



Università degli Studi di Ferrara

UNIVERSITÀ DEGLI STUDI DI FERRARA
CORSO DI LAUREA IN INFORMATICA

*Un framework sistematico logic-based
per spiegare modelli simbolici di
ensemble*

Relatore:

Prof. Guido SCIAVICCO

Laureando:

Marco PERROTTA

Correlatori:

Dott. Eduard Ionel STAN

Dott. Giovanni PAGLIARINI

ANNO ACCADEMICO 2024 – 2025

Indice

	Page
1 Introduzione	5
2 Background	9
2.1 Machine learning	10
2.1.1 La dicotomia tra approccio funzionale e simbolico	10
2.1.2 L'Intelligenza artificiale simbolica	10
2.1.3 Sistemi esperti e rappresentazioni simboliche	11
2.1.4 Vantaggi dell'approccio simbolico	11
2.1.5 Metriche	11
2.2 Logica proposizionale	12
2.2.1 Sintassi	13
2.2.2 Semantica	14
2.3 Minimizzazione di formule proposizionali	15
2.3.1 Forme normali: CNF e DNF	15
2.3.2 Minimizzazione DNF	17
2.4 Modelli simbolici	18
2.4.1 Dataset	19
2.4.2 Alberi e foreste decisionali	20

3	Lavori correlati	23
3.1	Spiegazioni locali e globali	23
3.1.1	Approcci alla IA interpretabile	25
3.2	Apporto di LUMEN	26
4	Framework logico per l'interpretazione di ensemble	29
4.1	Estrazione sistematica di regole minime da foreste decisionali	29
4.1.1	Formule di cammino e di classe negli alberi	30
4.1.2	Formule di cammino e di classe nelle foreste	31
4.1.3	Regole derivate da una foresta decisionale	32
4.1.4	Problemi di minimizzazione per regole forti e deboli	33
4.1.5	Problema della regola forte minima	33
4.1.6	Regole deboli	34
4.1.7	Problemi di complessità per le regole deboli	35
4.2	Un approccio di estrazione e minimizzazione	35
4.3	Estrazione e minimizzazione logica con LUMEN	36
4.3.1	Dal modello a tabelle di verità	37
4.3.2	Strategie di riduzione della complessità	38
4.3.3	Minimizzazione logica	41
4.3.4	L'algoritmo LUMEN	42
5	Sperimentazione e Analisi	45
5.1	Implementazione	46
5.1.1	Il pacchetto SolePostHoc	46
5.1.2	Riproducibilità e accessibilità del codice	48
5.1.3	Prospettive di sviluppo e distribuzione	48
5.2	Prima fase	50
5.3	Seconda fase	51
5.4	Terza fase	53
5.5	Considerazioni conclusive	53
6	Conclusioni e sviluppi futuri	55

Introduzione

L'avvento dell'Intelligenza Artificiale e del Machine Learning ha portato allo sviluppo di modelli predittivi di elevata complessità ed efficacia, come gli ensemble basati su modelli ad albero (e.g., Random Forests [7], GBDTs [19, 10, 26]). Questi modelli sono apprezzati per le loro forti prestazioni, la robustezza contro l'overfitting e la capacità di gestire dati ad alta dimensionalità [18]. Tuttavia, sebbene un singolo albero decisionale sia spesso considerato interpretabile, l'ensembling di numerosi alberi risulta in una "scatola nera", ponendo seri limiti in contesti critici (sanità, finanza, diritto) dove è fondamentale garantire trasparenza, giustificabilità e interpretabilità.

A tal proposito, il campo della eXplainable Artificial Intelligence (XAI) ha prodotto numerosi metodi per chiarire il comportamento dei modelli complessi [21, 31]. Tecniche di spiegazione locali, quali LIME [38], SHAP [30], Bellatrix [16] e metodi per ottenere spiegazioni logiche univoche [24], hanno rivoluzionato la comprensione delle decisioni a livello di singola istanza. Tuttavia, l'interpretabilità globale, volta a distillare l'intera logica decisionale di un ensemble, si confronta con sfide legate alla fedeltà della rappresentazione e all'accesso limitato ai dati di training, spingendo la ricerca verso soluzioni basate su potature euristiche, clustering o semplificazioni (e.s., [17, 45]).

In linea di principio, un ensemble – che sia costituito da tecniche di bagging come le Random Forests o da metodi di boosting come XGBoost [10] – codifica un complesso confine decisionale che, in teoria, può essere rappresentato tramite una formula logica proposizionale. L'estrazione di tale formula e la sua semplificazione in una forma gestibile, mantenendo intatta la fedeltà rispetto al modello originale, rappresenta una sfida fondamentale. In questo studio viene affrontato il problema ricorrendo ad algoritmi di minimizzazione logica noti e maturi, come ESPRESSO [5], per trasformare la struttura di una foresta decisionale in regole logiche espresse in Forma Normale Disgiuntiva (DNF).

Nel panorama dell'interpretabilità post-hoc, diversi approcci sono stati proposti:

- **BATrees (Born-Again Trees)** [6]: converte interi ensemble in un singolo albero decisionale semplificato, con il rischio di sacrificare parte della precisione.
- **InTrees (IT)** [17]: estrae, misura, pota e seleziona regole dagli ensemble, sintetizzandole in un modello semplificato, pur richiedendo l'accesso al dataset originale.
- **Surrogate Trees** (ad es., TREPAN [12, 14] e REFNE [46]): utilizzano il modello black box come oracolo per generare nuovi dataset sintetici, consentendo di addestrare modelli simbolici che approssimano il comportamento originale, sebbene con limitazioni in termini di fedeltà.
- **RuleCOSI+** [32]: costruisce una decision list comprensibile che tenta di riprodurre il comportamento del modello originale, potendo però presentare discrepanze.

Il contributo che si propone nella presente tesi è lo sviluppo di un ulteriore approccio di nome **LUMEN (Logic-driven Unified Minimal Extractor of Notions)**, un framework innovativo per l'estrazione di regole logiche globali da ensemble di modelli. LUMEN integra:

1. Algoritmi di minimizzazione noti come ESPRESSO [5] per la minimizzazione logica, trasformando il confine decisionale complesso in regole in DNF, garantendo fedeltà perfetta.

2. Un meccanismo di approssimazione controllata, che bilancia il compromesso tra accuratezza, interpretabilità e costo computazionale, supportato da garanzie teoriche sulla degradazione della fedeltà.
3. La possibilità di estensioni del metodo agli ensemble basati su boosting e, tramite tecniche di estrazione di concetti, anche a modelli sub-simbolici come le reti neurali. In quest'ottica, approcci per l'interpretabilità basati su concetti (es. TCAV [28], Network Dissection [2], Concept Bottleneck Models [29]) offrono la possibilità di convertire le feature in proposizioni semanticamente coerenti, favorendo un collegamento tra paradigmi simbolici e sub-simbolici.

Il presente lavoro si inserisce in un contesto di crescente attenzione verso la trasparenza e l'intelligibilità dei modelli di machine learning, in cui la capacità di estrarre e rappresentare in forma simbolica le regole decisionali può supportare sia l'analisi post-hoc che l'adozione in domini regolamentati. La presente soluzione si concretizza in una libreria open-source end-to-end, che standardizza le interfacce per l'interpretazione globale dei modelli, consentendo agli utenti di confrontare diverse tecniche mediante un'unica interfaccia e di condurre test comparativi personalizzati.

Di seguito si descrive la struttura della tesi. Nel Capitolo 2 verranno esaminati i fondamenti teorici necessari per una maggiore comprensione del problema affrontato, definendo i concetti chiave nel campo del Machine Learning e della logica. Nel Capitolo 3 si analizzeranno i lavori esistenti sull'estrazione di regole e sulle spiegazioni globali, evidenziando potenzialità e limitazioni dei metodi attuali. Nel Capitolo 4 verrà esplorata la complessità del problema dell'estrazione di regole minime e si illustrerà in dettaglio il framework LUMEN, includendo il processo di minimizzazione logica, i parametri di campionamento e le relative garanzie teoriche. Il Capitolo 5 presenterà i risultati sperimentali, con un confronto sistematico rispetto ai baseline esistenti. Il Capitolo 6 concluderà la tesi riassumendo i contributi e delineando le possibili direzioni future di ricerca.

Background

In questo capitolo si affronterà una trattazione che si apre introducendo concetti già consolidati in letteratura, essenziali per la piena comprensione del problema affrontato nei capitoli successivi, e dei risultati ottenuti. Sebbene alcuni di questi possano essere ampiamente riconosciuti dalla comunità scientifica, la loro rilevanza rimane fondamentale nel contesto di questo lavoro.

Il capitolo si articola in diverse sezioni. Inizialmente, viene esaminato l'apprendimento simbolico, sottocampo del machine learning, esplorandone le caratteristiche fondamentali che costituiscono il punto di partenza di questa tesi. Successivamente, si presenta la logica proposizionale, base teorica imprescindibile per comprendere i meccanismi di funzionamento di LUMEN, che verranno descritti nei capitoli successivi.

Si prosegue infine con la presentazione delle strutture principali utilizzate nell'ambito del machine learning interpretabile, come i decision tree e le random forest.

2.1 Machine learning

Il campo del machine learning presenta al suo interno una fondamentale dicotomia [15]: l'apprendimento funzionale e l'apprendimento simbolico. Mentre l'apprendimento funzionale mira a identificare una funzione matematica che modelli un fenomeno, l'apprendimento simbolico cerca di estrarre una rappresentazione logica e interpretabile dello stesso.

2.1.1 La dicotomia tra approccio funzionale e simbolico

L'apprendimento funzionale si concentra sulla ricerca di funzioni che mappano input a output, modellando i fenomeni attraverso rappresentazioni matematiche. Queste funzioni possono spaziare da modelli semplici e lineari fino a strutture complesse come le reti neurali profonde. L'obiettivo principale è l'ottimizzazione della performance predittiva, spesso a discapito dell'interpretabilità.

L'apprendimento simbolico, invece, si focalizza sulla costruzione di rappresentazioni logiche esplicite e comprensibili. Queste rappresentazioni si basano su simboli e regole che possono essere direttamente interpretati dagli esseri umani, rendendo trasparente il processo decisionale del modello.

2.1.2 L'Intelligenza artificiale simbolica

L'intelligenza artificiale simbolica comprende i metodi basati su rappresentazioni simboliche leggibili dall'uomo, logica formale e ricerca. Questo paradigma ha dominato la ricerca sull'IA dalla metà degli anni '50 fino alla fine degli anni '80, un periodo che John Haugeland ha definito "GOFAI" (Good Old-Fashioned Artificial Intelligence) nel suo influente libro del 1985 "Artificial Intelligence: The Very Idea", che esplorava le implicazioni filosofiche della ricerca sull'IA. Nel campo della robotica, il termine analogo è "GOFR" (Good Old-Fashioned Robotics).

L'approccio simbolico si fonda su quello che Allen Newell e Herbert A. Simon definirono a metà degli anni '60 come "ipotesi dei sistemi di simboli fisici": l'assunto che molti aspetti dell'intelligenza possano essere raggiunti mediante la manipolazione di simboli. Questa intuizione ha portato allo sviluppo di sistemi in grado di rappresentare la conoscenza attraverso un linguaggio formale e manipolare tale rappresentazione mediante regole logiche ben definite.

2.1.3 Sistemi esperti e rappresentazioni simboliche

Una delle applicazioni più rilevanti dell'IA simbolica sono i sistemi esperti, che codificano la conoscenza di dominio attraverso una rete di regole di produzione. Queste regole collegano i simboli in relazioni di tipo condizionale, simili a istruzioni “se-allora” (If-Then). I sistemi esperti elaborano tali regole per effettuare deduzioni logiche e determinare quali informazioni aggiuntive sono necessarie, formulando domande pertinenti e utilizzando rappresentazioni comprensibili per gli esseri umani.

2.1.4 Vantaggi dell'approccio simbolico

L'approccio simbolico offre numerosi vantaggi rispetto ai metodi puramente funzionali:

- **Interpretabilità:** Le rappresentazioni simboliche sono intrinsecamente comprensibili per gli esseri umani, rendendo trasparente il processo decisionale.
- **Spiegabilità:** I modelli simbolici possono fornire spiegazioni chiare e logiche per le loro previsioni, indicando quali regole sono state attivate.
- **Verificabilità:** È possibile verificare formalmente la correttezza di un sistema simbolico attraverso metodi logici e matematici.
- **Incorporazione di conoscenza a priori:** L'approccio simbolico permette di integrare facilmente conoscenze di dominio esistenti nel processo di apprendimento.

2.1.5 Metriche

In questa sezione verranno descritte le metriche utilizzate durante la trattazione in fase sperimentale per valutare e analizzare le prestazioni dei vari approcci di estrazione delle regole. Tutte queste metriche sono state selezionate in linea con i lavori precedenti nel campo.

- **Confidenza:** Viene calcolata come la proporzione di predizioni corrette sul totale delle predizioni. Formula: $\text{Confidenza} = \frac{\text{Predizioni Corrette}}{\text{Numero Totale di Predizioni}} \times 100\%$
Maggiore è la percentuale più il modello risulta accurato.

- **Specificità:** Viene rappresentata come la proporzione di previsioni negative correttamente identificate rispetto al totale delle previsioni negative. Formula: $\text{Specificità} = \frac{TN}{TN+FP}$
Dove TN rappresenta i veri negativi e FP i falsi positivi.
- **Sensibilità:** Viene misurata come la proporzione di previsioni positive correttamente identificate rispetto al totale delle previsioni positive. Formula: $\text{Sensibilità} = \frac{TP}{TP+FN}$
Dove TP rappresenta i veri positivi e FN i falsi negativi.
- **Lunghezza Media della Spiegazione Locale Minima:** Viene valutata come la dimensione della singola congiunzione che attiva l'istanza di test.
- **Lunghezza Totale della Regola:** Viene misurata in Forma Normale Disgiuntiva (DNF) per garantire la comparabilità delle spiegazioni locali tra diversi metodi.

2.2 Logica proposizionale

In questa trattazione si farà riferimento alle definizioni e alla nomenclatura adottate nel manuale [1] e nelle dispense [11] di Logica Informatica. Questi testi sono stati scelti perché offrono un equilibrio tra rigore teorico e chiarezza espositiva, fornendo una base solida e strutturata per la comprensione degli argomenti trattati. La logica proposizionale, indicata con \mathcal{PL} , si propone di formalizzare e analizzare i ragionamenti espressi nel linguaggio naturale, traducendoli in formule composte mediante connettivi quali: “e”, “o”, “sia...sia”, “né...né”, “ma non”, “o...o”, “e/o”, “se...allora”. Il linguaggio della logica proposizionale, ossia l'insieme dei simboli convenzionali impiegati nell'analisi matematica della logica, si articola nei seguenti gruppi:

- **Valori di verità:** \perp (che rappresenta “falso”) e \top (che rappresenta “vero”).
- **Variabili proposizionali o simboli atomici:** p, q, r, s, t, \dots
- **Connettivi logici:** \wedge (congiunzione), \vee (disgiunzione), \rightarrow (implicazione), \neg (negazione).

- **Simboli ausiliari:** le parentesi, necessarie per rimuovere ambiguità nelle strutture delle formule, quando si rappresentano come stringhe lineari di simboli.

Le costanti e le variabili rappresentano proposizioni “atomiche”, ovvero proposizioni prive di ulteriori connettivi. I connettivi operano sulle proposizioni per generarne di più complesse. La semantica dei connettivi, cioè il significato attribuito a ciascuno, sarà approfondita nella sezione successiva. In linea di principio, è possibile associare \wedge alla congiunzione “e”, \vee alla disgiunzione “o” (in senso non esclusivo, come nel latino “vel”), \rightarrow alla locuzione “se... allora” e \neg alla negazione “non”.

Definizione 1. *Sia L un linguaggio proposizionale. L'insieme delle formule del linguaggio L , indicato con Frm_L (o, semplicemente, Frm), è definito ricorsivamente dalle seguenti regole:*

- $\top \in Frm$ e $\perp \in Frm$;
- le variabili proposizionali appartengono a Frm ;
- se $A \in Frm$ e $B \in Frm$, allora anche $(A \wedge B)$, $(A \vee B)$ e $(A \rightarrow B)$ appartengono a Frm ;
- se $A \in Frm$, allora anche $(\neg A)$ appartiene a Frm .

L'insieme delle formule atomiche è costituito dalle variabili proposizionali insieme a \perp e \top , mentre tutte le altre formule sono definite “composte”.

La logica proposizionale si fonda su due aspetti fondamentali: la *sintassi*, che ne definisce la struttura formale e la costruzione delle *formule ben formate* (fbf); e la *semantica*, che ne determina la valutazione di verità.

2.2.1 Sintassi

La sintassi della logica proposizionale viene definita rispetto a un alfabeto specifico, denotato $\mathcal{A}_{\mathcal{PL}}$, strutturato come segue:

$$\mathcal{A}_{\mathcal{PL}} = \mathcal{AP} \cup \Omega_{\mathcal{PL}} \cup \{(\,,\,)\}$$

p	q
0	0
0	1
1	0
1	1

Tabella 2.1: Tabella di verità per le possibili interpretazioni su p e q .

dove \mathcal{AP} rappresenta l'insieme infinito e numerabile dei letterali, identificati da lettere minuscole (ad es. p, r, s, \dots), e $\Omega_{\mathcal{PL}}$ denota l'insieme dei connettivi logici. Nella trattazione, verrà fatto riferimento a $\{\perp, \top\}$ anche con le notazioni $\{Falso, Vero\}$ o $\{0, 1\}$, a seconda del contesto.

Dato l'alfabeto \mathcal{APL} , una formula ben formata (fbf) della logica proposizionale viene definita mediante la seguente grammatica, utilizzando le lettere greche $\{\varphi, \psi, \dots\}$, con $p \in \mathcal{AP}$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi.$$

I connettivi rimanenti, come \vee , \rightarrow , e \leftrightarrow , possono essere ottenuti in maniera triviale dalla grammatica proposta. Si indica inoltre con $|\varphi|$ la lunghezza di una formula, ossia il numero totale di simboli che la compongono; per esempio, per $\varphi = p \wedge q$, si ha $|\varphi| = 3$.

2.2.2 Semantica

Una volta definita la sintassi, la semantica di un sistema logico assegna un significato alle formule attraverso strutture interpretative. Ogni interpretazione attribuisce un valore di verità a ciascuna formula ben formata, basandosi sulla valutazione delle variabili atomiche che la compongono. Ad esempio, in una logica con $\mathcal{AP} = \{p, q\}$, esistono $2^2 = 4$ possibili interpretazioni, rappresentabili tramite una tabella di verità come illustrato in Figura (2.1).

Per garantire chiarezza, si indica con $\mathcal{I}(p)$ il valore di verità assegnato alla proposizione p in una data interpretazione, per esempio nell'interpretazione $\langle p = 1, q = 0 \rangle$.

La valutazione di una formula non dipende unicamente dai valori delle singole variabili, ma anche dalle trasformazioni determinate dai connettivi. In particolare,

si definisce che un'interpretazione \mathcal{I} soddisfa una formula φ , indicato con $\mathcal{I} \models \varphi$, secondo le seguenti regole:

$$\begin{aligned} \mathcal{I} \models p & \quad \text{se e solo se} \quad \mathcal{I}(p) = 1, \\ \mathcal{I} \models \neg\varphi & \quad \text{se e solo se} \quad \mathcal{I} \not\models \varphi, \\ \mathcal{I} \models \varphi \wedge \psi & \quad \text{se e solo se} \quad \mathcal{I} \models \varphi \text{ e } \mathcal{I} \models \psi. \end{aligned}$$

Un'interpretazione che soddisfa una data formula è definita *modello* e viene indicata con \mathcal{M} . A seconda del numero di modelli posseduti, una formula può essere classificata come:

- *soddisfacibile* (o *contingente*), se esiste almeno un modello;
- *tautologica*, se ogni interpretazione è un modello;
- *contraddittoria*, se non esiste alcun modello.

Durante lo sviluppo della trattazione, si farà frequentemente riferimento a formule equivalenti: due formule ben formate φ e ψ sono dette equivalenti, denotate con $\varphi \equiv \psi$, se condividono la medesima interpretazione, cioè se presentano identiche tabelle di verità.

2.3 Minimizzazione di formule proposizionali

La minimizzazione logica costituisce un problema di notevole importanza nell'ambito della sintesi logica, con applicazioni rilevanti in elettronica digitale, nella progettazione di circuiti integrati e nell'analisi di algoritmi. Il quesito fondamentale consiste nel determinare se, data una formula φ , esista una formula equivalente φ' avente una cardinalità inferiore a un certo valore k .

Nel caso di formule conformi a determinate forme normali, tale problema risulta appartenere al secondo livello della gerarchia polinomiale, precisamente a Σ_2^P considerabile quindi, nel peggiore dei casi, come un problema PSPACE.

2.3.1 Forme normali: CNF e DNF

Spesso è utile trasformare una formula ben formata (fbf) in un'altra equivalente che adotti una forma canonica prestabilita. Questo processo si realizza sostituendo

alcune componenti della formula con altre formule equivalenti, fino a raggiungere la struttura desiderata. La forma ottenuta, detta “normale”, è tale che non si possono applicare ulteriori riscritture.

Le forme normali considerate sono la forma normale congiuntiva (CNF) e quella disgiuntiva (DNF).

Una disgiunzione di fbf P_1, P_2, \dots, P_n è una formula della forma

$$P_1 \vee P_2 \vee \dots \vee P_n,$$

mentre una congiunzione di fbf P_1, P_2, \dots, P_n è una formula della forma

$$P_1 \wedge P_2 \wedge \dots \wedge P_n.$$

Si ricorda che un letterale è una proposizione atomica o la sua negazione.

Definizione 2 (Forma normale congiuntiva). *Una formula ben formata (fbf) P è detta in forma normale congiuntiva (CNF) se e solo se*

$$P = P_1 \wedge \dots \wedge P_n \quad \text{con } n \geq 1,$$

dove, per ogni $i = 1, \dots, n$, P_i è una disgiunzione di letterali.

Definizione 3 (Forma normale disgiuntiva). *Una fbf P è detta in forma normale disgiuntiva (DNF) se e solo se*

$$P = P_1 \vee \dots \vee P_n \quad \text{con } n \geq 1,$$

dove, per ogni $i = 1, \dots, n$, P_i è una congiunzione di letterali.

Teorema 1. *Per ogni fbf P esistono una forma normale congiuntiva P_C e una forma normale disgiuntiva P_D , tali che*

$$P \equiv P_C \quad \text{e} \quad P \equiv P_D.$$

2.3.2 Minimizzazione DNF

Nella presente tesi, la minimizzazione di formule DNF (Forma Normale Disgiuntiva), nota in letteratura come MIN-DNF [43], rappresenta un tassello cruciale dell'approccio LUMEN. Tale minimizzazione, storicamente nota come *two-level logic minimization*, riveste un'importanza fondamentale nell'ottimizzazione dei circuiti digitali. L'obiettivo principale è duplice:

- Garantire l'interpretabilità delle regole estratte
- Mantenere la complessità delle regole al minimo possibile

Complessità Computazionale. Il problema MIN DNF (trovare una formula $\varphi' \equiv \varphi$ con $|\varphi'| \leq k$) è classificato come Σ_2^P -complete, indicando un'elevata complessità computazionale che richiede sofisticati algoritmi di risoluzione.

Tassonomia degli Algoritmi. La classificazione degli algoritmi per la minimizzazione di formule in Forma Normale Disgiuntiva (DNF) si articola secondo un duplice criterio tassonomico che considera sia la qualità delle soluzioni prodotte sia le strategie computazionali adottate.

Dal punto di vista della qualità delle soluzioni, gli algoritmi si distinguono in due categorie principali: quelli che producono *formule globalmente ottime* e quelli che generano *formule localmente ottime*. Questa distinzione riflette la capacità dell'algoritmo di raggiungere effettivamente il minimo assoluto nel numero di token sintattici della formula risultante.

La seconda dimensione classificatoria riguarda l'approccio computazionale adottato, dividendo l'insieme delle soluzioni algoritmiche in *metodi esatti* ed *euristici*. I metodi esatti garantiscono la produzione di formule con il numero minimo assoluto di token sintattici, preservando rigorosamente l'equivalenza semantica con la formula di partenza. Tuttavia, tale garanzia di ottimalità comporta una complessità computazionale che cresce esponenzialmente rispetto alla cardinalità dell'input, rendendo questi approcci impraticabili per istanze di dimensioni significative.

I metodi euristici, al contrario, sacrificano la garanzia di ottimalità globale in favore di una maggiore efficienza computazionale. Questi algoritmi impiegano strategie di approssimazione che, pur non assicurando il raggiungimento del minimo

assoluto, tendono a produrre soluzioni di qualità elevata attraverso l'applicazione di funzioni obiettivo specificamente progettate per minimizzare la ridondanza e approssimare il minimo globale.

Le formule prodotte dagli algoritmi euristici vengono definite *localmente ottime* poiché rappresentano il miglior risultato ottenibile nell'ambito della specifica computazione effettuata, senza tuttavia garantire l'ottimalità rispetto all'intero spazio delle soluzioni possibili. Il termine "locale" sottolinea quindi la natura contingente dell'ottimizzazione raggiunta, circoscritta al particolare percorso esplorativo seguito dall'algoritmo.

L'attuale panorama algoritmico presenta diverse soluzioni consolidate per entrambe le categorie. Nell'ambito degli algoritmi euristici, lo stato dell'arte è rappresentato principalmente dalla famiglia di algoritmi ESPRESSO-II [41], che implementa tecniche sofisticate di riduzione basate su implicanti primi, e dall'algoritmo BOOM [23], che adotta strategie di ottimizzazione basate su operazioni booleane avanzate.

Per quanto concerne gli algoritmi esatti, le soluzioni più riconosciute e utilizzate nella letteratura scientifica includono ESPRESSO-EXACT [41], che estende l'approccio di ESPRESSO-II garantendo l'ottimalità globale attraverso un'esplorazione esaustiva dello spazio delle soluzioni, e SCHERZO [13], che implementa tecniche di branch-and-bound per la ricerca del minimo assoluto mantenendo una maggiore efficienza computazionale rispetto agli approcci puramente esaustivi.

Implicazioni nell'Interpretabilità. L'adozione di questi algoritmi nell'apprendimento automatico offre diversi vantaggi:

- Riduzione della ridondanza nelle regole estratte
- Miglioramento della leggibilità dei modelli
- Facilitazione dell'analisi e della verifica delle decisioni

2.4 Modelli simbolici

Si procede adesso alla trattazione di definizioni fondamentali riguardo i modelli di machine learning che andranno ad accompagnare l'intera esposizione. Si ricordi

che per quanto LUMEN verrà trattato all'interno di tale tesi come metodo che lavora con Decision forest esso è intrinsecamente adatto all'applicazione di generici modelli nell'ambito del machine learning, come vedremo in seguito.

2.4.1 Dataset

Nel presente paragrafo si introduce il concetto di dataset, elemento fondamentale per l'apprendimento dei modelli simbolici. Un dataset rappresenta una raccolta organizzata di dati, in cui ciascuna istanza è descritta attraverso un insieme di attributi. Tali dati costituiscono la base per l'estrazione di conoscenze, poiché definiscono il linguaggio del modello tramite l'insieme dei letterali proposizionali.

Definizione 4. *Un dataset è un insieme di m istanze $I = \{I_1, \dots, I_m\}$, ciascuna delle quali è descritta dai valori di n attributi $A = \{A_1, \dots, A_n\}$.*

Come da esempio:

Instance	A_1	A_2	A_3	\dots	A_n
I_1	1.6	0.5	1.9	\dots	0.5
I_2	2.9	8.8	4.3	\dots	4.0
I_3	6.0	0.8	1.8	\dots	5.5
\vdots					\vdots
I_m	6.5	0.8	1.8	\dots	4.2

Da un dataset I viene appreso un modello simbolico, il quale definisce il proprio linguaggio attraverso un insieme di letterali proposizionali P . Questi letterali rappresentano conoscenze relative agli attributi delle istanze, in modo da codificare le informazioni in forma simbolica:

$$p \equiv A_i \bowtie a \quad \text{con } a \in A_i,$$

dove \bowtie denota l'operatore di confronto appropriato.

Inoltre, le relazioni tra i letterali proposizionali possono essere formalizzate mediante una teoria \mathcal{T} , che sfrutta la natura intrinseca dei letterali. Ad esempio:

$$\text{se } p \equiv A_1 \leq 30 \text{ e } q \equiv A_1 \leq 20, \text{ allora } (q \implies p) \in \mathcal{T}.$$

Questa struttura consente di esprimere e gestire in modo preciso le conoscenze derivate dai dati.

2.4.2 Alberi e foreste decisionali

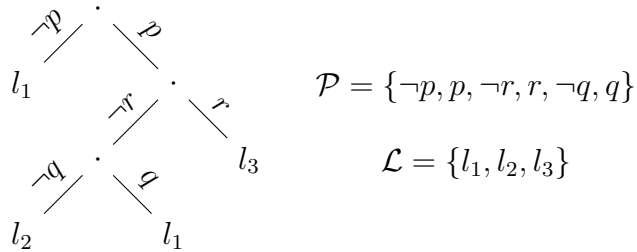
In questo paragrafo si introducono gli alberi e le foreste decisionali, strumenti fondamentali per la modellazione e l'analisi di processi decisionali basati su regole logiche. L'albero di decisione offre una rappresentazione gerarchica e intuitiva, in cui ogni nodo interno esegue un test su un letterale proposizionale e le foglie indicano le classi assegnate. Tale struttura consente di interpretare in maniera trasparente il ragionamento sottostante alla classificazione o alla presa di decisioni.

Definizione 5. Sia \mathcal{L} un insieme di classi e P un insieme finito di letterali proposizionali. Allora, un albero di decisione (su \mathcal{L}) è una tupla

$$\tau = \langle V, E, l, e \rangle$$

dove $\langle V, E \rangle$ è un albero diretto binario completo, l è una funzione di etichettatura delle foglie che assegna a ciascun nodo foglia in V una classe appartenente a \mathcal{L} , ed e è una funzione di etichettatura degli archi che assegna ad ogni arco in E una decisione da $\{p, \neg p \mid p \in P\}$, in modo tale che due nodi fratelli abbiano sempre decisioni opposte.

Come da esempio:



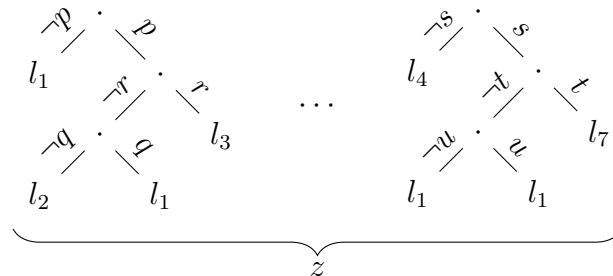
Con i metodi di apprendimento simbolico i dati e le relazioni vengono rappresentati utilizzando strutture simboliche. Gli alberi di decisione sono un classico esempio di modelli di apprendimento simbolico, in cui i rami dell'albero rappresentano tipicamente formule della logica proposizionale. Nonostante la loro efficacia, gli alberi di decisione soffrono di una capacità limitata di generalizzare su nuovi dati. Per affrontare tale problema, detto “di overfitting”, vengono comunemente utilizzati ensemble di alberi di decisione indipendenti, noti come foreste decisionali,

per migliorare la capacità di generalizzazione degli alberi singoli. Le foreste decisionali sono per loro natura simboliche, ma includono una componente funzionale per amalgamare l'output dei singoli alberi e possono pertanto essere classificate come tecniche miste simbolico/sub-simboliche; l'algoritmo più noto per l'apprendimento di foreste decisionali, ovvero il random forest, produce foreste decisionali per la classificazione o la regressione in cui la funzione di aggregazione è una semplice maggioranza.

Le foreste decisionali, costituite da un insieme di alberi di decisione, vengono impiegate quindi per migliorare la robustezza e l'accuratezza del modello, aggregando i giudizi di più alberi. Queste tecniche, ampiamente adottate in diversi ambiti applicativi, permettono di ridurre il rischio di overfitting e di ottenere una stima più affidabile delle decisioni finali.

Definizione 6. Una foresta di decisione $F = \{\tau_1, \dots, \tau_z\}$ (su \mathcal{L}) è un insieme di z alberi di decisione (su \mathcal{L}).

Come da esempio:



Si osservi, tuttavia, che questo modello comporta un chiaro trade-off: pur garantendo una maggiore robustezza contro l'overfitting, esso comporta una significativa perdita di trasparenza e di capacità esplicativa, che era invece presente nei singoli alberi decisionali.

In domini critici come la sanità, la finanza e il diritto, dove le decisioni dei modelli devono risultare comprensibili, giustificabili e trasparenti, l'interpretabilità riveste la stessa importanza delle prestazioni predittive. Come afferma O'Neil, i modelli opachi rappresentano il lato oscuro dei big data. In risposta, il campo

della eXplainable Artificial Intelligence (XAI) si è sviluppato, offrendo strumenti e tecniche per fare luce sui modelli complessi. Tali strumenti e tecniche, in cui si va a posizionare anche quella proposta, verranno affrontate nel capitolo successivo.

Lavori correlati

L'estrazione di regole basate su ensemble si situa all'intersezione di molteplici filoni di ricerca, tra cui l'interpretabilità spiegabile (XAI), la distillazione della conoscenza e i modelli teorici dell'apprendimento. Il metodo presentato nella tesi in oggetto, LUMEN, si ispira al paradigma del Probably Approximately Correct (PAC) [44], in cui argomentazioni basate sul campionamento contribuiscono a controllare la fedeltà delle formule logiche, facendo da ponte con le note complessità dell'apprendimento e della minimizzazione delle rappresentazioni in forma di DNF [4, 35]. Di seguito si illustrano in modo approfondito i principali contributi in tema di estrazione globale di regole, spiegazioni locali, interpretabilità basata su concetti per reti neurali, distillazione della conoscenza, minimizzazione logica e considerazioni teoriche sull'apprendimento, mettendo in relazione ciascun filone con il framework concettuale proposto.

3.1 Spiegazioni locali e globali

L'interpretabilità dei modelli di apprendimento automatico rappresenta una sfida cruciale in molti ambiti applicativi, in particolare in contesti in cui la trasparenza e la fiducia nelle decisioni automatizzate sono elementi imprescindibili. In que-

sto capitolo si analizzano e si confrontano due principali approcci interpretativi: l'estrazione globale e quella locale di regole. Tali metodologie, pur condividendo l'obiettivo comune di rendere esplicabile il funzionamento di modelli complessi, si differenziano sia per la portata che per il livello di dettaglio delle spiegazioni prodotte.

Estrazione globale di regole. L'interpretabilità globale mira a catturare l'intera logica decisionale del modello, piuttosto che concentrarsi su singole predizioni. Tra gli approcci tradizionali si annoverano metodi come InTrees, che identificano i pattern più frequenti nei percorsi decisionali dell'ensemble e li semplifica in regole più chiare ed esplicative. Altre proposte, meno convenzionali, includono Born Again Trees (BATrees), le quali si basano sulla generazione di nuovi campioni da sottoporre ad un ensemble già addestrato. Tale operazione consente di costruire un dataset ampliato dal quale estrarre un albero decisionale semplice, in grado di sintetizzare il comportamento dell'ensemble in maniera trasparente. Un'altra tecnica recentemente proposta, RuleCOSI+, enfatizza la fase di decomposizione dell'ensemble d'ingresso per successivamente costruire regole approssimative che ne imitano il funzionamento. Metodi come TREPAN utilizzano algoritmi induttivi specifici per generare dataset su cui costruire alberi decisionali. Questi alberi hanno lo scopo di spiegare il comportamento di una rete neurale, rendendo interpretabile il suo funzionamento. Vale la pena notare che TREPAN non si limita alle reti neurali, ma può essere applicato anche ad altri modelli di tipo "black box", ovvero modelli complessi il cui processo decisionale interno è difficile da interpretare. Infine, REFNE utilizza un meccanismo di campionamento innovativo dello spazio delle feature, producendo così risultati analoghi in termini di spiegabilità.

Spiegazione locale. I metodi di spiegazione locale sono orientati a chiarire le ragioni per cui un modello assegna una specifica etichetta a una determinata istanza. Tecniche consolidate, come LIME e SHAP, o approcci più recenti come Bellatrex, forniscono spiegazioni mirate per ogni singolo caso. In particolare, il lavoro di Izza e Marques-Silva affronta il problema di generare spiegazioni locali formali per le decisioni prese dagli ensemble di alberi, risolvendo un problema computazionalmente impegnativo, classificato come D^P -hard, per ogni istanza.

3.1.1 Approcci alla IA interpretabile

Interpretabilità basata su concetti nelle reti neurali. Nel contesto delle reti neurali, per affrontare il problema della spiegabilità è emersa l'idea di spiegazioni basate su concetti. Ad esempio, il metodo TCAV [28] identifica concetti ad alto livello, comprensibili dall'uomo, presenti all'interno dei modelli profondi e ne quantifica l'importanza per le predizioni. Allo stesso modo, Network Dissection [2] mette in relazione specifiche unità della rete con concetti semantici, mentre i Concept Bottleneck Models [29] progettano architetture di rete in cui la previsione dei concetti rappresenta un passaggio preliminare prima dell'output finale. Tali metodi evidenziano che le reti neurali possono, in linea di principio, generare un insieme significativo di proposizioni – variabili concettuali – che, se opportunamente estratte e combinate, possono dare luogo a una tabella di verità simile a quella di una foresta decisionale.

Distillazione della conoscenza. La distillazione della conoscenza [22] nasce originariamente nell'ambito della compressione dei modelli [9] e si concentra sul trasferimento di conoscenza da modelli complessi (come reti neurali o ensemble di reti) a modelli più semplici e trasparenti (ad esempio, alberi decisionali o liste di decisioni). Oltre al tradizionale obiettivo di migliorare l'efficienza, le tecniche di distillazione simbolica [20] della conoscenza e gli approcci correlati integrano modelli di deep learning con sistemi di ragionamento simbolico. In tale prospettiva, il metodo esaminato in tale tesi si inserisce in maniera coerente: l'estrazione di regole logiche minimali da ensemble complessi offre una rappresentazione interpretabile che può essere successivamente integrata in framework di rappresentazione e ragionamento della conoscenza [37] per compiti di inferenza più sofisticati, controlli di consistenza o integrazione con basi di conoscenza esperte.

Apprendimento PAC e complessità nell'apprendimento di DNF. Nel campo dell'apprendimento automatico una domanda fondamentale è capire quanto sia complesso apprendere le formule in forma normale disgiuntiva (DNF) a partire da esempi. Questo problema è stato studiato in profondità nel contesto dell'apprendimento probabilistico (PAC, Probabilistic Approximate Correct) [44].

È noto che alcune classi di funzioni booleane, come le liste di decisione con un numero limitato di elementi, gli alberi di decisione di profondità fissata o le formule DNF con un numero limitato di termini, possono essere apprese in maniera PAC sotto certe ipotesi [39, 4]. Tuttavia, quando si tratta di apprendere formule DNF in generale, il problema diventa NP-hard [35, 27] e la complessità, misurata in termini di dimensione VC (Vapnik-Chervonenkis), risulta particolarmente elevata.

Esistono varianti del modello PAC classico, come il modello di apprendimento attivo [25, 8], che permettono di apprendere le formule DNF in maniera più flessibile e applicabile a situazioni pratiche. In questo contesto l'approccio oggetto di tesi può, attraverso la manipolazione di appropriati parametri, ottenere una formula che sia sufficientemente accurata e interpretabile, piuttosto che ottenere una ricostruzione esatta di un modello complesso.

Infatti, pur utilizzando metodi come gli ensemble (ad esempio, le foreste casuali) per codificare confini decisionali complessi, estrarre una formula DNF minima equivalente a tutto l'ensemble presenta le stesse sfide computazionali. Perciò, uno degli obiettivi della tesi risulta anche bilanciare la tracciabilità computazionale e la capacità di approssimazione, ispirandosi alle idee del framework PAC, nella situazione in cui non si vuole cercare una ricostruzione perfettamente esatta del concetto originale.

3.2 Apporto di LUMEN

Dal precedente discorso e da un'analisi approfondita dei metodi esistenti, emergono numerose caratteristiche utili per valutare gli algoritmi di spiegazione globale applicati a modelli black-box. In particolare:

- **Decomponibilità:** indica se il metodo di spiegazione utilizza internamente componenti del modello originale, quali regole o rami decisionali.
- **Pedagogia:** si riferisce all'adozione di un approccio *insegnante-studente* per replicare il comportamento del modello.
- **Indipendenza dal dataset:** evidenzia la possibilità di applicare il metodo senza avere accesso diretto al set di addestramento originale.

	InTrees	BATrees	TREPAN	REFNE	RuleCOSI+	LUMEN
<i>decomponibilità</i>	✓	✓	×	×	✓	✓
<i>pedagogia</i>	×	×	✓	✓	×	✓
<i>indipendenza dal dataset</i>	×	✓	×	×	×	✓
<i>garanzia di minimalità</i>	×	×	×	×	×	✓
<i>fedeltà al dataset</i>	×	×	×	×	×	✓
<i>fedeltà universale</i>	×	×	×	×	×	✓
<i>applicabilità locale</i>	×	✓	✓	✓	×	✓
<i>copertura di tutte le classi</i>	✓	✓	✓	✓	×	✓
<i>mantenimento dell'alfabeto originale</i>	✓	✓	×	×	×	✓

Figura 3.1: Caratteristiche degne di nota degli algoritmi di spiegazione globale selezionati per foreste casuali e non.

- **Garanzia di minimalità:** verifica se le regole estratte sono state formalmente dimostrate come minimali.
- **Fedeltà al dataset:** valuta se il modello surrogato riproduce esattamente le predizioni del modello originale sui dati di addestramento, mentre la **fedeltà universale** accerta la riproduzione perfetta per ogni possibile input.
- **Applicabilità locale:** riguarda la capacità delle regole globali di chiarire anche le predizioni individuali.

In particolare, secondo quanto proposto da Guidotti et al. [21], si ritiene che un algoritmo di spiegazione globale supporti la spiegazione locale se ogni singolo caso è coperto da una regola unica e leggibile (ad esempio, una congiunzione), anziché da una struttura ramificata come una lista di decisioni.

La Figura Fig. 3.1 riassume queste dimensioni per alcuni algoritmi rappresentativi di spiegazione globale. Il confronto ha incluso InTrees [17], BATrees [6], TREPAN [14], REFNE [46], RuleCOSI+ [32] e l'oggetto di tesi LUMEN (Logic-based Unified Minimal Extractor of Notions). La maggior parte degli approcci non fornisce garanzie formali di minimalità o fedeltà perfetta, concentrandosi piuttosto su ricostruzioni approssimative (euristiche) basate sui dati esistenti o su tecniche di campionamento. La proposta che verrà meglio analizzata nel succe-

sivo capitolo (LUMEN), invece, è stata appositamente progettata per soddisfare simultaneamente tutti questi criteri.

Framework logico per l'interpretazione di ensemble

Questo capitolo si articola in due sezioni principali. La prima parte è dedicata all'approfondimento del problema dell'estrazione di regole minime da random forest. Verranno presentati i teoremi sulla complessità computazionale di tale problema, dimostrati nell'ambito di questa ricerca. Questi risultati teorici non solo rappresentano un contributo significativo nel campo dell'interpretabilità dei modelli di machine learning, ma hanno anche costituito le fondamenta concettuali per lo sviluppo del primo prototipo di LUMEN. Nella seconda parte verrà illustrato il framework logico oggetto della tesi il quale soddisfa i criteri affrontati nel precedente capitolo fornendo una spiegazione unificata, minima e fedele per i modelli di ensemble.

4.1 Estrazione sistematica di regole minime da foreste decisionali

Cardine, da cui sono partiti gli studi presentati nella tesi, è stato lo studio di come estrarre regole da alberi e poi da foreste decisionali; per fare ciò sono necessarie

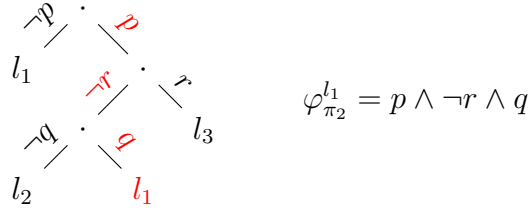
ulteriori definizioni le quali hanno contribuito alla stesura di diversi teoremi utili per la comprensione della complessità di tale problema [34].

4.1.1 Formule di cammino e di classe negli alberi

Dato un albero di decisione τ su \mathcal{L} e una classe $L \in \mathcal{L}$, allora:

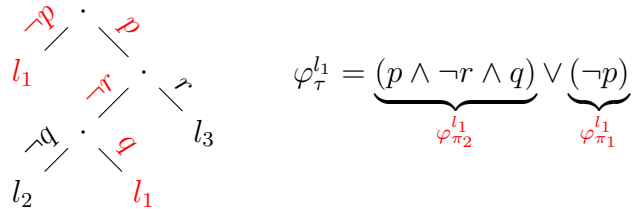
Definizione 7 (Formula di cammino L nell'albero). *Dato un cammino π in τ dalla radice a una foglia etichettata con L (un cammino L), la congiunzione di tutte le decisioni lungo π viene chiamata formula di cammino L , ed è denotata da φ_π^L .*

Come da esempio:



Definizione 8 (Formula di classe nell'albero). *La disgiunzione di tutte le formule di cammino L in τ viene chiamata formula di classe nell'albero, ed è denotata da φ_τ^L .*

Come da esempio:



4.1.2 Formule di cammino e di classe nelle foreste

Dato un insieme (foresta) di decisione F su \mathcal{L} contenente z alberi, e una classe $L \in \mathcal{L}$, allora:

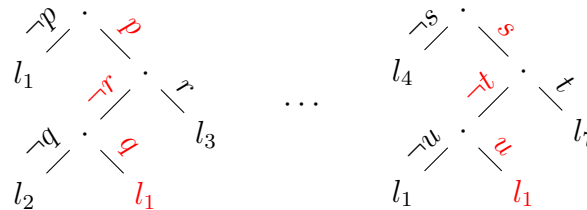
Definizione 9 (Formula parziale L della foresta). *Sia $\{\tau_{i_1}, \dots, \tau_{i_t}\} \subseteq F$ una collezione di alberi e, per ciascun albero τ_{i_j} ($1 \leq j \leq t$), sia dato un cammino L (cioè un cammino che termina in una foglia etichettata con L) denotato da π_{i_j} . La congiunzione di tutte le formule di cammino L relative ai cammini*

$$\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_t},$$

viene detta formula parziale di percorso L della foresta, e viene denotata da

$$\varphi_{\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_t}}^L.$$

Come da esempio:



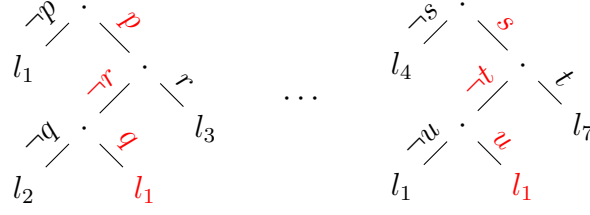
$$\varphi_{\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_t}}^L = (p \wedge \neg r \wedge q) \wedge (s \wedge \neg t \wedge u)$$

Definizione 10 (Formula di cammino L della foresta). *Una formula parziale di cammino nella foresta*

$$\varphi_{\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_t}}^L,$$

con $t > z/2$, viene detta formula di cammino L della foresta, e si denota con lo stesso simbolo.

Come da esempio:

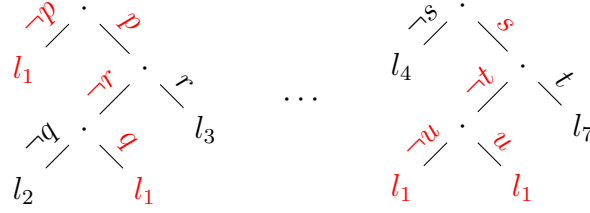


$$\varphi_{\pi_{i_1}, \pi_{i_2}, \dots, \pi_{i_t}}^L = (p \wedge \neg r \wedge q) \wedge \dots \wedge (s \wedge \neg t \wedge u)$$

Definizione 11 (Formula di classe nella foresta). *La disgiunzione di tutte le possibili formule L -path forest viene detta formula L di classe della foresta, e si denota con*

$$\varphi_F^L.$$

Come da esempio:



$$\varphi_F^L = \left((p \wedge \neg r \wedge q) \wedge (s \wedge \neg t \wedge \neg u) \right) \vee \dots \vee \left((\neg p) \wedge (s \wedge \neg t \wedge u) \right)$$

4.1.3 Regole derivate da una foresta decisionale

Si nota come un modello per la formula della foresta L -class (formula di classe L della foresta) corrisponda esattamente ad un'istanza classificata come L dalla foresta decisionale.

Per formalizzare tale comportamento si definisce il concetto di **regola forte**.

Definizione 12 (Regola forte L -class). *Dati una foresta decisionale F , appresa da un dataset \mathcal{I} , su \mathcal{L} , e una classe $L \in \mathcal{L}$, una regola forte L -class è un oggetto della forma*

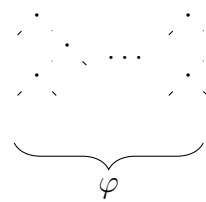
$$\varphi \Leftrightarrow L,$$

dove φ (l'antecedente) è una formula proposizionale nel linguaggio di F , tale che, per ogni $I \in \mathcal{I}$, si ha

$$I \models \varphi \quad \text{se e solo se} \quad F(I) = L.$$

Come da esempio:

Instance	A_1	A_2	A_3	\dots	A_n
I_1	1.6	0.5	1.9	\dots	0.5
I_2	2.9	8.8	4.3	\dots	4.0
I_3	6.0	0.8	1.8	\dots	5.5
\vdots					\vdots
I_m	6.0	0.8	1.8	\dots	4.2

\Leftrightarrow

 $\Leftrightarrow L$

$\underbrace{\hspace{10em}}_{\varphi}$

4.1.4 Problemi di minimizzazione per regole forti e deboli

Il modello, considerato come antecedente nella formula di foresta L -class, determina banalmente una regola forte L -class. Un aspetto importante riguarda la potenziale ridondanza, dovuta al fatto che alberi e foreste decisionali sono solitamente appresi tramite algoritmi subottimali che ignorano la teoria sottostante \mathcal{T} . Ciò ha ispirato la formulazione di un problema di minimizzazione, definibile nelle sue versioni funzionale e decisionale.

4.1.5 Problema della regola forte minima

Definizione 13 (Problema della regola forte minima). *Una regola forte L -class minima è una regola forte L -class $\varphi \Leftrightarrow L$ tale che, per ogni regola forte $\varphi' \Leftrightarrow L$ con*

$$\varphi \equiv_{\mathcal{T}}^{\mathcal{I}} \varphi',$$

cioè φ e φ' sono equivalenti modulo \mathcal{T} (almeno rispetto alle istanze in \mathcal{I}), si

ha

$$|\varphi| \leq |\varphi'|.$$

Definizione 14 (Problema decisionale della regola forte minima). *Dati una foresta decisionale F , appresa da un dataset \mathcal{I} , su \mathcal{L} , una classe $L \in \mathcal{L}$ e un numero q , si chiede se esista una regola forte L -class $\varphi \Leftrightarrow L$ tale che*

$$|\varphi| \leq q.$$

Teorema 2. *Dati una foresta decisionale F , un dataset \mathcal{I} , una classe $L \in \mathcal{L}$, una teoria \mathcal{T} e un numero q , il problema di stabilire se esiste una regola forte L -class $\varphi \Leftrightarrow L$ tale che $|\varphi| \leq q$ è in NEXPTIME.*

4.1.6 Regole deboli

Poiché il problema della regola forte minima è in NEXPTIME, per migliorare la trattabilità si definiscono delle regole *deboli* in due versioni simmetriche:

Regola debole destra L -class. Una *regola debole destra L -class* è un oggetto della forma

$$\varphi \Rightarrow L,$$

tale che, per ogni istanza I , se $I \models \varphi$ allora $F(I) = L$.

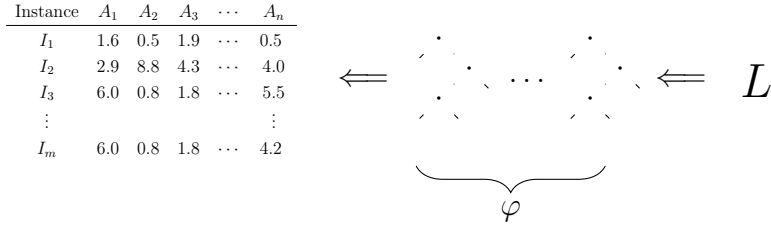
Instance	A_1	A_2	A_3	\dots	A_n
I_1	1.6	0.5	1.9	\dots	0.5
I_2	2.9	8.8	4.3	\dots	4.0
I_3	6.0	0.8	1.8	\dots	5.5
\vdots					\vdots
I_m	6.0	0.8	1.8	\dots	4.2

$$\Rightarrow \underbrace{\{I_1, I_2, I_3, \dots, I_m\}}_{\varphi} \Rightarrow L$$

Regola debole sinistra L -class. Una *regola debole sinistra L -class* è un oggetto della forma

$$L \Rightarrow \varphi,$$

tale che, per ogni istanza I , se $F(I) = L$ allora $I \models \varphi$.



4.1.7 Problemi di complessità per le regole deboli

Si è giunti quindi alle seguenti due importanti conclusioni:

Teorema 3. *Dati una foresta decisionale F su \mathcal{L} , una classe $L \in \mathcal{L}$, una teoria \mathcal{T} e un numero q , il problema di stabilire se esiste una regola debole destra L -class $\varphi \Rightarrow L$ tale che $|\varphi| \leq q$ è in NP.*

Teorema 4. *Dati una foresta decisionale F su \mathcal{L} , una classe $L \in \mathcal{L}$, una teoria \mathcal{T} e un numero q , il problema di stabilire se esiste una regola debole sinistra non banale L -class $L \Rightarrow \varphi$ tale che $|\varphi| \geq q$ è in NEXPTIME.*

4.2 Un approccio di estrazione e minimizzazione

Tutto ciò ha portato dunque ad una formulazione di un primo approccio illustrato in Fig. 4.1 che come si noterà nella prossima sezione, ha dato luce a LUMEN. I passi fondamentali sono:

- **L'estrazione dell'alfabeto** dalla foresta F ;

- **La costruzione di una tabella DNF T :** ogni termine rappresenta l'infinità di istanze che verrebbero classificate nella stessa classe da F ;
- **L'etichettatura di T mediante F come oracolo** (in tempo lineare), escludendo quelle non consistenti con la teoria \mathcal{T} ;
- **La minimizzazione dell'intera tabella T** (per ottenere una regola forte) o di una sua porzione (per ottenere una regola debole) utilizzando minimizzazione esatta (Quine-like [36]) oppure minimizzazione euristica (ESPRESSO-like), può essere dimostrato che fornisce regole consistenti con \mathcal{T} .

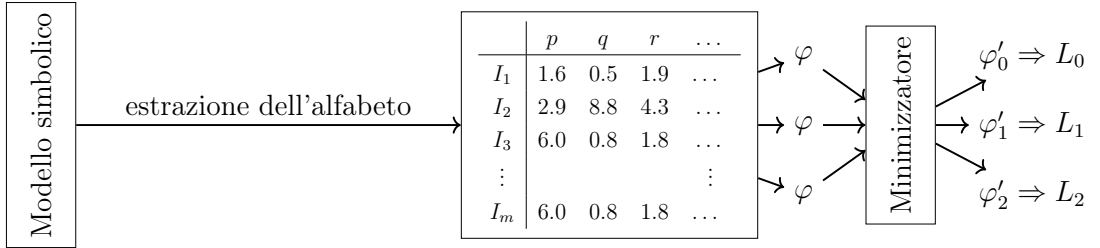


Figura 4.1: Schema primo approccio per estrazione e minimizzazione.

4.3 Estrazione e minimizzazione logica con LUMEN

LUMEN combina una strategia decomponenziale e pedagogica, preserva l'alfabeto originale delle soglie sulle feature derivato dall'ensemble e non richiede l'accesso ai dati di training. Garantisce la fedeltà estraendo una singola formula in forma normale disgiuntiva (DNF) per ogni classe, che riproduce esattamente le decisioni dell'ensemble originale, assicurando così la fedeltà sia rispetto al dataset sia rispetto all'universo considerato. Una procedura di ottimizzazione logica minimizza formalmente la DNF risultante, producendo una rappresentazione estremamente compatta. Poiché assicurare la fedeltà su tutto lo spazio delle feature può risultare computazionalmente oneroso, LUMEN supporta un'approssimazione parametrizzata che rilassa la copertura di alcune regioni, offrendo agli utenti il controllo sul compromesso tra esattezza e tempo di esecuzione. Inoltre, la classificazione di ogni istanza dipende da un singolo termine logico all'interno della formula DNF globale, consentendo allo stesso insieme di regole di spiegare le predizioni individuali. Tale

attivazione a termine singolo fornisce l'ambito locale necessario per comprendere perché il modello assegna una determinata etichetta a una data istanza. Sebbene le specifiche siano ottimizzate per il caso delle foreste casuali, le idee sottostanti si estendono naturalmente anche agli ensemble di boosting e, con alcuni aggiustamenti concettuali, alle reti neurali, a condizione che sia possibile identificare variabili proposizionali significative dalle loro rappresentazioni interne.

Si approfondiranno ora i dettagli del framework LUMEN.

4.3.1 Dal modello a tabelle di verità

Si consideri un contesto di classificazione in cui vi è uno spazio d'ingresso $\mathcal{X} \subset \mathbb{R}^d$ e un insieme finito di etichette $\mathcal{Y} = \{1, \dots, K\}$. Sia $F : \mathcal{X} \rightarrow \mathcal{Y}$ un modello basato su un ensemble di alberi, come ad esempio una foresta casuale. Ogni albero decisionale partiziona lo spazio degli ingressi applicando una serie di test su singole feature, ad esempio $x_j \leq \theta$, con $(x_1, \dots, x_d) \in \mathcal{X}$ e $\theta \in \mathbb{R}$. Si Raccogliono tutte le condizioni uniche presenti in tutti gli alberi. Si suppone, quindi, di avere in totale n condizioni, che formano un insieme di proposizioni $\mathcal{P} = \{p_1, \dots, p_n\}$, dove ciascun $p_i := (x_j \leq \theta)$ rappresenta un test booleano sull'ingresso. Ogni $x \in \mathcal{X}$ induce una *valutazione* di queste proposizioni, trasformandole in un vettore binario $\mathbf{x} \in \{0, 1\}^n$.

La logica decisionale dell'ensemble può essere semplicemente descritta da una tabella $\mathcal{S} \subseteq \{0, 1\}^n \times \mathcal{Y}$. Si indica il fatto che $x \in \mathcal{X}$ soddisfa esattamente i test in una certa riga \mathbf{x} con la notazione $x \in \mathbf{x}$, e utilizziamo $\mathcal{S}(\mathbf{x}) = y$ per denotare che $(\mathbf{x}, y) \in \mathcal{S}$. Per ogni classe y , si definisce una *tabella caratteristica* $\mathcal{S}_y \subseteq \{0, 1\}^n \times \{0, 1\}$ tale che, per ogni valutazione \mathbf{x} , se $(\mathbf{x}, y) \in \mathcal{S}$ (rispettivamente, $(\mathbf{x}, y) \notin \mathcal{S}$), allora $(\mathbf{x}, 1) \in \mathcal{S}_y$ (rispettivamente, $(\mathbf{x}, 0) \in \mathcal{S}_y$). In effetti, \mathcal{S}_y è una *tabella di verità* e la si indica con φ_y la corrispondente formula in DNF. Per una classe y , la regola

$$\varphi_y \Leftrightarrow y$$

caratterizza logicamente il comportamento di F rispetto a y .

Di seguito, per una formula generica φ , si utilizzerà la notazione $\mathbf{x} \models \varphi$ per indicare che \mathbf{x} *soddisfa* φ e, poiché per ogni istanza x esiste esattamente una corrispondente \mathbf{x} , verrà scritto $x \models \varphi$ in luogo di $\mathbf{x} \models \varphi$, abusando lievemente della notazione. Inoltre, \mathbf{x} è essa stessa una formula proposizionale (cioè, un

termine), pertanto, quando φ implica (rispettivamente, non implica) \mathbf{x} , si scrive $\varphi \rightarrow \mathbf{x}$ (rispettivamente, $\varphi \not\rightarrow \mathbf{x}$). La formula φ_y caratterizza l'intera logica del classificatore, anche se, per ora, potrebbe risultare troppo complessa per essere interpretata.

4.3.2 Strategie di riduzione della complessità

Si introducono, quindi, tre strategie semplici che riducono la complessità lungo tre diverse dimensioni. Innanzitutto, invece di enumerare tutte le assegnazioni binarie, si campiona una frazione $r \in (0, 1]$ di esse. La distribuzione di campionamento può essere scelta in modo da riflettere la plausibilità relativa di ciascun pattern d'ingresso secondo un dato modello del mondo oppure può essere effettuata casualmente. In secondo luogo, spesso le righe in \mathcal{S}_y risultano contraddittorie rispetto alla conoscenza del dominio, incluse le relazioni logiche tra le proposizioni. Ad esempio, tutte le frasi della forma $(x_j \leq \theta) \Rightarrow (x_j \leq \theta')$, con $\theta < \theta'$, rendono impossibili alcune valutazioni. Si raccolgono tali vincoli in una *teoria* \mathcal{T} e si controlla ogni riga campionata per verificare la sua consistenza con \mathcal{T} . In terzo luogo, non tutte le proposizioni possono essere ugualmente importanti. Alcune condizioni appaiono frequentemente nell'ensemble o influenzano fortemente le predizioni, mentre altre sono usate raramente. Campionando una frazione $c \in (0, 1]$ di variabili proposizionali, ovvero scegliendo un sottoinsieme $\mathcal{P}' \subseteq \mathcal{P}$, ci si focalizza su un numero gestibile di condizioni e si evita di trattare split irrilevanti o ridondanti; oltre che in base all'importanza, le proposizioni possono essere campionate anche casualmente. In ogni caso, è conveniente campionare proposizioni generate da feature differenti x_j per minimizzare la ridondanza. Per lo stesso motivo, i termini campionati dovrebbero differire almeno su una proposizione campionata.

Ogni riga consistente campionata \mathbf{x} da \mathcal{S}_y diventa così un termine della DNF ridotta $\varphi_y^{\text{sample}}$. Si indica la regola risultante con

$$\varphi_y^{\text{sample}} \Leftrightarrow_r^c y.$$

Per costruzione, si ha che $|\varphi_y^{\text{sample}}| < |\varphi_y|$ e in molti casi è possibile forzare la condizione $|\varphi_y^{\text{sample}}| \ll |\varphi_y|$. Qui, per una formula φ , verrà utilizzata $|\varphi|$ per denotare il numero dei suoi token. Nel seguito, per un termine $\mathbf{x} \in \mathcal{S}_y$, si userà $\bar{\mathbf{x}}$ per indicare la sua versione proiettata. Quando $r = c = 1$, φ^{sample} rappresenta

	p_1	p_2	p_3	l		p_1	p_2	p_3	l	
\mathbf{x}_1	0	0	0	1	\rightarrow	\mathbf{x}_1	0	0	0	1
\mathbf{x}_2	0	0	1	0		\mathbf{x}_3	0	1	0	1
\mathbf{x}_3	0	1	0	1						
\mathbf{x}_4	0	1	1	1						
\mathbf{x}_5	1	0	0	0						
\mathbf{x}_6	1	0	1	0						
\mathbf{x}_7	1	1	0	0						
\mathbf{x}_8	1	1	1	1						

$\varphi_y^{\text{sample}}$
 $=$
 $\bar{\mathbf{x}}_1 \vee \bar{\mathbf{x}}_3$
 $=$
 $00 \vee 01$
 \equiv
 $000 \vee 001 \vee 010 \vee 011$

Figura 4.2: Illustrazione del campionamento di righe e colonne per la costruzione di una DNF parziale. La tabella a sinistra rappresenta la tabella di verità originale. Vengono campionate due righe positive (in alto a destra) e successivamente proiettate su due colonne (p_1, p_2), formando la DNF ridotta $\varphi_y^{\text{sample}}$ (in basso).

fedelmente la logica del classificatore; tuttavia, l'uso di valori minori per r o c introduce potenzialmente un errore.

Esempio 4.3.1. Sia \mathcal{S}_y la tabella di verità in Fig. 4.2 (in alto), con $n = 3$, per una certa classe y . In questo esempio, si ha $\mathcal{T} = \{p_1 \rightarrow p_2\}$, pertanto ci sono due termini inconsistenti (le righe \mathbf{x}_5 e \mathbf{x}_6). Dalla tabella \mathcal{S}_y sono state campionate due righe positive (\mathbf{x}_1 e \mathbf{x}_3). Su questi termini campionati sono state prese in considerazione solo due colonne (p_1 e p_2), in modo tale che la DNF risultante contenga due termini parziali ($\bar{\mathbf{x}}_1$ e $\bar{\mathbf{x}}_3$). A seguito del campionamento, delle quattro righe positive originali ($\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4$, e \mathbf{x}_8), due sono state coperte direttamente dal campionamento delle righe (\mathbf{x}_1 e \mathbf{x}_3), una è stata coperta indirettamente grazie al campionamento delle colonne (\mathbf{x}_4), mentre una rimane scoperta (\mathbf{x}_8); inoltre, un termine negativo è stato erroneamente incluso come effetto collaterale del campionamento delle colonne (\mathbf{x}_2).

Poiché il campionamento può far sì che $\varphi_y^{\text{sample}}$ devii da φ_y , seguendo [40], si vuole ora valutare la sua *fedeltà*, misurando gli errori di Tipo I (falsi positivi) e Tipo II (falsi negativi). Per semplificare il trattamento, si assuma che: (i) le istanze in \mathcal{X} siano rappresentate uniformemente dai termini in \mathcal{S}_y , ovvero, $Pr[x \in \mathbf{x}] = Pr[x \in \mathbf{x}']$ per ogni $x, \mathbf{x}, \mathbf{x}'$; (ii) l'assegnazione delle etichette in \mathcal{S}_y si comporti come una variabile di Bernoulli, cioè $Pr[\mathbf{x} \text{ è positivo}] = z$ per ogni riga \mathbf{x} e per una

costante z ; e (iii) la consistenza rispetto a \mathcal{T} in \mathcal{S}_y si comporti come una variabile di Bernoulli, cioè $Pr[\mathbf{x} \text{ è consistente}] = t$ per ogni riga \mathbf{x} e per una costante t . Si osserva che sia z che t possono essere stimati durante il campionamento; Il fine ultimo di ciò è valutare la probabilità di errore in funzione di n , r e c .

Sotto queste ipotesi, la probabilità di un errore di Tipo II corrisponde esattamente alla probabilità che un termine positivo e consistente in \mathcal{S}_y non sia implicato da $\varphi_y^{\text{sample}}$, mentre la probabilità di un errore di Tipo I corrisponde esattamente alla probabilità che un termine negativo e consistente in \mathcal{S}_y sia implicato da $\varphi_y^{\text{sample}}$.

Teorema 5. *Sia F un ensemble di alberi con vocabolario \mathcal{P} di cardinalità n , appreso sullo spazio di ingresso \mathcal{X} con insieme delle etichette \mathcal{Y} , e sia $y \in \mathcal{Y}$. Siano inoltre V_I (rispettivamente, V_{II}) la variabile probabilistica che misura il numero di istanze negative (rispettivamente, positive) non-classificate. Allora, fissati i valori r, c , si ha*

$$\mathbb{E}[V_{II}] = \lceil (1 - r) \cdot 2^n \cdot z \cdot t \rceil - \lfloor r \cdot (2^{c \cdot n} - 1) \cdot z \cdot t \rfloor.$$

e

$$\mathbb{E}[V_I] = \lceil r \cdot 2^n \cdot z \cdot t \cdot (2^{n \cdot (1-c)} - 1) \cdot (1 - z) \cdot t \rceil.$$

Dimostrazione. Sia $x \in \mathcal{X}$. Si consideri la probabilità che un'istanza positiva venga etichettata erroneamente da $\varphi_y^{\text{sample}}$, ovvero,

$$Pr[\text{errore di tipo II}] = Pr[F(x) = y \wedge x \not\models \varphi_y^{\text{sample}}].$$

Dal momento che ogni istanza ha la stessa probabilità di essere rappresentata da un qualsiasi termine, la probabilità di errore sulle istanze può essere modellata come la probabilità di errore sui termini, ossia,

$$Pr[\text{errore di tipo II}] = Pr[(\mathbf{x}, 1) \in \mathcal{S}_y \wedge \varphi_y^{\text{sample}} \not\models \mathbf{x}].$$

La probabilità che un termine sia positivo e consistente è $z \cdot t$, e la probabilità che una riga venga campionata è r . Pertanto, il numero di errori di tipo II è limitato superiormente da $\lceil (1 - r) \cdot 2^n \cdot z \cdot t \rceil$. Tuttavia, alcuni termini positivi non campionati possono essere coperti da altri termini campionati; il numero di tali eventi è limitato inferiormente da $\lfloor r \cdot (2^{c \cdot n} - 1) \cdot z \cdot t \rfloor$. Ne segue che

$$\mathbb{E}[V_{II}] = \lceil (1 - r) \cdot 2^n \cdot z \cdot t \rceil - \lfloor r \cdot (2^{c \cdot n} - 1) \cdot z \cdot t \rfloor.$$

Analogamente, la probabilità che un'istanza negativa venga non-classificata da $\varphi_y^{\text{sample}}$ è data da

$$Pr[\text{errore di tipo I}] = Pr[F(x) \neq y \wedge x \models \varphi_y^{\text{sample}}].$$

Poiché ogni istanza ha la stessa probabilità di essere rappresentata da un qualsiasi termine, la probabilità di errore sulle istanze è modellabile come la probabilità di errore sui termini, ossia,

$$Pr[\text{errore di tipo I}] = Pr[(\mathbf{x}, 0) \in \mathcal{S}_y \wedge \varphi_y^{\text{sample}} \rightarrow \mathbf{x}].$$

La probabilità che un termine sia positivo e consistente è $z \cdot t$ e la probabilità che una riga venga campionata è r ; per ogni termine positivo campionato, esattamente $2^{n \cdot (1-c)} - 1$ termini sono implicati (dato che si è assicurato che i termini campionati differiscano per almeno una proposizione campionata), e la probabilità che ciascuno di questi termini sia negativo e consistente è $(1 - z) \cdot t$. Di conseguenza,

$$\mathbb{E}[V_I] = \lceil r \cdot 2^n \cdot z \cdot t \cdot (2^{n \cdot (1-c)} - 1) \cdot (1 - z) \cdot t \rceil.$$

□

4.3.3 Minimizzazione logica

Poiché $\varphi_y^{\text{sample}}$ potrebbe rimanere di dimensioni eccessive e contenere ridondanze, ci si rivolge ora alla minimizzazione logica per produrre la regola

$$\varphi_y^{\text{min}} \Leftrightarrow_c^r y,$$

ovvero, un antecedente in DNF minimale, logicamente equivalente a $\varphi_y^{\text{sample}}$. Lo standard accettato nella minimizzazione logica è ESPRESSO, che può essere eseguito in modalità *esatta*, configurandosi praticamente come un'implementazione efficiente del metodo originario di Quine. La minimizzazione logica esatta garantisce che l'antecedente non possa essere rappresentato con un numero inferiore di letterali e clausole; tuttavia, le lettere proposizionali vanno interpretate *modulo teoria*. La teoria sottostante \mathcal{T} ha già giocato un ruolo nel controllo di consistenza di ogni termine di $\varphi_y^{\text{sample}}$; tuttavia, anche dopo la minimizzazione, i termini possono rimanere (leggermente) ridondanti, come verrà discusso nella sezione sperimentale.

4.3.4 L'algoritmo LUMEN

Risulta importante ricordare che l'approccio descritto non si basa sulla struttura ad albero di una foresta casuale, a parte la capacità di ottenere un insieme di proposizioni. Per gli ensemble di boosting, è possibile pesare le proposizioni in base al loro contributo all'ensemble, incorporando la logica della maggioranza pesata nelle formule proposizionali. In maniera ancora più ambiziosa, se si riesce a estrarre concetti coerenti e comprensibili dall'uomo da una rete neurale — utilizzando metodi come TCAV [28], Network Dissection [2] o i Concept Bottleneck Models [29] — lo stesso approccio logico può essere applicato, sostituendo semplicemente le proposizioni derivate dagli alberi con proposizioni derivate dai concetti e procediamo come prima.

L'Algoritmo 1 illustra l'approccio trattato, implementato nella funzione `EXTRACTANDMINIMIZE(F, y, r, c)`. L'algoritmo prende in input un ensemble ad alberi F , una classe target y , e due parametri $0 < r, c \leq 1$. In output restituisce una regola in DNF minimale $\varphi_y^{\min} \Leftrightarrow_r^c y$ che spiega come F decide la classe y . In primo luogo, si estrae l'insieme di tutte le proposizioni sulle soglie delle feature \mathcal{P} da F e si costruisce una teoria \mathcal{T} che cattura i vincoli del dominio (ad esempio, l'ordinamento delle soglie). Successivamente, si sceglie un sottoinsieme $\mathcal{P}' \subseteq \mathcal{P}$ contenente una frazione c delle proposizioni originali, eventualmente guidato dall'importanza delle feature. Si campionano quindi fino a $r \cdot 2^{|\mathcal{P}|}$ vettori binari da $\{0, 1\}^{|\mathcal{P}|}$. Ogni vettore \mathbf{x} viene controllato per la consistenza con \mathcal{T} ; se risultasse consistente, si istanzia un punto rappresentativo $x \in \mathbf{x}$ nello spazio d'ingresso e lo si classifica con F . Se $F(x) = y$, si proietta \mathbf{x} sul sottoinsieme \mathcal{P}' , ottenendo $\bar{\mathbf{x}}$, che viene aggiunto al set di campionamento $\mathcal{S}_y^{\text{sample}}$. Al termine, si genera una DNF preliminare $\varphi_y^{\text{sample}}$ a partire da tutte le assegnazioni parziali in $\mathcal{S}_y^{\text{sample}}$ e si applica un algoritmo di minimizzazione logica (ad esempio ESPRESSO) per ottenere la DNF finale minima φ_y^{\min} . Questa formula, combinata con l'etichetta y , fornisce la regola esplicativa $\varphi_y^{\min} \Leftrightarrow_r^c y$.

Algoritmo 1: LUMEN.

Input: F, y, r, c **Output:** Regola $\varphi_y^{\min} \Leftrightarrow_r^c y$ **Function** ExtractAndMinimize(F, y, r, c):

```

 $\mathcal{P} \leftarrow \text{ExtractAlphabet}(F);$ 
 $\mathcal{T} \leftarrow \text{ExtractTheory}(\mathcal{P});$ 
 $\mathcal{P}' \leftarrow \text{RandomSubset}(\mathcal{P}, c);$ 
 $\mathcal{S}_y^{\text{sample}} \leftarrow \emptyset;$ 
for  $s \in [0, \dots, \lceil r \cdot 2^{|\mathcal{P}|} \rceil]$  do
     $i \leftarrow \text{Random}(0, 2^{|\mathcal{P}|});$ 
     $\mathbf{x} \leftarrow \text{BinaryRepresentation}(i, |\mathcal{P}|);$ 
     $fail \leftarrow \text{Close}(\mathbf{x}, \mathcal{T});$ 
    if not ( $fail$ ) then
         $x \leftarrow \text{RepresentativeInstance}(\mathbf{x});$ 
         $l \leftarrow \text{Classify}(F, x, y);$ 
        if  $l = 1$  then
             $\bar{\mathbf{x}} \leftarrow \text{Project}(\mathbf{x}, \mathcal{P}');$ 
             $\mathcal{S}_y^{\text{sample}} \leftarrow \mathcal{S}_y^{\text{sample}} \cup \{\bar{\mathbf{x}}\};$ 
        end
    end
end
 $\varphi_y^{\text{sample}} \leftarrow \text{ProduceDNF}(\mathcal{S}_y^{\text{sample}});$ 
 $\varphi_y^{\min} \leftarrow \text{Minimize}(\varphi_y^{\text{sample}});$ 
return Rule( $\varphi_y^{\min}, y$ );

```

end function

Sperimentazione e Analisi

In questo capitolo, dopo aver fornito un accurato riferimento all'implementazione realizzata, si presenta una valutazione completa di LUMEN in confronto con i metodi più avanzati di estrazione di regole per l'interpretabilità delle random forest e non solo. Si sono condotti esperimenti su dieci dataset ben noti del repository UCI per valutare sia la fedeltà che l'interpretabilità delle regole estratte. Una volta aver descritto le premesse sperimentali, si analizzano i risultati in tre fasi: prima esaminando le prestazioni su cinque dataset iniziali con un focus sulla perfetta fedeltà delle regole, poi investigando le prestazioni su cinque dataset aggiuntivi con foreste più piccole, ma sempre con una accuratezza elevata; e infine esplorando l'effetto del campionamento delle colonne sul compromesso tra interpretabilità e fedeltà. Durante tutto il processo, verrà confrontato LUMEN con InTrees, BATrees, TREPAN, REFNE e RuleCOSI+, valutando la capacità di ciascun metodo di produrre regole che rappresentino fedelmente la random forest originale rimanendo al contempo concise e interpretabili.

5.1 Implementazione

L'implementazione di LUMEN e l'intera sperimentazione presentata in questa tesi sono state realizzate utilizzando il linguaggio di programmazione Julia [3], selezionato per le sue caratteristiche peculiari nel contesto del calcolo scientifico ad alte prestazioni e per la naturale integrazione con paradigmi di programmazione simbolica.

5.1.1 Il pacchetto SolePostHoc

Al fine di supportare la ricerca condotta e garantire un'implementazione sistematica degli algoritmi analizzati, è stato sviluppato **SolePostHoc**¹, un pacchetto software che si inserisce nell'ecosistema del framework Sole (**S**ymb**O**lic **L**earning) [33] visibile in Fig. 5.1. Tale pacchetto fornisce non solamente l'implementazione di LUMEN, ma comprende anche implementazioni standardizzate di tutti gli algoritmi state-of-the-art per l'apprendimento post-hoc considerati in questo lavoro, includendo InTrees, BATrees, TREPAN, REFNE e RuleCOSI+.

Uno delle principali contributi di **SolePostHoc** risiede nell'uniformazione dell'interfaccia algoritmica. Tradizionalmente, i diversi approcci di post-hoc learning producono output in formati eterogenei: alcuni algoritmi (BATrees, TREPAN, REFNE) generano strutture ad albero, mentre altri (InTrees, RuleCOSI+) producono liste decisionali. **SolePostHoc** risolve tale frammentazione attraverso l'introduzione della funzione unificata `ExtractRuleSet()`, la quale sfrutta il sistema di multiple dispatch di Julia per fornire un'interfaccia consistente che produce invariabilmente output standardizzati nel formato `DecisionSet`.

Il paradigma implementativo può essere formalizzato come segue:

$$\text{Approccio tradizionale: } \mathcal{A}_i(\boldsymbol{\theta}) \rightarrow \mathcal{O}_i \text{ (eterogeneo)} \quad (5.1)$$

$$\text{Approccio SolePostHoc: } \text{ExtractRuleSet}(\mathcal{A}_i, \boldsymbol{\theta}) \rightarrow \mathcal{D} \text{ (uniforme)} \quad (5.2)$$

dove \mathcal{A}_i rappresenta l' i -esimo algoritmo, \mathcal{O}_i l'output specifico, $\boldsymbol{\theta}$ i parametri di configurazione e \mathcal{D} il `DecisionSet` standardizzato.

¹Repository disponibile all'indirizzo: <https://github.com/aclai-lab/SolePostHoc.jl>. Il pacchetto è rilasciato sotto licenza open-source per favorire la ricerca collaborativa e contiene l'implementazione completa di LUMEN insieme agli algoritmi di confronto.

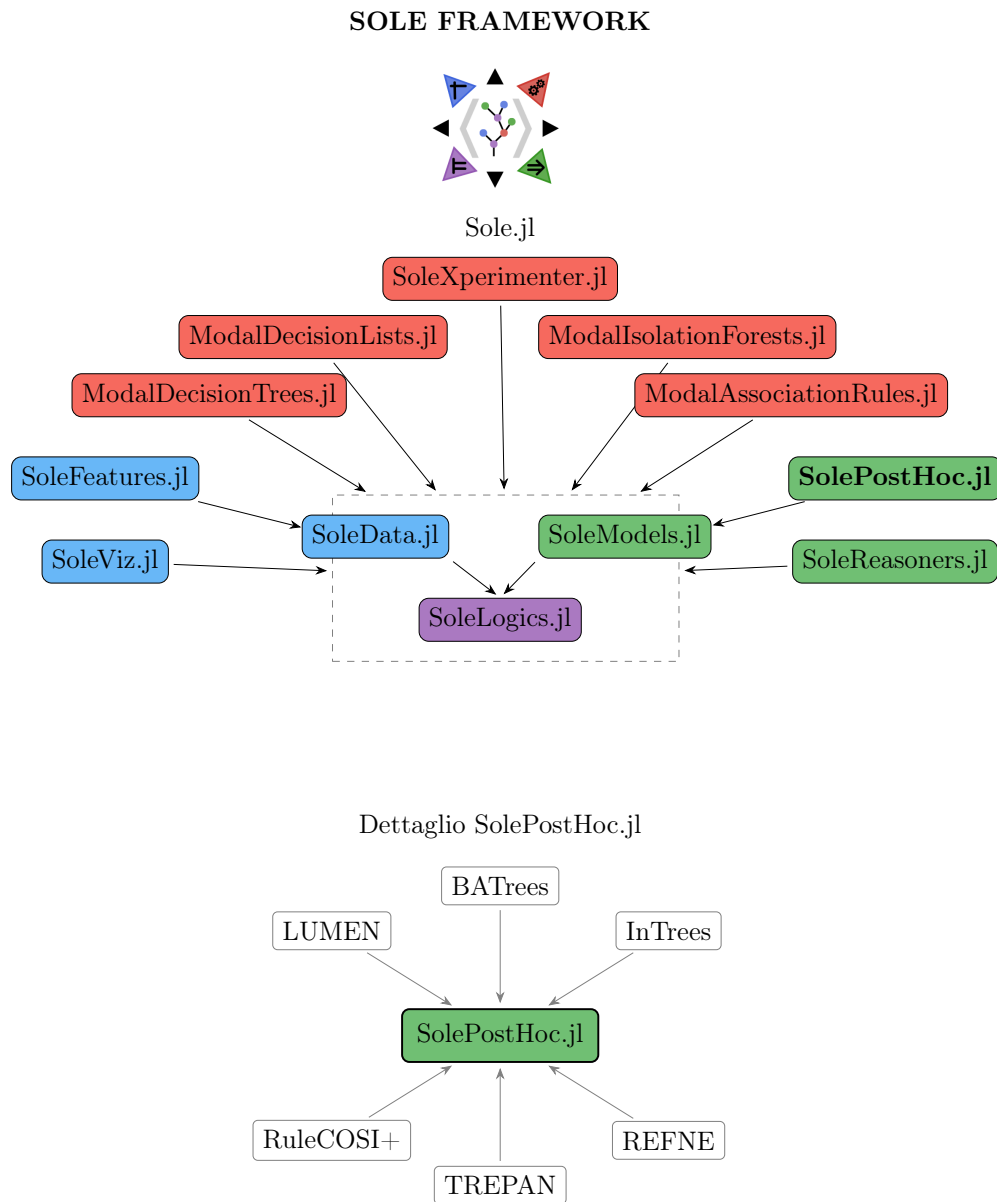


Figura 5.1: In alto struttura completa del framework **Sole** per l'apprendimento simbolico (il pacchetto in viola fornisce strumenti per la manipolazione delle formule logiche; i pacchetti in blu forniscono strumenti per l'elaborazione simbolica dei dati; quelli in rosso forniscono strumenti per l'apprendimento di modelli simbolici; quelli in verde forniscono strumenti per la manipolazione di modelli simbolici). In basso il dettaglio del pacchetto **SolePostHoc**.

5.1.2 Riproducibilità e accessibilità del codice

Per garantire la completa riproducibilità dei risultati sperimentali presentati l'intero codice sorgente, i dataset utilizzati e gli script di sperimentazione sono resi disponibili attraverso un repository pubblico dedicato².

Il repository comprende:

- Documentazione tecnica completa con specifiche API e esempi d'uso
- Script di benchmark che riproducono integralmente tutti gli esperimenti condotti
- Dataset preprocessati in formato standardizzato
- Istruzioni per l'installazione automatizzata dell'ambiente di sviluppo

La struttura del repository è stata progettata per consentire la riproduzione esatta dei risultati attraverso un processo completamente automatizzato, eliminando le barriere tecniche che potrebbero impedire la verifica indipendente dei risultati.

5.1.3 Prospettive di sviluppo e distribuzione

`SolePostHoc` è destinato alla registrazione nell'ecosistema ufficiale dei pacchetti Julia, garantendo accessibilità attraverso il package manager nativo del linguaggio. Tale processo assicurerà:

- Installazione semplificata mediante `Pkg.add("SolePostHoc")`
- Versionamento semantico e compatibilità retroattiva

²Repository degli esperimenti disponibile all'indirizzo: <https://github.com/Perro2110/Un-framework-sistematico-logic-based-per-spiegare-modelli-simbolici-di-ensemble>. Questo repository contiene tutti gli script necessari per riprodurre integralmente gli esperimenti presentati nella tesi, inclusi i dataset preprocessati e le configurazioni utilizzate.

- Integrazione nativa con l’ecosistema del framework Sole, che comprende diversi pacchetti specializzati tra cui `SoleModels`³ e `SoleLogics`⁴
- Supporto continuativo e manutenzione da parte della community

La scelta progettuale di utilizzare Julia si è dimostrata particolarmente vantaggiosa per i requisiti computazionali degli algoritmi di post-hoc learning, combinando l’espressività sintattica tipica dei linguaggi di alto livello con prestazioni comparabili a implementazioni in linguaggi compilati a basso livello.

In sintesi, l’implementazione presentata non si limita a fornire una verifica sperimentale dei risultati teorici, ma costituisce un contributo software duraturo per la comunità scientifica, facilitando future ricerche nel dominio dell’interpretabilità dei modelli di machine learning attraverso strumenti robusti, documentati e immediatamente utilizzabili.

Premesse

Sono stati condotti esperimenti su dieci dataset pubblicamente disponibili dal repository UCI: *Monks-1*, *Monks-2*, *Monks-3*, *Hayes-roth*, *Balance-scale*, *Car*, *Post-operative*, *Urinary-d1*, *Urinary-d2* e *Iris*. Ogni dataset è stato suddiviso in una porzione di training (80% dell’originale) e una porzione di test (20%). Un classificatore random forest è stato quindi addestrato sul set di training (sono stati selezionati solo modelli con accuratezza iniziale $\geq 70\%$), e il modello risultante è stato fornito come input a ciascun algoritmo di estrazione delle regole. Dove possibile, gli iperparametri dei metodi concorrenti sono stati ottimizzati per massimizzare la fedeltà sul set di test.

Seguendo i lavori precedenti sull’interpretabilità globale, la valutazione si è concentrata su sensibilità, specificità, lunghezza media della spiegazione locale minima (cioè, la dimensione della singola congiunzione “attivante” per ogni istanza di test), e lunghezza totale della regola (in DNF, in modo che le spiegazioni locali

³Repository disponibile all’indirizzo: <https://github.com/aclai-lab/SoleModels.jl>. Questo pacchetto fornisce le strutture dati e le primitive fondamentali per la rappresentazione di modelli simbolici nel framework Sole.

⁴Repository disponibile all’indirizzo: <https://github.com/aclai-lab/SoleLogics.jl>. Questo pacchetto implementa le logiche modali e temporali che costituiscono la base teorica del framework Sole, fornendo gli strumenti per la manipolazione di formule logiche complesse.

fossero comparabili tra tutti i metodi). Nei casi in cui il metodo di estrazione delle regole restituiva una lista di decisioni, questa è stata convertita classe per classe in DNF per una misurazione coerente.

5.2 Prima fase

In questa fase si sono analizzate le prestazioni su cinque dataset con un focus sulla perfetta fedeltà delle regole, i risultati di questa fase sono visualizzabili nella Tabella 5.1 che riassume i risultati su *Monks-1*, *Monks-2*, *Monks-3*, *Hayes-roth* e *Balance-scale*, ciascuno con una foresta di 10 alberi (ogni foresta con accuratezza, rispetto alla partizione di test, \geq del 70%). Per tutti i dataset, LUMEN è stato eseguito con $r = c = 1$, il che significa che non è stato effettuato alcun campionamento di righe o colonne. LUMEN ha costantemente raggiunto una fedeltà perfetta in tutte le classi, producendo regole che corrispondevano esattamente alla random forest su ogni istanza di test. Gli altri metodi tipicamente presentavano o una sensibilità inferiore, o una specificità inferiore, o set di regole significativamente più lunghi. Su *Monks-1*, solo BATrees ha anche raggiunto una fedeltà perfetta in entrambe le classi ma ha prodotto regole approssimativamente da due a tre volte più lunghe di quelle di LUMEN e ha generato spiegazioni locali quasi due volte più lunghe. Per *Monks-2*, nessuno degli approcci standard ha raggiunto una fedeltà perfetta, producendo spesso regole da due a cinque volte più lunghe di quelle di LUMEN. In *Monks-3*, la maggior parte dei concorrenti si è avvicinata anche alla fedeltà perfetta, ma ha comunque prodotto regole leggermente più lunghe di quelle di LUMEN, tranne in un caso di BATrees che ha abbassato la sensibilità complessiva in una classe. Il dataset *Hayes-roth* si è rivelato più impegnativo: nessun concorrente ha raggiunto la fedeltà 1. Alcuni metodi hanno prodotto regole di dimensioni moderate ma con sensibilità bassa fino a 0.50 per una delle classi; altri non sono riusciti a estrarre regole per alcune classi. Infine, su *Balance-scale*, dove la classe 3 non è mai stata predetta dalla foresta addestrata, LUMEN ha riconosciuto questa situazione non generando una regola per la classe 3 (coerentemente con il comportamento della foresta), mentre RuleCOSI+ ha cercato di adattare una regola non necessaria o inaffidabile. In questi cinque dataset, LUMEN ha costantemente restituito regole perfettamente fedeli ma più corte o di dimensioni comparabili a quelle dei metodi concorrenti.

	Metric	Monks-1		Monks-2		Monks-3		Hayes-roth			Balance-scale		
Class		1	2	1	2	1	2	1	2	3	1	2	3
InTrees	Sens	0.82	0.91	0.88	0.85	1.00	0.98	0.58	1.00	0.53	0.90	0.90	-
	Spec	0.91	0.82	0.85	0.88	0.98	1.00	0.66	0.76	1.00	0.90	0.90	-
	Min	3.60	4.87	7.46	5.04	4.62	3.78	8.75	4.50	6.85	5.09	4.46	-
	Len	>10 ³	784	>10 ³	>10 ³	603	525	>10 ³	55	>10 ³	>10 ³	>10 ³	-
BATrees	Sens	1.00	1.00	0.41	1.00	1.00	1.00	0.66	1.00	1.00	0.89	1.00	-
	Spec	1.00	1.00	1.00	0.41	1.00	1.00	1.00	0.92	0.85	1.00	0.89	-
	Min	5.04	4.70	6.57	2.16	2.00	1.37	5.50	5.00	4.33	5.60	4.43	-
	Len	133	78	32	76	2	3	22	52	36	91	96	-
TREPAN	Sens	0.94	0.83	0.64	0.94	1.00	1.00	0.75	1.00	1.00	0.84	0.88	-
	Spec	0.83	0.94	0.94	0.64	1.00	1.00	1.00	0.88	1.00	0.88	0.84	-
	Min	6.35	7.75	7.64	6.13	3.00	1.74	6.66	6.00	5.32	7.41	5.41	-
	Len	430	427	189	318	5	4	50	25	39	359	331	-
REFNE	Sens	0.94	0.91	0.41	0.93	1.00	1.00	0.50	1.00	0.76	0.96	0.58	-
	Spec	0.91	0.94	0.93	0.41	1.00	1.00	0.80	0.92	0.71	0.58	0.96	-
	Min	5.07	4.98	6.00	3.44	3.70	2.01	4.00	3.00	3.42	2.11	2.18	-
	Len	145	116	69	79	11	12	4	10	7	5	7	-
RuleCOSI+	Sens	1.00	0.83	0.58	0.84	1.00	1.00	-	0.50	0.92	0.87	0.25	0.00
	Spec	0.83	1.00	0.84	0.58	1.00	1.00	-	0.56	0.78	0.41	1.00	0.85
	Min	4.95	5.12	5.00	4.06	3.00	2.62	-	4.75	3.53	1.19	3.00	4.00
	Len	>10 ³	148	208	736	18	14	-	156	36	3	6	4
LUMEN	Sens	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-
	Spec	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-
	Min	2.57	3.25	4.29	1.38	2.00	1.00	4.33	2.0	2.61	2.36	2.30	-
	Len	42	44	46	22	4	2	88	24	65	62	51	-

Tabella 5.1: Prestazioni sui cinque dataset: *Monks-1*, *Monks-2*, *Monks-3*, *Hayes-roth*, e *Balance-scale*. Le colonne mostrano sensibilità (*Sens*), specificità (*Spec*), lunghezza media della spiegazione locale minima (*Min*), e lunghezza totale della regola (*Len*).

5.3 Seconda fase

In questa fase si sono analizzate le prestazioni su cinque dataset con un focus sulla perfetta fedeltà delle regole, i risultati di questa fase sono visualizzabili attraverso la Tabella 5.2 la quale riporta in dettaglio le prestazioni su *Car*, *Post-operative*, *Urinary-d1*, *Urinary-d2* e *Iris*. Le random forest qui contenevano meno alberi (sette per *Car* e tre per gli altri dataset), permettendo uno studio più approfondito di come i metodi di estrazione si comportano con foreste più semplici con un'accuratezza di circa il 70%. LUMEN ha nuovamente raggiunto una fedeltà perfetta in tutte le classi. Altri metodi occasionalmente hanno fatto lo stesso per alcune classi, ma tendevano a produrre regole complessivamente più lunghe o mostravano

	Metric	Car		Post-operative		Urinary-d1		Urinary-d2		Iris		
Class		1	2	1	2	1	2	1	2	1	2	3
InTrees	Sens	0.77	1.00	1.00	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00
	Spec	1.00	0.77	1.00	1.00	1.00	0.92	1.00	1.00	1.00	1.00	1.00
	Min	2.15	6.00	4.70	5.00	3.33	2.41	2.44	1.60	2.00	4.22	1.00
	Len	107	72	> 10 ³	444	22	30	8	5	2	59	1
BATrees	Sens	1.00	1.00	0.94	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00
	Spec	1.00	1.00	0.00	0.94	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	Min	1.81	4.00	2.00	2.00	0.00	2.00	2.00	2.00	3.00	3.00	2.00
	Len	10	4	4	4	4	10	4	4	9	9	2
TREPAN	Sens	0.97	0.71	0.88	0.00	1.00	1.00	1.00	1.00	0.77	0.77	1.00
	Spec	0.71	0.97	0.00	0.88	1.00	1.00	1.00	1.00	0.90	0.90	1.00
	Min	3.65	9.13	4.25	3.50	4.30	2.36	2.11	2.53	5.00	3.66	1.00
	Len	529	437	55	38	25	19	14	13	29	20	1
REFNE	Sens	0.98	0.41	1.00	1.00	0.61	0.90	0.66	0.80	0.55	1.00	1.00
	Spec	0.41	0.98	1.00	1.00	0.90	0.61	0.80	0.66	1.00	0.80	1.00
	Min	1.49	6.12	4.41	7.00	4.33	5.06	5.00	4.80	3.00	3.00	2.00
	Len	40	39	294	255	140	78	155	99	6	6	4
RuleCOSI+	Sens	0.39	1.00	1.00	-	0.92	1.00	0.77	0.80	1.00	-	1.00
	Spec	1.00	0.39	0.00	-	1.00	0.92	0.80	0.77	0.57	-	1.00
	Min	1.00	1.00	2.16	-	4.33	1.66	2.00	1.00	1.00	-	1.00
	Len	1	1	18	-	60	4	2	3	1	-	1
LUMEN	Sens	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Spec	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Min	1.00	4.00	2.64	4.00	1.61	2.00	2.00	2.00	2.00	2.00	1.00
	Len	4	16	25	30	7	6	4	4	7	21	1

Tabella 5.2: Prestazioni sui cinque dataset: *Car*, *Post-operative*, *Urinary-d1*, *Urinary-d2*, e *Iris*. Le colonne mostrano sensibilità (*Sens*), specificità (*Spec*), lunghezza media della spiegazione locale minima (*Min*), e lunghezza totale della regola (*Len*).

una copertura sbilanciata.

Su *Car*, BATrees ha eguagliato la perfetta fedeltà di LUMEN ma ha prodotto spiegazioni locali che erano in media più lunghe. Alcuni metodi hanno approssimato la foresta con regole più corte o di uguale lunghezza in alcune classi, ma hanno sacrificato precisione o praticità nell'interpretazione locale. In *Post-operative*, InTrees e REFNE hanno raggiunto una fedeltà molto alta ma, in molti casi, hanno generato regole molto più lunghe di quelle di LUMEN. Altri metodi non sono riusciti a estrarre regole soddisfacenti per una classe o hanno restituito regole triviali e sbilanciate. Per *Urinary-d1*, solo TREPAN tra i concorrenti ha raggiunto una fedeltà perfetta, con regole tre volte più lunghe di quelle di LUMEN; BATrees ha restituito regole sbilanciate e inaffidabili che predicevano sempre una sola classe.

Un modello simile è emerso per *Urinary-d2*, dove la maggior parte degli approcci aveva o regole di lunghezza moderata con fedeltà inferiore o era completamente sbilanciata, ma BATrees ha ottenuto prestazioni comparabilmente buone in termini di dimensioni e fedeltà. Su *Iris*, alcuni metodi hanno estratto regole brevi per le classi 1 e 2, ma le spiegazioni locali erano più lunghe di quelle di LUMEN, e nessun metodo ha eguagliato la simultanea minimalità e fedeltà totale di LUMEN in tutte e tre le classi.

5.4 Terza fase

In questa terza fase si sono svolti una serie finale di test, scegliendo di far variare solo il parametro di campionamento delle colonne c da 0.2 a 1.0 (con incrementi di 0.2) mantenendo $r = 1$. Le colonne (cioè, le variabili proposizionali) vengono selezionate in base all'importanza delle caratteristiche estrapolate dai modelli in input attraverso l'approccio noto come MDI (Mean Decrease in Impurity) [42]. La Figura 5.2 illustra come la regolazione di c influenzi la specificità dei set di regole e la lunghezza media della spiegazione locale, aggregata su tutti i dataset con una foresta di 10 alberi. Ridurre c generalmente accorcia le regole estratte ma aumenta leggermente la possibilità di errate classificazioni. Il compromesso tra fedeltà e interpretabilità risulta particolarmente favorevole per $c = 0.6$, dove LUMEN supera o eguaglia ancora la fedeltà di altri approcci ma mantiene spiegazioni locali più brevi.

5.5 Considerazioni conclusive

Questi esperimenti mostrano che LUMEN o eguaglia o supera la fedeltà degli algoritmi di spiegazione globale esistenti fornendo anche regole che sono spesso più brevi e più facili da interpretare sia a livello globale (intera formula) che locale (singola congiunzione). I risultati evidenziano come ciascun algoritmo alternativo fatica a replicare perfettamente la logica della foresta, producendo spesso o una precisione inferiore o formule DNF significativamente più lunghe. Questo risultato dimostra che la maggior parte degli approcci non mira esplicitamente a una ricostruzione *perfettamente fedele* e si concentra invece su spiegazioni post-hoc

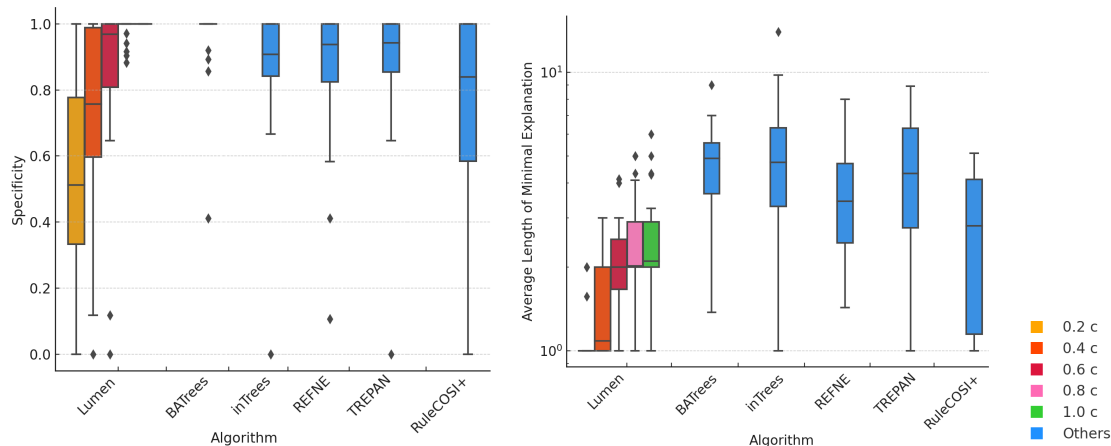


Figura 5.2: Effetto della variazione del parametro di campionamento delle colonne c sulla specificità (grafico a sinistra) e sulla lunghezza media della spiegazione locale minima (grafico a destra), aggregato su tutti i dataset con una foresta di 10 alberi. Un c più basso riduce la complessità delle regole ma può aumentare leggermente il rischio di errata classificazione. Intorno a $c \in [0.6, 0.8]$, LUMEN raggiunge un compromesso vantaggioso tra fedeltà e interpretabilità.

approssimative. Al contrario, il framework di campionamento e minimizzazione logica di LUMEN garantisce un'equivalenza esatta quando $r = c = 1$, e un equilibrio flessibile tra complessità e fedeltà altrimenti. Sebbene il tempo di esecuzione di LUMEN cresca notevolmente con la dimensione della foresta, gli esperimenti fino a dieci alberi per foresta sono stati completati in meno di un minuto su un sistema dual Intel Xeon con 224 GB di RAM. Questo livello di prestazioni è sufficiente per molti scenari del mondo reale in cui l'interpretabilità, l'affidabilità e l'attendibilità delle regole sono fondamentali.

Conclusioni e sviluppi futuri

In questa tesi è stato presentato **LUMEN**, un framework innovativo basato sulla logica formale per l'estrazione di spiegazioni globali da classificatori black-box, con particolare attenzione alle random forest. A differenza di numerosi approcci precedenti, che si affidano principalmente a tecniche di potatura approssimative o surrogati guidati dai dati, LUMEN adotta un approccio rigoroso, esente da euristiche, che traduce direttamente la struttura decisionale di ogni ensemble in logica proposizionale, applicando successivamente sofisticate tecniche di minimizzazione logica per garantire una rappresentazione al contempo compatta e fedele al modello originale.

I risultati sperimentali ottenuti dimostrano che LUMEN raggiunge costantemente una fedeltà perfetta rispetto al modello originale, un obiettivo dove i metodi concorrenti spesso mostrano limitazioni significative. È particolarmente degno di nota come il framework riesca a fornire regole frequentemente più brevi o ugualmente concise sia a livello globale che locale, migliorando così l'interpretabilità delle spiegazioni generate.

Un aspetto fondamentale di LUMEN è la sua intrinseca parallelizzabilità, caratteristica che ne aumenta notevolmente la potenziale efficienza computazionale e ne facilita l'applicazione a problemi di dimensioni considerevoli. Questa proprietà

permette di compensare la maggiore complessità algoritmica rispetto ad approcci più semplificati, consentendo elaborazioni distribuite su sistemi multi-core o cluster computazionali.

Altrettanto rilevante è la natura modulare del framework, che rappresenta uno dei suoi punti di forza principali. LUMEN è stato progettato seguendo un'architettura a componenti indipendenti, prevedendo di permettere agli utenti di “attaccare e staccare” nella pipeline del framework diversi moduli funzionali, quali:

- Minimizzatori logici alternativi, adattabili a diverse esigenze di precisione e velocità
- Metodi differenziati per l'estrazione di righe e colonne significative
- Strategie di campionamento personalizzabili
- Tecniche di traduzione in formule logiche ottimizzate per specifici tipi di ensemble

Sebbene inizialmente progettato per le random forest, l'approccio si adatta naturalmente ai metodi di boosting (attraverso l'introduzione di proposizioni pesate) e può essere esteso alle reti neurali mediante tecniche di interpretabilità basate su concetti, offrendo così una visione unificata sia per modelli simbolici che subsimbolici. LUMEN è completamente open-source, con esperimenti facilmente riproducibili e confronti sistematici con molteplici baseline di riferimento nel campo.

Il lavoro futuro esplorerà strategie di campionamento più sofisticate per accelerare ulteriormente l'estrazione delle regole e ridurre la complessità computazionale, includendo tecniche di apprendimento attivo per la selezione ottimale delle righe e algoritmi avanzati di ragionamento su ordini parziali per eliminare termini ridondanti in presenza di vincoli di dominio. Inoltre, l'ottimizzazione della parallelizzazione e l'implementazione di un controllo più granulare dei meccanismi di approssimazione potranno migliorare significativamente la scalabilità del framework su forest più grandi o dati ad alta dimensionalità.

Si prevede che questi miglioramenti consolideranno ulteriormente la posizione di LUMEN come strumento rigoroso, flessibile e affidabile per l'apprendimento automatico interpretabile, applicabile con successo in diversi domini applicativi

dove la comprensibilità dei modelli predittivi risulta essenziale, dalla diagnostica medica all'analisi finanziaria, dai sistemi di raccomandazione alla previsione di rischi in ambito industriale.

Ringraziamenti

Nonostante la tesi si basi su un framework che ambisce all'esattezza, distaccandosi da metodi euristici e probabilistici, devo ammettere di essere stato estremamente fortunato e ritengo giunto il momento di ringraziare tutti coloro che hanno reso possibile questo percorso.

Inizio ringraziando di cuore la donna che ha sempre creduto in me e che mi è sempre stata a fianco, sbattendo la testa contro il muro pur di non rinunciare. È sicuramente grazie a te, che non hai mai smesso di credere in me, se oggi sono qui. Il mio metodo di studio, il mio modo di affrontare le innumerevoli sfide che mi si sono presentate dinanzi in questo percorso di studi, lo devo a te, mamma, che fin da piccolino non hai rinunciato, che mi hai insegnato che sbattendo la testa più e più volte, non rinunciando mai, magari in qualche modo mio, diverso, qualche tabellina l'avrei imparata. E oggi sono qui, non so neanche io come, con una laurea in una materia STEM, in una disciplina che basa le sue radici nella matematica. Chi l'avrebbe mai detto che il ragazzino che ti faceva penare tanto alle elementari e alle medie forse ce l'avrebbe fatta... Grazie mamma di non aver mai rinunciato a me.

E come potrei non ringraziare il mio papà, che prima ancora che un padre è per me un migliore amico, compagno di film Marvel, di mangiate epocali da Giorgione, di giornate e giornate a ripetere materie d'esame. Papà, in qualunque momento di bisogno ci sei sempre per me; quello che sono lo devo a te, che sei e rimarrai per sempre l'esempio di ciò che voglio rappresentare nel mio futuro come padre e

come uomo.

A tutta la mia enorme famiglia, Snoopy, Noemi, Sonia, Mamma e Papà: LUMEN è dedicato a voi che siete stati e siete la mia luce.

Un ringraziamento enorme va dato anche alla mia maestra delle elementari Simonetta, che oltre ad essere stata una maestra per me è stata molto di più, che probabilmente ha gettato le basi accademiche che oggi posso dire essere fiorite in tutto questo. Simo, se so fare ciò che so fare, se ho sviluppato ciò su cui spero si fonderà tutta la mia carriera, la logica, è soprattutto merito tuo; quindi questa tesi non posso che dedicarla anche a te, sapendo che non è neanche la metà di ciò che mi hai donato.

Ai miei zii, Zio Toti, Zio Max, Zio Robby e Zio Tony: grazie per essere parte della mia vita.

Ai miei nonni, a nonna Rossa, che sono sicuro citerà questo momento alla signora Maione ed a sua figlia Rosa. nonna, grazie per essere sempre stata una figura così importante per me; le nostre partite a scopa non le dimenticherò mai. E a nonna Pina, mi spiace che tu non abbia potuto vedere questo momento, mi spiace per molte cose, ma spero che da lassù tu sia fiera di me. Dentro di me porterò sempre il tuo affetto incondizionato e i ricordi bellissimi come quei 20 euro di panzarotti e zeppolelle presi per disperazione.

Ai miei due nonni, nonno Gianni e nonno Carmine: sono sicuro che ci saremmo fatti una bella mangiata per festeggiare. Sono fiero di essere vostro nipote e mi sarebbe piaciuto passare più tempo con voi. Mangerò sempre anche per voi.

Alla mia ragazza Lidia Negri, amore: solo tu puoi sapere quanto mi hai supportato e sopportato, quando probabilmente per mesi, forse per tutto l'anno, il mio chiodo fisso era LUMEN, i minimizzatori, la programmazione. Tu sei l'unica persona che in un modo o nell'altro riusciva a farmi staccare la testa, a pensare ad altro, a farmi stare bene. Lidia, non so come esprimerti il mio amore perché dopo tutti questi anni ancora mi supporti e mi ami. Grazie di avermi regalato momenti unici e spero che siano solo l'inizio di ciò che vivremo insieme.

Un ringraziamento più che speciale ai ragazzi dell'ennesimo gruppo, il gruppo $n+1$ all'interno dell'hotel infinito dei gruppi. Il gruppo di cui avevi bisogno nonostante l'infinito numero di gruppi di cui già facevi parte prima di essere inserito in questo gruppo. Siete tutti speciali e parte fondamentale di questo incredibile percorso, grazie ragazzi.

Al futuro Ing. Adriano e alla futura Ing. Federica, ai futuri dottori Cecilia e Fabrizio: grazie di cuore per tutte le esperienze incredibili, e a tutti coloro che non riesco a citare per via della grandezza immensa del nostro gruppo che ha generato una vera e propria famiglia. Non posso però non citare la dottoressa Alice Zalambani con i suoi tormentoni che da bravo compagno di banco/corso mi sono sempre sorbito, colei che si è rivelata una persona davvero speciale per me, così come i dottori Lorenzo Negrini (sappiamo entrambi che sei il futuro Mark Zuckerberg), Davide Caravita (fa strano non chiamarti Kiogre), e grazie anche a te (purtroppo Ing.) Paolo Buso, compagni di questo incredibile viaggio; Grazie.

Non posso poi non ringraziare ciò che è diventata per me come una seconda casa: tutti i ragazzi di ACLAI e ACLAI+. Se è stato possibile tutto questo, lo devo anche a voi. Impossibile non citare le incredibili SOLE-reunion di colui che è il mio esempio da seguire come programmatore, Giovanni (giuro, prima o poi scriverò dei test). Grazie a tutti, grazie al mio compagno di trasferta Edoardo, grazie all'incredibile energia che mi ha sempre trasmesso Mauro e ai consigli sempre ponderati e corretti, all'istruzione incredibile che mi hai dato tu, Michi. Grazie ad Albe e a Fede per essere dei punti di riferimento a cui è sempre possibile chiedere qualunque cosa e grazie a te Ila (giuro che risponderò a tutte le tue mail da oggi in poi); grazie per la estrema pazienza e per le conoscenze che tutti voi mi state trasmettendo. Spero un giorno di essere all'altezza di tutto ciò che mi avete donato.

Infine, non posso non citare colui grazie al quale tutto ciò è stato possibile. Non è solo un professore per me ma è un mentore ed un esempio da seguire. Spero un giorno di poter essere al Suo livello, Professore. Grazie Guido per avermi dato fiducia, per aver riposto estrema fiducia in me. Spero di poterti ripagare un giorno di tutto questo. Grazie di essere per me un esempio da seguire.

Grazie a tutti coloro che ho e non ho potuto citare (Ing. Moriconi, se so studiare le funzioni è forse e soprattutto merito tuo, Professor D'Andria, Professor Piva, e molti altri...), ma potrei quasi aver scritto di più nei ringraziamenti che nel resto della tesi. GRAZIE A TUTTI... In questo momento così importante non posso che riflettere su quanto ogni singola persona tra voi mi abbia regalato e insegnato. Questa tesi è soprattutto merito vostro.

Bibliografia

- [1] Agata Ciabattoni Andrea Asperti. *Logica a Informatica*. McGraw-Hill Education, 2021.
- [2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, 2017.
- [3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.
- [4] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.
- [5] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. 1984.
- [6] L. Breiman and N. Shang. Born again trees. Unpublished. Available at <https://www.stat.berkeley.edu/~breiman/>, 1996.
- [7] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

- [8] N.H. Bshouty, J.C. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, 2004.
- [9] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541, 2006.
- [10] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016.
- [11] Francesca Ciarullo. *Dispense di Elementi di logica matematica*. Università degli Studi di Palermo, Facoltà di Scienze MM. FF. NN., 2008. Per i corsi di laurea triennale in Matematica per l’Informatica e la Comunicazione Scientifica e Matematica.
- [12] Roberto Confalonieri, Tillman Weyde, Tarek R. Besold, and Fermín Moscoso del Prado Martín. Trepan reloaded: A knowledge-driven approach to explaining black-box models. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, pages 2457–2464, 2020.
- [13] Olivier Coudert and Jean Christophe Madre. New ideas for solving covering problems. 1995.
- [14] M. Craven and J. Shavlik. Extracting tree-structured representations of trained networks. In *Proc of the 8th Annual Conference on Advances in Neural Information Processing Systems*, volume 8, pages 25 – 30, 1995.
- [15] Artur d’Avila Garcez, Marco Gori, Pascal Hitzler, Claude Sierra, Giovanni Tonda, and Tarek R. Besold. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint*, 2019.
- [16] K. Dedja, F.K. Nakano, K. Pliakos, and C. Vens. BELLATREX: building explanations through a locally accurate rule extractor. *IEEE Access*, 11:41348–41367, 2023.

-
- [17] Houtao Deng. Interpreting tree ensembles with inTrees. *International Journal on Data Science Analysis*, 7(4):277–287, 2019.
 - [18] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Proc. of the 1st International Workshop on Multiple Classifier Systems (MCS)*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15, 2000.
 - [19] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
 - [20] J. Gou, B. Yu, S.J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.
 - [21] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):93:1–93:42, 2019.
 - [22] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.
 - [23] J. Hlavicka and P. Fiser. Boom-a heuristic boolean minimizer. In *Proc. of the IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 439–442, 2001.
 - [24] Yacine Izza and João Marques-Silva. On explaining random forests with SAT. In *Proc. of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2584–2591, 2021.
 - [25] J.C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.
 - [26] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proc. of the 30th Annual Conference on Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

-
- [27] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. 1994.
- [28] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proc. of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682, 2018.
- [29] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proc. of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348, 2020.
- [30] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. of the 30th Annual Conference on Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.
- [31] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.
- [32] Josue Obregon and Jae-Yoon Jung. Rulecosi+: Rule extraction for interpreting classification tree ensembles. *Information Fusion*, 89:355–381, 2023.
- [33] Giovanni Pagliarini. Modal symbolic learning: from theory to practice. 2024. PhD thesis.
- [34] Giovanni Pagliarini, Andrea Paradiso, Marco Perrotta, and Guido Sciavicco. Minimal rules from decision forests: a systematic approach. In *OVERLAY 2024, 6th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, Bolzano, Italy, November 2024. Conference Long Abstract.
- [35] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

-
- [36] Willard V Quine. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531, 1952.
- [37] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. 2016.
- [38] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144, 2016.
- [39] Ronald L Rivest. Learning decision lists. *Machine learning*, 2:229–246, 1987.
- [40] M. Robnik-Sikonja and M. Bohanec. Perturbation-based explanations of prediction models. In *Human and Machine Learning - Visible, Explainable, Trustworthy and Transparent*, pages 159–175. 2018.
- [41] R.L. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued minimization for pla optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(5):727–750, 1987.
- [42] Erwan Scornet. Trees, forests, and impurity-based variable importance, 2021.
- [43] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001.
- [44] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [45] Thibaut Vidal and Maximilian Schiffer. Born-again tree ensembles. In *Proc. of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 9743–9753, 2020.
- [46] Z-H. Zhou, Y. Jiang, and S. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Communications*, 16:3–15, 2003.