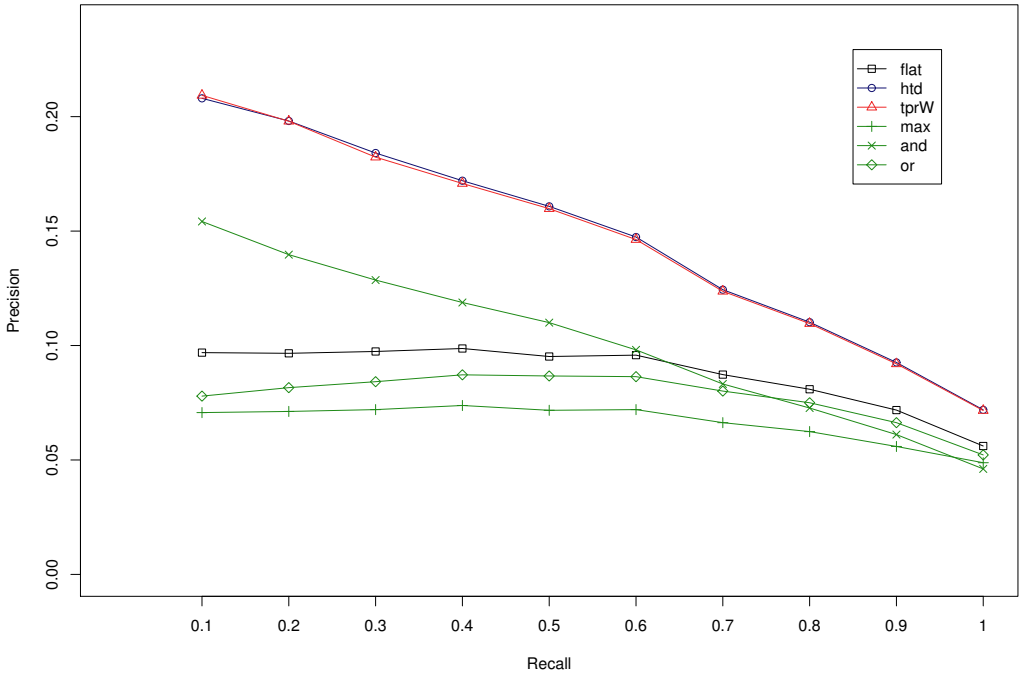


Figura 4.5: Rete: HS. Ontologia: MF. Precisione media a fissati livelli di recall.

HTD e la variante ensemble TPR-W, non si è registrata nessuna differenza statisticamente significativa, mentre è significativa la loro differenza rispetto all'approccio *Flat*. Questi

risultati sono confermati anche dalle curve precisione-recall in cui si può chiaramente vedere come le curve del metodo HTD e della variante ensemble TPR-W, che praticamente assumono lo stesso andamento, prevalgono di misura sulla curva del metodo *Flat* e su quelle dei metodi di ensemble gerarchici euristici. In particolare, stando ai dati mostrati nella tabella 4.11, il metodo HTD e la variante gerarchica TPR-W riescono ad aumentare la precisione rispetto al metodo *Flat* tra il 74% ed il 116% per i livelli di recall compresi tra 0.1 e 0.4. È interessante notare anche come il metodo gerarchico euristico AND migliori l'accuratezza delle predizioni fino al livello di recall di 0.6 e più precisamente tra il 20% ed il 59% per valori di recall compresi tra 0.1 e 0.4 (Tabella 4.11). Al contrario i metodi di ensemble gerarchici euristici MAX e OR, non sono in grado di migliorare le precisione rispetto all'approccio *Flat* a nessun livello di recall (Figura 4.5), confermando nuovamente i risultati ottenuti in letteratura nel contesto della predizione genica [31].

Consideriamo ora l'ontologia CC della rete HS. Nella tabella 4.12 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.8610	**0.8634**	**0.8634**	0.8566	0.8559	0.8582
P10R	0.1836	**0.3764**	**0.3830**	0.1047	0.2270	0.1267
P20R	0.1930	**0.3607**	**0.3648**	0.1190	0.2135	0.1442
P40R	0.1965	**0.3221**	**0.3261**	0.1309	0.1789	0.1563

Tabella 4.12: Rete: **HS**. Ontologia: **CC**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

tabella 4.12 la variante TPR-W ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat* e ai tre metodi euristici sia in termini di AUC che di P10R, P20R P40R, mentre non si è registrata nessuna differenza statisticamente significativa, in accordo con il test di *Wilcoxon-Mann-Whitney*, rispetto al metodo HTD. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.6), in cui appare evidente come la curva della variante gerarchica TPR-W e quella del metodo HTD, con la prima leggermente sopra la seconda, si trovino abbondantemente più in alto rispetto alla curva del metodo *Flat* e a quelle dei metodi gerarchici euristici MAX, AND e OR, mostrando che mediamente ottengono risultati significativamente migliori rispetto a tutti gli altri metodi comparati. Sempre in figura 4.6, appare evidente come mentre i metodi di ensemble gerarchici euristici MAX e OR non sono capaci di migliorare l'accuratezza delle predizioni rispetto al metodo *Flat* a nessun livello di recall, il metodo euristico AND è in grado di migliorare la precisione

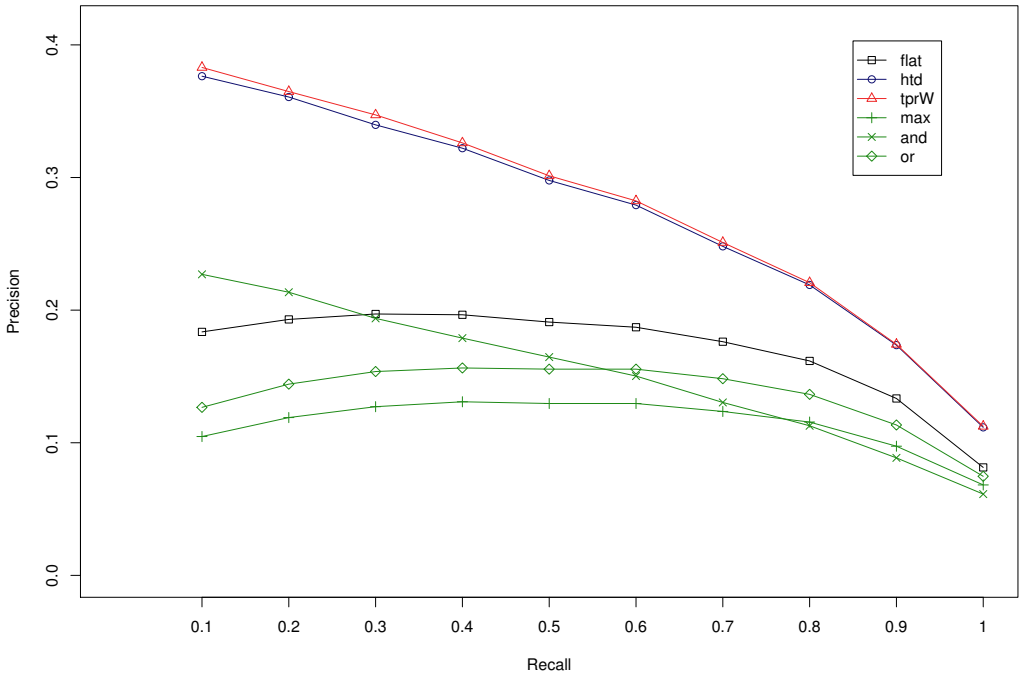


Figura 4.6: Rete: HS. Ontologia: CC. Precisione media a fissati livelli di recall.

in modo significativo a bassi livelli di racall. In particolare tra i livelli di recall compresi tra 0.1 e 0.2 migliora la precisione tra il 10% ed il 24% (Tabella 4.12).

Dal momento che la variante TPR-W è parametrica, nella tabella 4.13, si riporta per ogni ontologia della rete HS, il valore del parametro w (eq. 3.12) per il quale l'algoritmo TPR-W ha ottenuto le performance riportate nelle tabelle 4.10, 4.11 e 4.12.

	AUC	**P10R**	**P20R**	**P40R**
BP	0.9	0.9	0.9	0.9
MF	0.9	0.9	0.9	0.9
CC	0.9	0.8	0.8	0.8

Tabella 4.13: Migliore valore del parametro w per il quale la variante TPR-W ha ottenuto le performance migliori rispettivamente per l'ontologia BP, MF e CC della rete HS.

Passiamo ora ad analizzare l'ontologia BP della rete AT. Nella tabella 4.14 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare nella tabella 4.14 anche se la differenza in favore di HTD è piccola (0.8225 vs 0.8314), considerando l'AUC medio di ognuno dei 3409 termini, l'algoritmo HTD migliora i risultati rispetto all'approccio *Flat* in 2170, ottiene lo stesso risultato in 942 termini e ottiene risultati peggiori in 297 termini. Questo significa che nel 91% dei termini GO-BP si ha un miglioramento delle performance e questo spiega anche perché, in accordo con il test di *Wilcoxon-Mann-Whitney*, la differenza tra i due metodi è statisticamente significativa al

	FLAT	HTD	TPR-W	MAX	AND	OR
AUC	0.8134	**0.8225**	0.8219	0.7966	0.8217	0.8056
P10R	0.1950	**0.3261**	**0.3268**	0.0952	0.2339	0.1090
P20R	0.1803	**0.2932**	**0.2942**	0.0893	0.2053	0.1090
P40R	0.1537	**0.2363**	**0.2364**	0.0821	0.1623	0.1075

Tabella 4.14: Rete: AT. Ontologia: BP. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando come classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

livello di significatività $\alpha = 10^{-5}$. Risultati nettamente migliori si ottengono considerando la precisione a fissati livelli di recall, dove la variante gerarchica TPR-W è in grado di migliorare la precisione in modo significativo rispetto all'approccio *Flat* tra il 54% e il 68%, almeno per i livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.14). Mentre per le

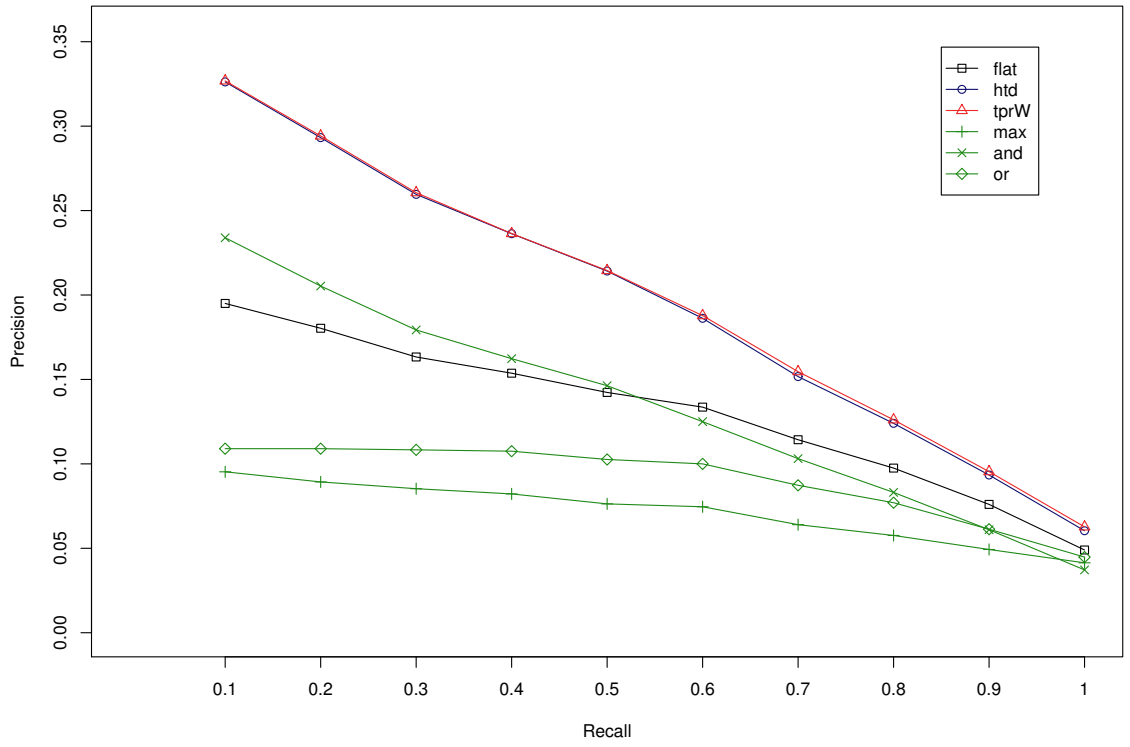


Figura 4.7: Rete: AT. Ontologia: BP. Precisione media a fissati livelli di recall.

varie performance PxR, tra la variante TPR-W e il metodo HTD, non è stata registrata alcuna differenza statisticamente significativa, per la metrica AUC il risultato migliore e statisticamente significativo è restituito dal metodo HTD. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.7), in cui è facile osservare come la curva della variante gerarchica TPR-W e quella del metodo HTD (che praticamente assumono lo stesso andamento) si trovino abbondantemente più in alto rispetto alla curva del metodo *Flat* e a quelle dei metodi gerarchici euristici MAX, AND e OR, mostrando che mediamente

ottengono risultati migliori rispetto a tutti gli altri metodi comparati. Invece i metodi di ensemble gerarchici euristici MAX e OR non sono i grado di migliorare l'accuratezza delle predizioni rispetto all'approccio *Flat* a nessun livello di recall, confermando i risultati ottenuti in letteratura nel contesto della predizione genica [31]. Al contrario, il metodo gerarchico euristico AND riesce a migliorare la precisione rispetto al metodo *Flat* tra il 6% ed il 20% per valori di recall compresi tra 0.1 e 0.4 (Tabella 4.14).

Analizziamo ora l'ontologia MF della rete AT. Nella tabella 4.15 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla tabel-

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.7347	**0.7370**	**0.7370**	0.7306	0.7375	0.7324
P10R	0.1584	**0.2670**	**0.2682**	0.1267	0.1922	0.1289
P20R	0.1578	**0.2573**	**0.2567**	0.1251	0.1831	0.1313
P40R	0.1613	**0.2366**	**0.2367**	0.1302	0.1622	0.1414

Tabella 4.15: Rete: **AT**. Ontologia: **MF**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

la 4.15, per tutte le metriche di performance considerate, i migliori risultati sono dati dal metodo HTD e dalla variante gerarchica TPR-W. Tra i due metodi non è stata registrata nessuna differenza statisticamente significativa. Questi risultati sono confermati anche

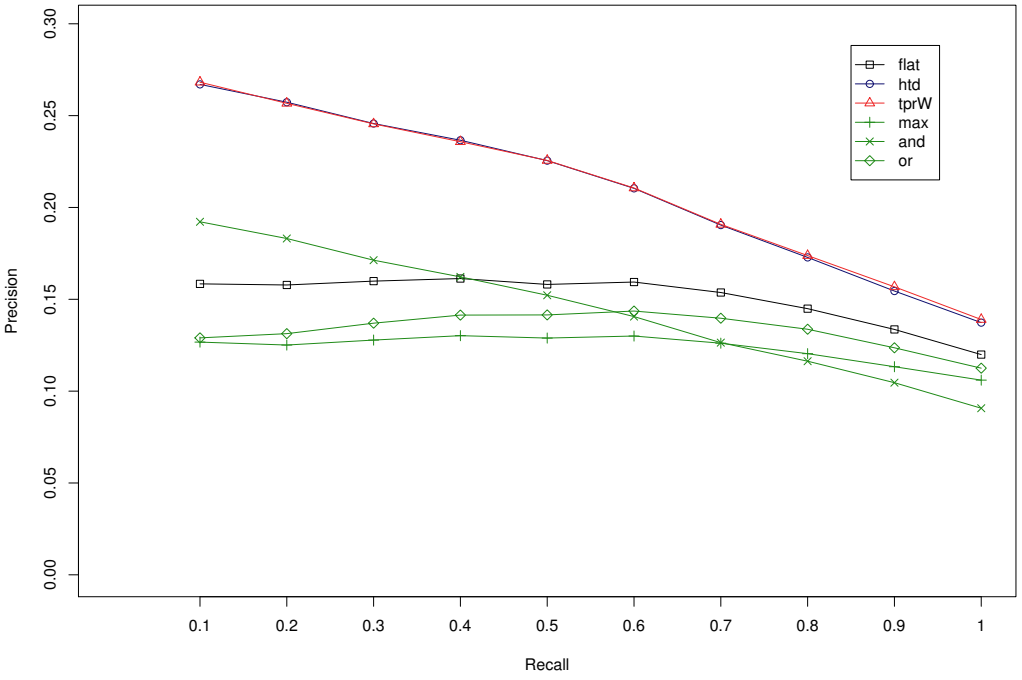


Figura 4.8: Rete: AT. Ontologia: MF. Precisione media a fissati livelli di recall.

dalle curve precisione-recall (Figura 4.8), in cui si può chiaramente osservare come la curva della variante gerarchica TPR-W e quella metodo HTD (le due curve sono sovrapposte) prevalgono nettamente su tutti gli altri metodi comparati, mostrando che mediamente ottengono risultati migliori rispetto a quelli di tutti gli altri metodi comparati. In particolare, facendo riferimento anche ai dati mostrati nella tabella 4.15, il metodo HTD e la variante gerarchica TPR-W sono in grado di aumentare la precisione rispetto all'approccio *Flat* tra il 47% ed il 68% almeno per i valori di recall compresi tra 0.1 e 0.4. Sempre in figura 4.8, appare evidente come, mentre i metodi di ensemble gerarchici euristici MAX e OR non sono capaci di migliorare l'accuratezza delle predizioni rispetto al metodo *Flat* a nessun livello di recall, il metodo euristico AND è in grado di migliorare la precisione per bassi livelli di racall, più precisamente fino al livello di recall 0.3. In particolare tra i livelli di recall compresi tra 0.1 e 0.2 migliora la precisione tra il 16% ed il 21% (Tabella 4.15).

Consideriamo ora l'ontologia CC della rete AT. Nella tabella 4.16 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.8513	**0.8531**	**0.8532**	0.8469	0.8379	0.8480
P10R	0.2917	**0.5188**	**0.5297**	0.2162	0.3414	0.2344
P20R	0.2986	**0.5081**	**0.5167**	0.2329	0.3312	0.2528
P40R	0.3167	**0.4704**	**0.4927**	0.2598	0.2969	0.2821

Tabella 4.16: Rete: **AT**. Ontologia: **CC**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

tabella 4.16, la variante TPR-W ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat* e ai metodi di ensemble euristici sia in termini di AUC che di PxR, mentre non si è registrata nessuna differenza statisticamente significativa rispetto al metodo HTD. Ottimi risultati si ottengono anche considerando la precisione a fissati livelli di recall, in cui la variante gerarchica TPR-W è in grado di aumentare la precisione rispetto al metodo *Flat* tra il 55% e 81% almeno per livelli di recall compresi tra 0.1 e 0.4. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.9), in cui appare evidente come la curva della variante gerarchica TPR-W, che si trova leggermente più sopra rispetto a quella del metodo HTD, sia abbondantemente più in alto rispetto alla curva del metodo *Flat* e a quelle dei metodi gerarchici euristici MAX, AND e OR, mostrando che mediamente ottengono risultati significativamente migliori rispetto a tutti gli altri metodi comparati. Sempre in figura 4.9, si osserva anche come, mentre i metodi di ensemble gerarchici euristici MAX e OR non sono in grado di migliorare l'accuratezza delle predizioni rispetto al metodo

Flat a nessun livello di recall, il metodo euristico AND migliora la precisione solamente a bassi livelli di recall (tra 0.1 e 0.2), ma tale differenza non è statisticamente significativa in accordo con il test di *Wilcoxon-Mann-Whitney* (Tabella 4.16).

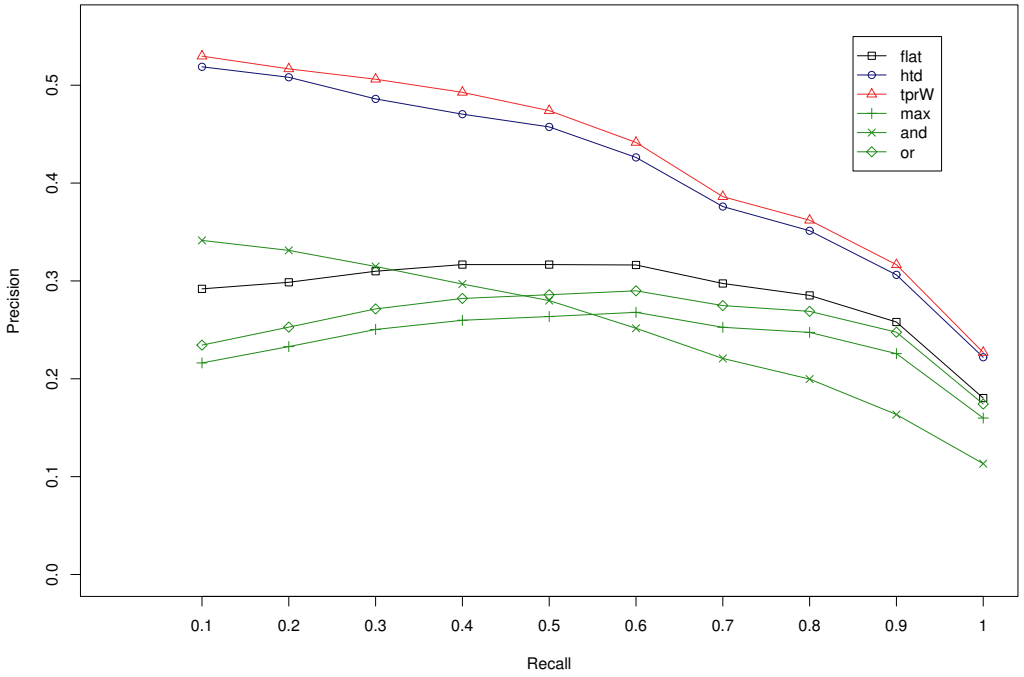


Figura 4.9: Rete: AT. Ontologia: CC. Precisione media a fissati livelli di recall.

Ricordando che la variante TPR-W è parametrica, nella tabella 4.21 si riporta, per ogni ontologia della rete AT, il valore del parametro w (eq. 3.12) per il quale l'algoritmo TPR-W ha ottenuto le performance riportate nelle tabelle 4.14, 4.15 e 4.16.

	AUC	**P10R**	**P20R**	**P40R**
BP	0.9	0.9	0.9	0.9
MF	0.9	0.8	0.8	0.9
CC	0.9	0.7	0.7	0.7

Tabella 4.17: Migliore valore del parametro w per il quale la variante TPR-W ottiene le performance migliori rispettivamente per l'ontologia BP, MF e CC della rete AT.

Consideriamo ora l'ontologia BP della rete BACTERIA. Nella tabella 4.18 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare nella tabella 4.18, l'algoritmo HTD ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat* sia in termini di AUC che di PxR. Anche se la differenza in favore di HTD è piccola (0.7914 vs 0.7885), considerando l'AUC medio di ognuno dei 2461 termini, l'algoritmo HTD migliora i risultati rispetto all'approccio *Flat* in 1616, ottiene lo stesso risultato in 774 termini e ottiene risultati peggiori in 71 termini. Questo significa che nel

	FLAT	HTD	TPR-W	MAX	AND	OR
AUC	0.7885	**0.7914**	0.7912	0.7833	0.7907	0.7854
P10R	0.1617	**0.3749**	0.3740	0.0683	0.2900	0.0744
P20R	0.1650	**0.3654**	0.3644	0.0774	0.2636	0.0900
P40R	0.1611	**0.3263**	**0.3267**	0.0842	0.2059	0.1052

Tabella 4.18: Rete: BACTERIA. Ontologia: BP. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

97% dei termini GO-BP si ha un miglioramento delle performance e questo spiega anche perché, in accordo con il test di *Wilcoxon-Mann-Whitney*, la differenza tra i due metodi è statisticamente significativa al livello di significatività $\alpha = 10^{-5}$. Risultati notevolmente migliori si ottengono considerando la precisione a fissati livelli di recall. Infatti il metodo HTD migliora la precisione rispetto all'approccio *Flat* tra 103% e il 132% almeno per i livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.18). Il metodo HTD ottiene risultati significativamente migliori anche rispetto agli altri metodi di ensemble gerarchici considerati, eccetto per la metrica di performance P40R, dove tra il metodo HTD e la variante ensemble TPR-W non si è registrata nessuna differenza statisticamente significativa (Tabella 4.18). Inoltre, è interessante notare come il metodo AND, per i livelli di recall compresi tra 0.1 e

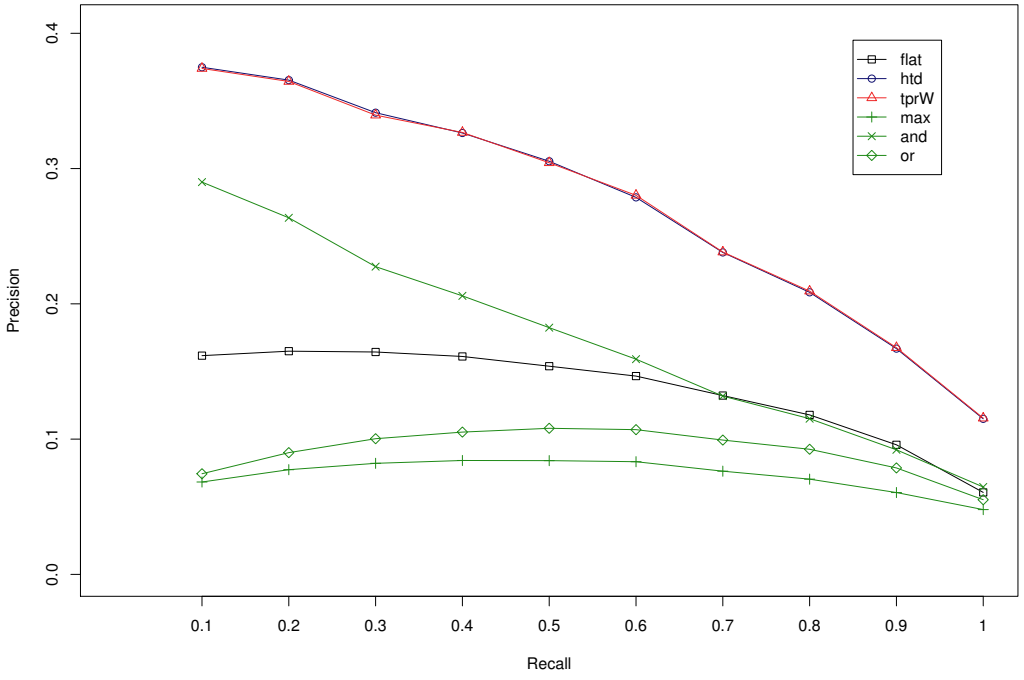


Figura 4.10: Rete: BACTERIA. Ontologia: BP. Precisione media a fissati livelli di recall.

0.4, è in grado di migliorare l'accuratezza delle predizioni rispetto all'approccio *Flat* tra il 30% e il 79% (Tabella 4.18). Questi risultati sono confermati anche dalla curva precisione-

recall (Figura 4.10), in cui è facile osservare come la curva del metodo HTD e della variante gerarchica TPR-W, che assumono lo stesso andamento, sono abbondantemente più in alto rispetto alle curve di tutti gli altri metodi, sottolineando ancora una volta come le strategie HTD e TPR ottengono mediamente risultati migliori rispetto agli altri metodi. Dalla figura 4.10 è facile osservare anche come i metodi di ensemble gerarchici euristici MAX e OR, non sono in grado di migliorare la precisione rispetto alla predizione *Flat*, mentre il metodo AND migliora in modo significativo la precisione rispetto al metodo *Flat* fino al livello di recall 0.6.

Analizziamo ora l'ontologia MF della rete BACTERIA. Nella tabella 4.19 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante TPR-W, e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.6630	**0.6644**	0.6643	0.6608	**0.6650**	0.6617
P10R	0.0919	**0.2652**	0.2621	0.0459	0.1974	0.0495
P20R	0.0906	**0.2597**	**0.2574**	0.0489	0.1886	0.0544
P40R	0.0924	**0.2432**	0.2398	0.0577	0.1608	0.0661

Tabella 4.19: Rete: **BACTERIA**. Ontologia: **MF**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

tabella 4.19 il metodo HTD ottiene sempre risultati significativamente migliori rispetto all'approccio *Flat*, sia in termini di AUC che di P10R, P20R e P40R. Il metodo HTD ottiene risultati significativamente migliori anche rispetto agli altri metodi di ensemble gerarchici. Tuttavia, non si sono registrate differenze statisticamente significative tra il metodo HTD e il metodo euristico AND per la metrica di performance AUC e tra HTD e la variante gerarchica TPR-W per la metrica P20R. Risultati significativi si ottengono considerando la precisione a fissati livelli di recall, dove il metodo HTD riesce a migliorare l'accuratezza delle predizioni rispetto all'approccio *Flat* tra il 163% e il 189%, almeno per i livelli di recall compresi tra 0.1 e 0.4 (Tabella 4.19). Inoltre, è interessante notare come anche il metodo di ensemble gerarchico euristico AND è in grado di migliorare la precisione rispetto al metodo *Flat* tra il 74% e il 115%, almeno per i livelli di recall compresi tra 0.1 e 0.2. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.11), in cui è facile osservare come la curva del metodo HTD e quella della variante gerarchica TPR-W, con la curva del primo metodo leggermente più sopra quella del secondo, sono ampiamente più in alto rispetto a quelle di tutti gli altri metodi, evidenziando ancora una volta come il metodo HTD ottiene in media risultati significativamente migliori.

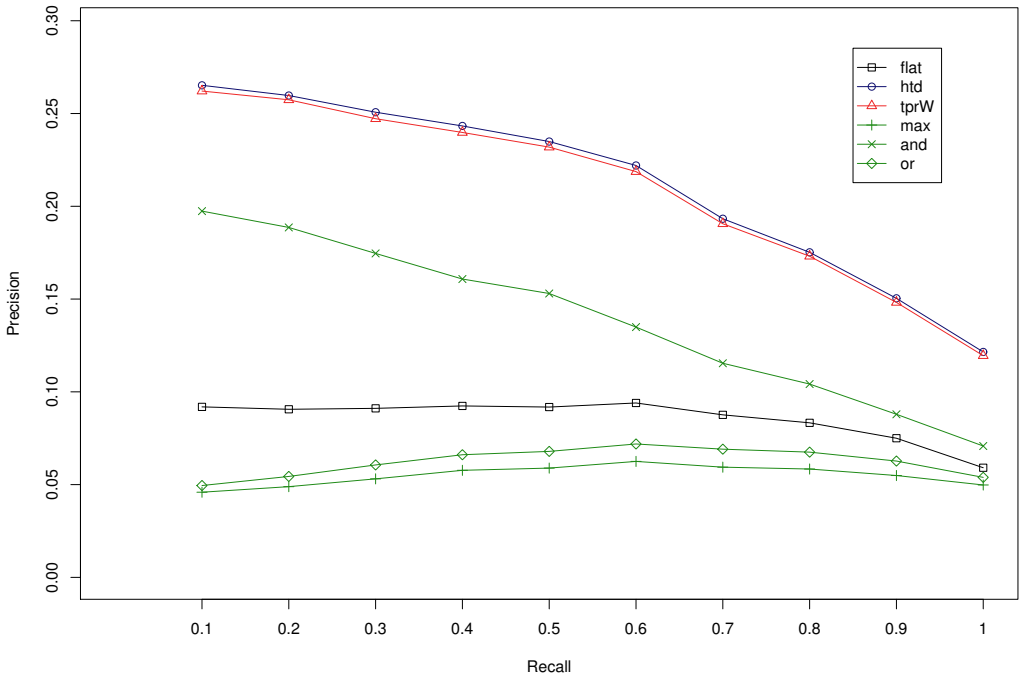


Figura 4.11: Rete: BACTERIA. Ontologia: MF. Precisione media a fissati livelli di recall.

Inoltre dalla figura 4.11 appare evidente come il metodo di ensemble gerarchico euristico AND riesca a migliorare la precisione rispetto all'approccio *Flat* in tutti i livelli di recall, mentre i metodi di ensemble gerarchici euristici MAX e OR, non sono in grado di migliorare l'accuratezza delle predizioni rispetto alla predizione *Flat*, confermando ancora una volta i risultati precedentemente ottenuti in letteratura nel contesto della predizione genica [31].

Prendiamo in esame ora l'ontologia CC della rete BACTERIA. Nella tabella 4.20 si riportano i risultati medi per termine ottenuti per il metodo gerarchico HTD, per la variante Weighted True Path Rule (TPR-W), e per i metodi di ensemble gerarchici euristici MAX, AND e OR. Come si può notare dalla tabella 4.20, la variante gerarchica TPR-W ottiene sempre risultati

	FLAT	**HTD**	**TPR-W**	**MAX**	**AND**	**OR**
AUC	0.8599	**0.8612**	**0.8612**	0.8583	0.8585	0.8589
P10R	0.3817	**0.5273**	**0.5591**	0.2406	0.4467	0.2490
P20R	0.3909	**0.5217**	**0.5579**	0.2685	0.4358	0.2844
P40R	0.3949	**0.4964**	**0.5189**	0.2944	0.3674	0.3218

Tabella 4.20: Rete: **BACTERIA**. Ontologia: **CC**. AUC medio e precisione media a 10%, 20%, 40% di recall (P10R, P20R e P40R), utilizzando il classificatore RANKS. I metodi che sono significativamente migliori rispetto agli altri in accordo con il test di *Wilcoxon-Mann-Whitney* sono evidenziati in grassetto ($\alpha = 10^{-5}$).

significativamente migliori rispetto all'approccio *Flat*, sia in termini di AUC che di P10R, P20R e P40R e anche rispetto ai metodi di ensemble gerarchici euristici, mentre non si è registrata nessuna differenza statisticamente significativa rispetto al metodo di ensemble

HTD. Si ottengono ottimi risultati considerano anche la precisione a fissati livelli di recall, dove la variante gerarchica TPR-W migliora la precisione rispetto all'approccio *Flat* tra il 46% e il 31% almeno per i livelli di recall compresi tra 0.1 e 0.4. Questi risultati sono confermati anche dalle curve precisione-recall (Figura 4.12), dove è facile osservare come la curva della variante gerarchica TPR-W sia leggermente più in alto rispetto a quella del metodo HTD, per tutti i livelli di recall, e abbondantemente più in alto rispetto a metodi euristici e all'approccio *Flat*. Inoltre dalla figura 4.12 si osserva anche come il

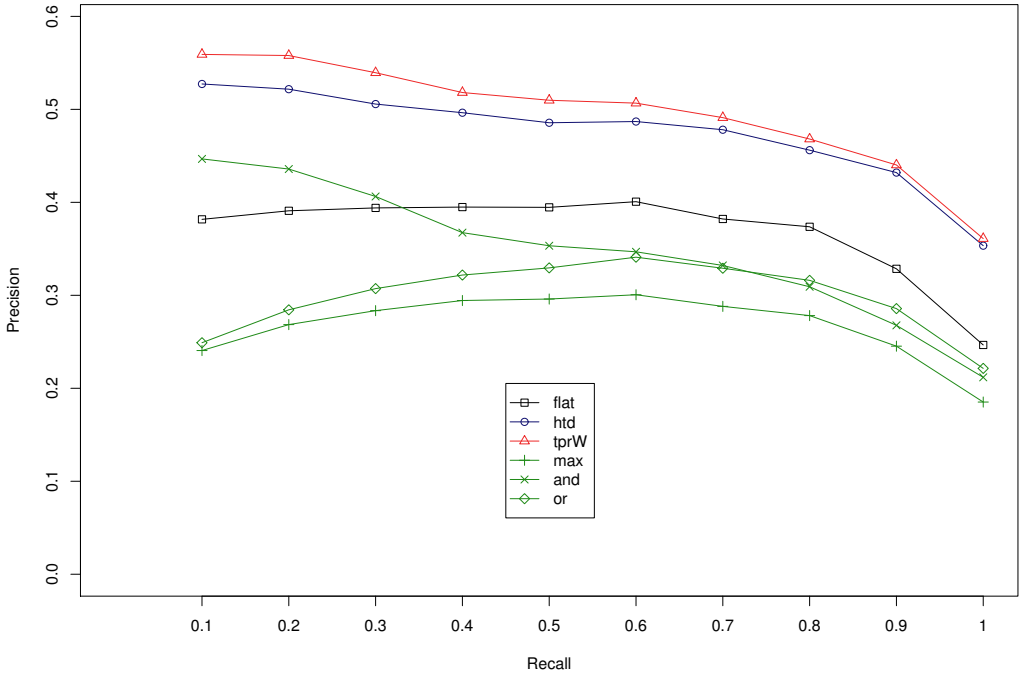


Figura 4.12: Rete: BACTERIA. Ontologia: CC. Precisione media a fissati livelli di recall.

metodo gerarchico AND sia in grado di migliorare la precisione rispetto al metodo *Flat* del 10% fino al livello di recall 0.2, ma tale differenza non è statisticamente significativa in accordo con il test di *Wilcoxon-Mann-Whitney*. I metodi di ensemble gerarchici euristici MAX e OR, non sono in grado di migliorare la precisione rispetto alla predizione *Flat* a nessun livello di recall, confermando i risultati precedentemente ottenuti in letteratura nel contesto della predizione genica [31].

Nella tabella 4.21 si riporta, per ogni ontologia della rete BACTERIA, il valore del parametro w (eq. 3.12) per il quale l'algoritmo TPR-W ha ottenuto le performance riportate nelle tabelle 4.18, 4.19 e 4.20.

	AUC	**P10R**	**P20R**	**P40R**
BP	0.9	0.9	0.9	0.9
MF	0.9	0.9	0.9	0.9
CC	0.9	0.6	0.6	0.5

Tabella 4.21: Migliore valore del parametro w per il quale la variante TPR-W ottiene le performance migliori rispettivamente per l'ontologia BP, MF e CC della rete BACTERIA

4.3.3 Analisi Empirica dei Tempi di Calcolo

Di seguito si riportano i tempi computazionali (in secondi) degli algoritmi gerarchici utilizzati per correggere i DAG-GO delle tre reti prese in esame. Come si può vedere dalle tabelle 4.22, 4.23 e 4.24 gli algoritmi gerarchici, efficientemente implementati nel linguaggio di programmazione R, sono in grado di computare la correzione gerarchica sull'intera ontologia, eseguendo le operazioni su un calcolatore avente un processore Intel(R) Xeon(R) CPU $E5-1620$ @ 3.60GHz e una RAM da 32 Gb.

	HTD	TPR-W	MAX	AND	OR
BP	457	1230	667	636	510
MF	131	223	150	149	68
CC	50	92	64	64	38

Tabella 4.22: Tempo computazionale degli algoritmi gerarchici di ogni ontologia della rete HS.

	HTD	TPR-W	MAX	AND	OR
BP	110	246	156	156	105
MF	54	92	64	16	26
CC	14	28	17	56	11

Tabella 4.23: Tempo computazionale degli algoritmi gerarchici di ogni ontologia della rete AT

	HTD	TPR-W	MAX	AND	OR
BP	110	250	145	144	106
MF	84	18	98	94	40
CC	9	141	11	10	6

Tabella 4.24: Tempo computazionale degli algoritmi gerarchici di ogni ontologia della rete BACTERIA.

Conclusioni

Svariati problemi, che vanno dalla classificazione dei testi alla predizioni delle funzioni proteiche, sono caratterizzati da una classificazione gerarchica multi-etichetta [111]. In questo contesto i metodi *Flat* forniscono predizioni inconsistenti e più in generale non sono in grado di sfruttare i vincoli gerarchici tra le classi. Infatti, molti metodi computazionali che sfruttano dati "omici" possono essere applicati con successo alla predizione o all'ordinamento dei geni rispetto ad un fenotipo patologico di una malattia mendeliana rara (ontologia HPO) [109] o rispetto ad una funzione proteica (tassonomia GO), ma generalmente le predizioni restituite non sono consistenti con la gerarchia, in quanto non obbediscono alle relazioni gerarchiche tra i termini della tassonomia DAG-strutturata (e.g., un gene potrebbe ottenere uno score per un termine figlio maggiore rispetto a quello del suo termine genitore). Nel presente lavoro di tesi è stato mostrato come i metodi di ensemble gerarchici, in particolar modo la strategia HTD e quella TPR, forniscano predizioni che sono sempre consistenti con la *true path rule* che governa sia l'ontologia HPO che quella GO. Inoltre, dai risultati sperimentali ottenuti, è stato osservato come queste due strategie, sfruttando rispettivamente le relazioni gerarchiche tra i termini dei DAG soggiacenti rispettivamente la HPO e la GO, migliorino le predizioni "flat" indipendentemente dal tipo di classificatore utilizzato. Tuttavia predittori che forniscono uno score (e.g., RANKS) o una probabilità (e.g., LP) che un gene appartenga ad una data classe, sono più adatti sia per la strategia HTD (sottosezione 3.2) che per gli algoritmi di ensemble gerarchici TPR-DAG (sottosezione 3.3). Di conseguenza i metodi HTD e TPR, possono essere visti come uno strumento software flessibile che può essere utilizzato per migliorare le performance di un qualsiasi metodo di apprendimento *Flat*, con lo scopo di rendere le predizioni più consistenti e affidabili per tassonomie DAG-strutturate. Ad esempio, nel contesto della predizione delle associazioni gene-fenotipo patologico, predizioni consistenti con la gerarchia potrebbero rappresentare un passo importante per la scoperta di nuovi geni associati a malattie genetiche umane rare. Inoltre i metodi HTD e TPR possono essere applicati anche a tassonomie strutturate ad albero, poichè gli alberi, per definizione, sono DAGs. Lavori di ricerca futuri potrebbero sfruttare altri classificatori, inclusi quelli supervisionati, e potrebbero comparare le strategie HTD e TPR con altri metodi gerarchici, inclusi i metodi con output kernel strutturato [111]. L'efficacia dei metodi HTD e TPR potrebbe essere valutata anche in altri domini applicativi caratterizzati da tassonomie strutturate secondo DAG, come ad esempio la classificazione gerarchica di documenti nel web.

Appendice

A Implementazione dell'algoritmo HTD-DAG

```
htd <-  function(S,g, root="00"){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           maxscore <- max(S);
           z <- rep(maxscore,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        # nodes are scanned from top to bottom: a list par.tod with the parents
        # for each node (ordered from top to bottom) is obtained
        par.tod <- get.parents.top.down(g,levels,root)
        for(i in 1:length(par.tod)){
          child <- S[,names(par.tod[i])];               # child node scores
          parents <- as.matrix(S[,par.tod[[i]]]);       # parents nodes scores
          # Note: the version with an apply and an ifelse statement is slower
          for(j in 1:length(child)){
            x <- min(parents[j,]);
            if(x < child[j])
               child[j] <- x;
          }
          S[,names(par.tod[i])] <- child;
        }
        # remove the "dummy root column"
        S <- S[,-which(colnames(S)==root)];
        return(S);
}
```

Script 1: Funzione che implementa l'algoritmo HTD-DAG

B Implementazione dell'algoritmo TPR-DAG

```
tpr.threshold <- function(S,g, root="00",t=0.5){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           maxscore <- max(S);
           z <- rep(maxscore,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        # per-level bottom-up visit of the graph
        chd.bup <- get.children.bottom.up(g,levels);
        for(i in 1:length(chd.bup)){
          if(length(chd.bup[[i]])!=0){
            parent <- S[,names(chd.bup[i])];
            children <- as.matrix(S[,chd.bup[[i]]]);
            for(j in 1:length(parent)){
              child.set <- children[j,] > t;           # positive children choice
              child.pos <- children[j,][child.set];
             # flat prediction correction
              parent[j] <- (parent[j] + sum(child.pos))/(1+length(child.pos));
            }
            S[,names(chd.bup[i])] <- parent;
          }
        }
        # per-level top-down visit of the graph
        par.tod <- get.parents.top.down(g,levels,root)
        for(i in 1:length(par.tod)){
          child <- S[,names(par.tod[i])];
          parents <- as.matrix(S[,par.tod[[i]]]);
          # Note: the version with an apply and an ifelse statement is slower
          for(j in 1:length(child)){
            x <- min(parents[j,]);
            if(x < child[j]){
               child[j] <- x;    # hierarchy correction
            }
          }
          S[,names(par.tod[i])] <- child;
        }
        # remove the "dummy root column"
        S <- S[,-which(colnames(S)==root)];
        return(S);
}
```

Script 2: Funzione che implementa la variante TPR-T

```
tpr.threshold.free <- function(S,g, root="00"){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           maxscore <- max(S);
           z <- rep(maxscore,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        # per-level bottom-up visit of the graph
        chd.bup <- get.children.bottom.up(g,levels)
        for(i in 1:length(chd.bup)){
          if(length(chd.bup[[i]])!=0){
            parent <- S[,names(chd.bup[i])];
            children <- as.matrix(S[,chd.bup[[i]]]);
            for(j in 1:length(parent)){
              child.set <- children[j,] > parent[j];   # positive children choice
              child.pos <- children[j,][child.set];
              # flat prediction correction
              parent[j] <- (parent[j] + sum(child.pos))/(1+length(child.pos));
            }
            S[,names(chd.bup[i])] <- parent;
          }
        }
        # per-level top-down visit of the graph
        par.tod <- get.parents.top.down(g,levels,root)
        for(i in 1:length(par.tod)){
          child <- S[,names(par.tod[i])];
          parents <- as.matrix(S[,par.tod[[i]]]);
          # Note: the version with an apply and an ifelse statement is slower
          for(j in 1:length(child)){
            x <- min(parents[j,]);
            if(x < child[j]){
               child[j] <- x;      # hierarchy correction
            }
          }
          S[,names(par.tod[i])] <- child;
        }
        # remove the "dummy root column"
        S <- S[,-which(colnames(S)==root)];
        return(S);
}
```

Script 3: Funzione che implementa la variante TPR-TF

```
tpr.weighted.threshold <- function(S,g, root="00",t=0.5,w=0.5){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           maxscore <- max(S);
           z <- rep(maxscore,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        # per-level bottom-up visit of the graph
        chd.bup <- get.children.bottom.up(g,levels)
        for(i in 1:length(chd.bup)){
          if(length(chd.bup[[i]])!=0){
            parent <- S[,names(chd.bup[i])];
            children <- as.matrix(S[,chd.bup[[i]]]);
            for(j in 1:length(parent)){
              child.set <- children[j,] > t;      # positive children choice
              child.pos <- children[j,][child.set];
              if(length(child.pos)!=0)
               # flat prediction correction
                parent[j] <- w*parent[j] + (1-w)*sum(child.pos)/length(child.pos);
            }
            S[,names(chd.bup[i])] <- parent;
          }
        }
        # per-level top-down visit of the graph
        par.tod <- get.parents.top.down(g,levels,root)
        for(i in 1:length(par.tod)){
          child <- S[,names(par.tod[i])];
          parents <- as.matrix(S[,par.tod[[i]]]);
          # Note: the version with an apply and an ifelse statement is slower
          for(j in 1:length(child)){
            x <- min(parents[j,]);
            if(x < child[j])
                child[j] <- x;    # hierarchy correction
          }
          S[,names(par.tod[i])] <- child;
        }
        # remove the "dummy root column"
        S <- S[,-which(colnames(S)==root)];
        return(S);
}
```

Script 4: Funzione che implementa la variante TPR-WT

```
tpr.weighted.threshold.free <- function(S,g, root="00",w=0.5){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           max.score <- max(S);
           z <- rep(max.score,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        # per-level bottom-up visit of the graph
        chd.bup <- get.children.bottom.up(g,levels)
        for(i in 1:length(chd.bup)){
          if(length(chd.bup[[i]])!=0){
            parent <- S[,names(chd.bup[i])];
            children <- as.matrix(S[,chd.bup[[i]]]);
            # colnames(children) <- chd.bup[[i]]
            for(j in 1:length(parent)){
              child.set <- children[j,] > parent[j];   # positive children choice
              child.pos <- children[j,][child.set];
              if(length(child.pos)!=0)
                # flat prediction correction
                parent[j] <- w*parent[j] + (1-w)*sum(child.pos)/length(child.pos);
            }
            S[,names(chd.bup[i])] <- parent;
          }
        }
        # per-level top-down visit of the graph
        par.tod <- get.parents.top.down(g,levels,root)
        for(i in 1:length(par.tod)){
          child <- S[,names(par.tod[i])];
          parents <- as.matrix(S[,par.tod[[i]]]);
          # Note: the version with an apply and an ifelse statement is slower
          for(j in 1:length(child)){
            x <- min(parents[j,]);
            if(x < child[j])
               child[j] <- x;    # hierarchy correction
          }
          S[,names(par.tod[i])] <- child;
        }
        # remove the "dummy root column"
        S <- S[,-which(colnames(S)==root)];
        return(S);
}
```

Script 5: Funzione che implementa la variante TPR-W

```
iso.tpr.threshold <- function(S,g, root="00",t=0.5){
        levels <- levels.graph(g,root)
        # a "dummy root column" is added if it is absent in the flat score matrix
        if(!(root %in% colnames(S))){
           max.score <- max(S);
           z <- rep(max.score,nrow(S));
           S <- cbind(z,S);
           colnames(S)[1] <- root;
        }
        S.cor <- S;
        # per-level bottom-up visit of the graph
        chd.bup <- get.children.bottom.up(g,levels);
        for(i in 1:length(chd.bup)){
          if(length(chd.bup[[i]])!=0){
            parent <- S.cor[,names(chd.bup[i])];
            children <- as.matrix(S.cor[,chd.bup[[i]]]);
            for(j in 1:length(parent)){
              child.set <- children[j,] > t;      #  positive children choice
              child.pos <- children[j,][child.set];
              parent[j] <- (parent[j] + sum(child.pos))/(1+length(child.pos));
            }
            S.cor[,names(chd.bup[i])] <- parent;
          }
        }
        # partial order isotonic regression
        S.cor.iso <- S.cor[,nodes(g)];
        eM <- matrixconstraints(g);     # matrix of constraints
        for(i in 1:nrow(S.cor.iso)){
          y <- S.cor.iso[i,];
          w <- rep(1,length(y));
          S.cor.iso[i,] <- activeSet(eM, "LS", y=y, weights=w, maxiter=1000)$x;
        }
        S.cor.iso <- S.cor.iso[,colnames(S)];
        S.cor.iso <- S.cor.iso[,-which(colnames(S.cor.iso)==root)];
        return(S.cor.iso);
}
```

Script 6: Funzione che implementa la variante ISO-TPR

C Funzioni di Utilità per Processare Grafi

```
levels.graph <- function(g,root="00") {
        ed <- edges(g);
        ew <- edgeWeights(g);
        for(i in 1:length(ed)){
              l <- length(ew[[i]]);
              if(l!=0)
                ew[[i]][1:l] <- -1;
        }
        edL <- vector(mode="list", length=length(ed));
        names(edL) <- names(ed);
        for(i in 1:length(ed)){
               edL[[i]] <- list(edges=ed[[i]], weights=ew[[i]]);
            }
        G <- graphNEL(nodes=nodes(g), edgeL=edL, edgemode="directed");
        depth.G <- bellman.ford.sp(G,root)$distance;
        depth.G <- -depth.G
        levels <- vector(mode="list", length=max(depth.G)+1);
        names(levels) <- paste(rep("level", max(depth.G)+1), 0:max(depth.G), sep="_");
            for(i in 1:(max(depth.G)+1)){
               levels[[i]] <- names(which(depth.G==i-1));
            }
        return(levels);
}
```

Script 7: Funzione per trovare per ogni nodo del grafo il suo cammino più lungo dalla root

```
get.parents.top.down <- function(g,levels, root="00") {
        ord.nd <- unlist(levels);
        ndL <- vector(mode="list", length=length(ord.nd));
        names(ndL) <- ord.nd;
        ed <- edges(g);
        for(i in 1:length(ed)){
          children <- ed[[i]];
          parent   <- names(ed[i]);
          if(length(children)!=0){
           for(j in 1:length(children))
             # store for each node its parent
             ndL[[children[j]]] <- c(ndL[[children[j]]],parent);
          }
        }
        ndL <- ndL[-which(names(ndL)==root)];  # remove root node
        return(ndL);
}
```

Script 8: Funzione per trovare i genitori di ogni nodo del grafo tramite una visita top-down per livelli

```
get.children.bottom.up <- function(g,levels){
        ed <- edges(g);
        nd <- c();
        n.levels <- length(levels);
        for(i in n.levels:1){
          level.nodes <- levels[[i]];
          nd <- append(nd,ed[level.nodes]);
        }
        return(nd);
}
```

Script 9: Funzione per trovare i figli di ogni nodo tramite una visita per livelli bottom-up del grafo

```
root.node <- function(g){
        og <- compute.flipped.graph(g);
        root <- find.leaves(og);
        return(root);
}
```

Script 10: Funzione per trovare il nodo root

```
compute.flipped.graph <- function(g){
        ed <- edges(g);
        ndL <- vector(mode="list", length=length(ed));
        names(ndL) <- names(ed);
        for(i in 1:length(ed)){
          children <- ed[[i]];
          parent   <- names(ed[i]);
          if(length(children)!=0){
            for(j in 1:length(children))
             ndL[[children[j]]] <- c(ndL[[children[j]]],parent);
          }
        }
        for (i in 1:length(ndL)) {
          ndL[[i]] <- list(edges=ndL[[i]]);
        }
        og <- graphNEL(nodes=nodes(g), edgeL=ndL, edgemode="directed");
        return(og);
}
```

Script 11: Funzione per determinare un grafo diretto con gli archi nella direzione opposta

```
find.leaves <- function(g){
        child <- edges(g);
        leaves <- c();
        for(i in 1:length(child)){
          if(length(child[[i]])==0)
            leaves <- append(leaves,names(child[i]));
        }
        return(leaves);
}
```

Script 12: Funzione per trovare i nodi foglia

```
constraints.matrix <- function(g){
        eM <- edgeMatrix(g);
        eM <- cbind(eM[2,],eM[1,]);
        nd <- nodes(g);
        dimnames(eM) <- list(nd[eM[,2]], c("child","parent"))
        return(eM);
}
```

Script 13: Funzione che computa la matrice delle costrizioni

Ringraziamenti

Desidero ringraziare il Professore *Giorgio Valentini* ed il Dr. *Matteo Re* per la disponibilità, la cortesia e per tutti i preziosi consigli forniti durante il tirocinio.

Un sentito ringraziamento ai miei genitori che con il loro sostegno morale ed economico, mi hanno permesso di raggiungere questo primo ed importante traguardo, nonché trampolino di lancio, nell'affascinante e stimolante mondo della ricerca.

Un forte ringraziamento ad Elisa per avermi sempre incoraggiato e sostenuto nei momenti difficili.

Un ultimo e sincero ringraziamento a Claudia e agli "amici del drone" (Andrea, Giovanni e Simone) per aver rallegrato le giornate trascorse nel laboratorio di Bioinformatica.

Bibliografia

[1] I. Friedberg, "Automated protein function prediction-the genomic challenge," *Brief. Bioinformatics*, vol. 7, pp. 225–242, 2006.

[2] G. Valentini, "Hierarchical Ensemble Methods for Protein Function Prediction," *ISRN Bioinformatics*, vol. 2014, no. Article ID 901419, p. 34 pages, 2014.

[3] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Guldener, G. Mannhaupt, M. Munsterkotter, and H. Mewes, "The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucleic Acids Research*, vol. 32, no. 18, pp. 5539–5545, 2004.

[4] The Gene Ontology Consortium, "Gene ontology: tool for the unification of biology," *Nature Genet.*, vol. 25, pp. 25–29, 2000.

[5] A. Valencia, "Automatic annotation of protein function," *Curr. Opin. Struct. Biol.*, vol. 15, pp. 267–274, 2005.

[6] N. Youngs, D. Penfold-Brown, K. Drew, D. Shasha, and R. Bonneau, "Parametric bayesian priors and better choice of negative examples improve protein function prediction," *Bioinformatics*, vol. 29, no. 9, pp. 1190–1198, 2013.

[7] A. Juncker, L. Jensen, A. Perleoni, A. Bernsel, M. Tress, P. Bork, G. von Heijne, A. Valencia, A. Ouzounis, R. Casadio, and S. Brunak, "Sequence-based feature prediction and annotation of proteins," *Genome Biology*, vol. 10:206, 2009.

[8] R. Sharan, I. Ulitsky, and R. Shamir, "Network-based prediction of protein function," *Mol. Sys. Biol.*, vol. 8, no. 88, 2007.

[9] A. Sokolov and A. Ben-Hur, "Hierarchical classification of Gene Ontology terms using the GOstruct method," *Journal of Bioinformatics and Computational Biology*, vol. 8, no. 2, pp. 357–376, 2010.

[10] Z. Barutcuoglu, R. Schapire, and O. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[11] H. Chua, W. Sung, and L. Wong, "An efficient strategy for extensive integration of diverse biological data for protein function prediction," *Bioinformatics*, vol. 23, no. 24, pp. 3364–3373, 2007.

[12] G. Lanckriet, T. De Bie, N. Cristianini, M. Jordan, and W. Noble, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, pp. 2626–2635, 2004.

[13] P. Pavlidis, J. Weston, J. Cai, and W. Noble, "Learning gene functional classification from multiple data," *J. Comput. Biol.*, vol. 9, pp. 401–411, 2002.

[14] M. Re and G. Valentini, "Simple ensemble methods are competitive with state-of-the-art data integration methods for gene function prediction," *Journal of Machine Learning Research, W&C Proceedings, Machine Learning in Systems Biology*, vol. 8, pp. 98–111, 2010.

[15] A. Clare and R. King, "Predicting gene function in *saccharomices cerevisiae*," *Bioinformatics*, vol. 19, no. Supp.2, pp. II42–II49, 2003.

[16] Y. Bilu and M. Linial, "The advantage of functional prediction based on clustering of yeast genes and its correlation with non-sequence based classification," *Journal of Computational Biology*, vol. 9, pp. 193–210, 2002.

[17] L. Pena-Castillo *et al.*, "A critical assessment of Mus musculus gene function prediction using integrated genomic evidence," *Genome Biology*, vol. 9, p. S1, 2008.

[18] W. Tian, L. Zhang, M. Tasan, F. Gibbons, O. King, J. Park, Z. Wunderlich, J. Cherry, and F. Roth, "Combining guilt-by-association and guilt-by-profiling to predict Saccharomices cerevisiae gene function," *Genome Biology*, vol. 9, p. S7, 2008.

[19] W. Kim, C. Krumpelman, and E. Marcotte, "Inferring mouse gene functions from genomic-scale data using a combined functional network/classification strategy," *Genome Biology*, vol. 9, p. S5, 2008.

[20] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris, "GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function," *Genome Biology*, vol. 9, no. S4, 2008.

[21] I. Lee, B. Ambaru, P. Pranjali-Thakkar, E. Marcotte, and S. Rhee, "Rational association of genes with traits using a genome-scale gene network for arabidopsis thaliana," *Nat. Biotechnol.*, vol. 28, pp. 149–156, 2010.

[22] P. Radivojac *et al.*, "A large-scale evaluation of computational protein function prediction," *Nature Methods*, vol. 10, no. 3, pp. 221–227, 2013.

[23] O. Troyanskaya *et al.*, "A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *saccharomices cerevisiae*)," *Proc. Natl Acad. Sci. USA*, vol. 100, pp. 8348–8353, 2003.

[24] K. Tsuda, H. Shin, and B. Scholkopf, "Fast protein classification with multiple networks," *Bioinformatics*, vol. 21, no. Suppl 2, pp. ii59–ii65, 2005.

[25] J. Xiong *et al.*, "Genome wide prediction of gene function via a generic knowledge discovery approach based on evidence integration," *BMC Bioinformatics*, vol. 7, no. 268, 2006.

[26] U. Karaoz *et al.*, "Whole-genome annotation by using evidence integration in functional-linkage networks," *Proc. Natl Acad. Sci. USA*, vol. 101, pp. 2888–2893, 2004.

[27] B. Done, P. Khatri, A. Done, and S. Draghici, "Predicting novel human gene ontology annotations using semantic analysis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 1, pp. 91–99, 2010.

[28] G. Valentini and F. Masulli, "Ensembles of learning machines," in *Neural Nets WIRN-02*, vol. 2486 of *Lecture Notes in Computer Science*, pp. 3–19, Springer, 2002.

[29] B. Shahbaba and M. Neal, "Gene function classification using Bayesian models with hierarchy-based priors," *BMC Bioinformatics*, vol. 7, no. 448, 2006.

[30] C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, pp. 185–214, 2008.

[31] G. Obozinski, G. Lanckriet, C. Grant, J. M., and W. Noble, "Consistent probabilistic output for protein function prediction," *Genome Biology*, vol. 9, no. S6, 2008.

[32] G. Valentini, "True path rule hierarchical ensembles," in *Multiple Classifier Systems. Eighth International Workshop, MCS 2009, Reykjavik, Iceland* (J. Kittler, J. Benediktsson, and F. Roli, eds.), vol. 5519 of *Lecture Notes in Computer Science*, pp. 232–241, Springer, 2009.

[33] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool.," *Journal of Molecular Biology*, vol. 215, 1990.

[34] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[35] A. Conesa, S. Gotz, J. Garcia-Gomez, J. Terol, T. M., and M. Robles, "Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research," *Bioinformatics*, vol. 21, no. 18, pp. 3674–3676, 2005.

[36] A. Prlic, T. Down, E. Kulesha, R. Finn, A. Kahari, and T. Hubbard, "Integrating sequence and structural biology with DAS," *BMC Bioinformatics*, vol. 8, no. 233, 2007.

[37] M. Falda, S. Toppo, A. Pescarolo, E. Lavezzo, D. B., A. Facchinetti, E. Cilia, R. Velasco, and P. Fontana, "Argot2: a large scale function prediction tool relying on semantic similarity of weighted Gene Ontology terms," *BMC Bioinformatics*, vol. 13, no. Suppl 4:S14, 2012.

[38] T. Hamp *et al.*, "Homology-based inference sets the bar high for protein function prediction," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S7, 2013.

[39] E. Marcotte, M. Pellegrini, M. Thompson, T. Yeates, and D. Eisenberg, "A combined algorithm for genome-wide prediction of protein function," *Nature*, vol. 402, pp. 83–86, 1999.

[40] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani, "Global protein function prediction from protein-protein interaction networks," *Nature Biotechnology*, vol. 21, pp. 697–700, 2003.

[41] A. Bertoni, M. Frasca, and G. Valentini, "Cosnet: a cost sensitive neural network for semi-supervised learning in graphs," in *European Conference on Machine Learning, ECML PKDD 2011*, vol. 6911 of *Lecture Notes on Artificial Intelligence*, pp. 219–234, Springer, 2011.

[42] M. Frasca, A. Bertoni, M. Re, and G. Valentini, "A neural network algorithm for semi-supervised node label learning from unbalanced data," *Neural Networks*, vol. 43, pp. 84–98, 2013.

[43] M. Deng, T. Chen, and F. Sun, "An integrated probabilistic model for functional prediction of proteins," *J. Comput. Biol.*, vol. 11, pp. 463–475, 2004.

[44] Y. Kourmpetis, A. Van Dijk, M. Bink, R. Van Ham, and C. Ter Braak, "Markov random field analysis for protein function prediction based on network data," *PLoS ONE*, vol. 5, no. 2:e9293, 2010.

[45] S. Oliver, "Guilt-by-association goes global," *Nature*, vol. 403, pp. 601–603, 2000.

[46] R. McDermott, J.and Bumgarner and R. Samudrala, "Functional annotation from predicted protein interaction networks," *Bioinformatics*, vol. 21, no. 15, pp. 3217–3226, 2005.

[47] M. Re and G. Valentini, "Cancer module genes ranking using kernelized score functions," *BMC Bioinformatics*, vol. 13, no. Suppl 14/S3, 2012.

[48] M. Re and G. Valentini, "Large scale ranking and repositioning of drugs with respect to drugbank therapeutic categories," in *International Symposium on Bioinformatics Research and Applications (ISBRA 2012)*, vol. 7292 of *Lecture Notes in Computer Science*, pp. 225–236, Springer, 2012.

[49] I. Cattinelli, G. Valentini, E. Paulesu, and A. Borghese, "A Novel Approach to the Problem of Non-uniqueness of the Solution in Hierarchical Clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 7, pp. 1166–1173, 2013.

[50] G. Bakir, T. Hoffman, B. Scholkopf, B. Smola, A.J. andTaskar, and S. Vishwanathan, *Predicting structured data*. Cambridge, MA: MIT Press, 2007.

[51] I. Tsochantaridis, T. Joachims, T. Hoffman, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[52] K. Astikainen, L. Holm, E. Pitkanen, S. Szedmak, and J. Rousu, "Towards structured output prediction of enzyme function," *BMC Proceedings*, vol. 2, no. Suppl 4:S2, 2008.

[53] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[54] T. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy* (J. Kittler and F. Roli, eds.), vol. 1857 of *Lecture Notes in Computer Science*, pp. 1–15, Springer-Verlag, 2000.

[55] O. Okun, G. Valentini, and M. Re, *Ensembles in Machine Learning Applications*, vol. 373 of *Studies in Computational Intelligence*. Berlin: Springer, 2011.

[56] M. Re and G. Valentini, "Ensemble methods: a review," in *Advances in Machine Learning and Data Mining for Astronomy*, Data Mining and Knowledge Discovery, pp. 563–594, Chapman & Hall, 2012.

[57] X. Jiang, N. Nariai, M. Steffen, S. Kasif, and E. Kolaczyk, "Integration of relational and hierarchical network information for protein function prediction," *BMC Bioinformatics*, vol. 9, no. 350, 2008.

[58] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, and S. Dzeroski, "Predicting gene function using hierarchical multi-label decision tree ensembles," *BMC Bioinformatics*, vol. 11, no. 2, 2010.

[59] R. Eisner, B. Poulin, D. Szafron, and P. Lu, "Improving protein prediction using the hierarchical structure of the Gene Oontology," in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2005.

[60] H. Blockeel, L. Schietgat, and A. Clare, "Hierarchical multilabel classification trees for gene function prediction," in *Probabilistic Modeling and Machine Learning in Structural and Systems Biology* (J. Rousu, S. Kaski, and E. Ukkonen, eds.), (Tuusula, Finland), Helsinki University Printing House, 2006.

[61] R. Cerri and A. de Carvalho, "Hierarchical multilabel classification using top-down label combination and artificial neural networks," in *SBRN '10 Proceedings of the 2010 Eleventh Brazilian Symposium on Neural Networks*, pp. 253–258, IEEE Computer Society, 2010.

[62] R. Cerri and A. de Carvalho, "Hierarchical multilabel protein function prediction using local neural networks," in *Advances in Bioinformatics and Computational Biology*, no. 6832 in Lecture Notes in Computer Science, pp. 10–17, Springer-Verlag, 2011.

[63] J. Hernandez, L. Sucar, and E. Morales, "A hybrid global-local approach forhierarchcial classification," in *Proc. of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*, pp. 432–437, 2013.

[64] B. Paes, A. Plastion, and A. Freitas, "Improving loacal per level hierarchical classification," *Journal of Information and Data Management*, vol. 3, no. 3, pp. 394–409, 2012.

[65] G. Valentini and M. Re, "Weighted True Path Rule: a multilabel hierarchical algorithm for gene function prediction," in *MLD-ECML 2009, 1st International Workshop on learning from Multi-Label Data*, (Bled, Slovenia), pp. 133–146, 2009.

[66] G. Valentini, "True Path Rule hierarchical ensembles for genome-wide gene function prediction," *IEEE ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 3, pp. 832–847, 2011.

[67] B. Chen and J. Hu, "Hierarchical multi-label classification based on over-sampling and hierarchy constraint for gene function prediction," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 7, pp. 183–189, 2012.

[68] R. Cerri and A. de Carvalho, "New top-down methods using SVMs for hierarchical multilabel classification problems," in *IJCNN 2010*, pp. 1–8, IEEE Computer Society, 2010.

[69] D. Rumelhart, R. Durbin, and Y. Chauvin, "Backpropagation: the basic theory," in *Backpropagation: Theory, Architectures and Applications* (R. D. Chauvin Y., ed.), pp. 1–34, Hillsdale, NJ: Lawrence Erlbaum, 1995.

[70] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-Labelsets for Multi-Label Classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[71] Gene Ontology Consortium, "True path rule," 2010. http://www.geneontology.org/GO.usage.shtml#truePathRule.

[72] A. Smola and I. Kondor, "Kernel and regularization on graphs," in *Proc. of the Annual Conf. on Computational Learning Theory* (B. Scholkopf and M. Warmuth, eds.), Lecture Notes in Computer Science, pp. 144–158, Springer, 2003.

[73] C. Leslie *et al.*, "Mismatch string kernels for svm protein classification," in *Advances in Neural Information Processing Systems*, (Cambridge, MA), pp. 1441–1448, MIT Press, 2003.

[74] R. Barlow and H. Brunk, "The isotonic regression problem and its dual," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 140–147, 1972.

[75] O. Burdakov, O. Sysoev, A. Grimvall, and M. Hussian, "An $\mathcal{O}(n^2)$ algorithm for isotonic regression," in *Large-Scale Nonlinear Optimization*, no. 83 in Nonconvex Optimization and Its Applications, pp. 25–33, Springer-Verlag, 2006.

[76] J. Quinlan, "Induction of decision trees," *Machine Learning*, no. 1, pp. 81–106, 1986.

[77] R. King, A. Karwath, A. Clare, and L. Dehaspe, "The utility of different representations of protein sequence for predicting functional class," *Bioinformatics*, vol. 17, no. 5, pp. 445–454, 2001.

[78] F. Otero, A. Freitas, and C. Johnson, "A hierarchical classification ant colony algorithm for predicting gene ontology terms," in *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, no. 5483 in Lecture Notes in Computer Science, pp. 68–79, Springer-Verlag, 2009.

[79] N. Cesa-Bianchi, M. Re, and G. Valentini, "Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference," *Machine Learning*, vol. 88, no. 1, pp. 209–241, 2012.

[80] W. Noble and A. Ben-Hur, "Integrating information for protein function prediction," in *Bioinformatics - From Genomes to Therapies* (T. Lengauer, ed.), vol. 3, pp. 1297–1314, Wiley-VCH, 2007.

[81] D. Cozzetto, D. Buchan, K. Bryson, and D. Jones, "Protein function prediction by massive integration of evolutionary analyses and multiple data sources," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S1, 2013.

[82] L. Lan, N. Djuric, Y. Guo, and V. S., "MS-kNN: protein function prediction by integrating multiple data sources," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S8, 2013.

[83] A. Sokolov, C. Funk, K. Graim, K. Verspoor, and A. Ben-Hur, "Combining heterogeneous data sources for accurate functional annotation of proteins," *BMC Bioinformatics*, vol. 14, no. Suppl 3:S10, 2013.

[84] M. desJardins, P. Karp, M. Krummenacker, T. Lee, and C. Ouzounis, "Prediction of enzyme classification from protein sequence without the use of sequence similarity," in *Proc. of the 5th ISMB*, pp. 92–99, AAAI Press, 1997.

[85] Y. Guan, C. Myers, D. Hess, Z. Barutcuoglu, A. Caudy, and O. Troyanskaya, "Predicting gene function in a hierarchical context with an ensemble of classifiers," *Genome Biology*, vol. 9, no. S2, 2008.

[86] C. Myers and O. Troyanskaya, "Context-sensitive data inegration and prediction of biological networks," *Bioinformatics*, vol. 23, pp. 2322–2330, 2007.

[87] S. Mostafavi and Q. Morris, "Fast integration of heterogeneous data sources for predicting gene function with limited annotation," *Bioinformatics*, vol. 26, no. 14, pp. 1759–1765, 2010.

[88] M. Mesiti, E. Jimenez-Ruiz, I. Sanz, R. Berlanga-Llavori, P. Perlasca, G. Valentini, and D. Manset, "XML-Based Approaches for the Integration of Heterogeneous Bio-Molecular Data," *BMC Bioinformatics*, vol. S12, p. S7, 2009.

[89] G. Lanckriet, R. G. Gert, M. Deng, N. Cristianini, M. Jordan, and W. Noble, "Kernel-based data fusion and its application to protein function prediction in yeast," in *Proceedings of the Pacific Symposium on Biocomputing*, pp. 300–311, 2004.

[90] D. Lewis, T. Jebara, and W. Noble, "Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure," *Bioinformatics*, vol. 22, no. 22, pp. 2753–2760, 2006.

[91] N. Cesa-Bianchi, M. Re, and G. Valentini, "Functional inference in FunCat through the combination of hierarchical ensembles with data fusion methods," in *ICML-MLD 2nd International Workshop on learning from Multi-Label Data*, (Haifa, Israel), pp. 13–20, 2010.

[92] G. Yu, H. Rangwala, C. Domeniconi, G. Zhang, and Z. Zhang, "Protein function prediction by integrating multiple kernels," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, (Beijing, China), 2013.

[93] D. Titterington, G. Murray, D. Spiegelhalter, A. Skene, J. Habbema, and G. Gelpke, "Comparison of discriminant techniques applied to a complex data set of head injured patients," *Journal of the Royal Statistical Society*, vol. 144, no. 2, 1981.

[94] N. de Condorcet, *Essai sur l' application de l' analyse à la probabilité des decisions rendues à la pluralité des voix*. Paris: Imprimerie Royale, 1785.

[95] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[96] L. Kuncheva, J. Bezdek, and R. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 34, no. 2, pp. 299–314, 2001.

[97] M. Re and G. Valentini, "Noise tolerance of Multiple Classifier Systems in data integration-based gene function prediction," *Journal of Integrative Bioinformatics*, vol. 7, no. 3:139, 2010.

[98] G. Valentini, "Notes on hierarchical ensemble methods for DAG-structured taxonomies," *CoRR*, vol. abs/1406.4472, 2014.

[99] N. Cesa-Bianchi and G. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," *Journal of Machine Learning Research, W&C Proceedings, Machine Learning in Systems Biology*, vol. 8, pp. 14–29, 2010.

[100] A. Burred, J.and Lerch, "A hierarchical approach to automatic musical genre classification," in *Proc. of the 6th Int. Conf. on Digital Audio Effects*, pp. 8–11, 2003.

[101] C. DeCoro, Z. Barutcuoglu, and R. Fiebrink, "Bayesian aggregation for hierarchical genre classification," in *Proc. of the 8th Int. Conf. on Music Information Retrieval*, pp. 77–80, 2007.

[102] K. Trohidis, G. Tsoumahas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. of the 9th International Conference on Music Information Retrieval*, pp. 325–330, 2008.

[103] A. Binder, M. Kawanabe, and U. Brefeld, "Bayesian aggregation for hierarchical genre classification," in *Proc. of the 9th Asian Conf. on Computer Vision*, 2009.

[104] Z. Barutcuoglu and C. DeCoro, "Hierarchical shape classification using bayesian aggregation," in *Proc. of the IEEE Conf. on Shape Modeling and Applications*, 2006.

[105] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "An empirical study of multi-label methods for video annotation," in *Proc. 7th International Workshop on Content-Based Multimedia Indexing, CBMI 09*, (Chania, Greece), 2009.

[106] K. Punera and J. Ghosh, "Enhanced hierarchical classification via isotonic smoothing," in *WWW 2008*, (Beijing, China), pp. 151–160, ACM, 2008.

[107] M. Ceci and D. Malerba, "Classifying web documents in a hierarchy of categories: A comprehensive study," *Journal of Intelligent Information Systems*, vol. 28, no. 1, pp. 1–41, 2007.

[108] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: Combining Bayes with SVM," in *Proc. of the 23rd Int. Conf. on Machine Learning*, pp. 177–184, ACM Press, 2006.

[109] G. Valentini, S. Kohler, M. Re, M. Notaro, and N. Peter Robison, "Prediction of human gene-phenotype association by exploiting the hierarchical structure of the Human Phenotype Ontology," *IWBBIO*, 2015.

[110] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Boston: MIT Press, 2009.

[111] N. Peter Robison, M. Frasca, S. Kohler, M. Notaro, M. Re, and G. Valentini, "Multi-label prediction in DAG-structured taxonomies using True Path Rule ensembles," 2015. (in press).

[112] M. Re, M. Mesiti, and G. Valentini, "A Fast Ranking Algorithm for Predicting Gene Functions in Biomolecular Networks," *IEEE ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 6, pp. 1812–1818, 2012.

[113] M. Re and G. Valentini, "Network-based Drug Ranking and Repositioning with respect to DrugBank Therapeutic Categories," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 6, pp. 1359–1371, 2013.

[114] X. Zhu *et al.*, "Semi-supervised learning with gaussian fields and harmonic functions," in *Proc. of the 20th Int. Conf. on Machine Learning*, (Washintgton DC, USA), 2003.

[115] P. Robinson, P. Krawitz, and S. Mundlos, "Strategies for exome and genome sequence data analysis in disease-gene discovery projects," *Cin. Genet.*, vol. 80, pp. 127 – 132, 2011.

[116] J. Amberger, C. Bocchini, and A. Amosh, "A new face and new challenges for Online Mendelian inheritance in Man (OMIM)," *Hum. Mutat.*, vol. 32, pp. 564–7, 2011.

[117] S. Kohler *et al.*, "The human phenotype ontology project: linking molecular biology and disease through phenotype data.," *Nucleic Acids Research*, vol. 42, no. (Database issue), pp. D966–74, 2014.

[118] G. Valentini, A. Paccanaro, H. Caniza, A. Romero, and M. Re, "An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods," *Artificial Intelligence in Medicine*, vol. 61, no. 2, pp. 63–78, 2014.

[119] G. Wu, X. Feng, and L. Stein, "A human functional protein interaction network and its application to cancer data analysis," *Genome Biol*, vol. 11, p. R53, 2010.

[120] I. Lee, U. Blom, P. I. Wang, J. Shim, and E. Marcotte, "Prioritizing candidate disease genes by network-based boosting of genome-wide association data," *Genome Research*, vol. 21, no. 7, pp. 1109–1121, 2011.