ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

AI simbolica e sub-simbolica per XAI: stato dell'arte ed esperimenti con reti neurali e vincoli logici

Tesi di laurea in
Sistemi Autonomi

Relatore
Prof. Andrea Omicini

Correlatore
Dott. Roberta Calegari
Dott. Giovanni Ciatto

Candidato
Giuseppe Pisano

IV Appello di Laurea
Anno Accademico 2018-2019

Abstract

L'intelligenza artificiale ha visto nel tempo il delinearsi di due paradigmi distinti: quello simbolico, basato sulla manipolazione di simboli come metodo di approssimazione dell'intelligenza umana, e quello sub-simbolico, basato invece sull'applicazione di procedure statistiche o numeriche. Sebbene il secondo goda oggi di un rinnovato successo – anche grazie ai vantaggi dal punto di vista di scalabilità e capacità di gestione della conoscenza contestuale –, esso manca di uno dei principali pregi delle tecniche simboliche: la *comprensibilità*. Le tecniche sub-simboliche infatti producono spesso predittori difficilmente comprensibili ad un osservatore umano, rendendo le decisioni basate su essi difficilmente interpretabili. D'altra parte, i modelli simbolici – che viceversa sono facilmente comprensibili – non sono arrivati finora ad un'ampia diffusione, presentando limiti sia in termini di performance che in termini di capacità di apprendere. Muovendo da queste considerazioni, è nato un nuovo campo di ricerca, che mira ad unificare e sfruttare in maniera sinergica il paradigma simbolico e sub-simbolico. Le tecniche ibride – che combinano cioè i due approcci a livello di modello – potrebbero fornire la chiave per il superamento dei limiti di entrambi, sfruttandone al contempo i pregi, a particolare beneficio della spiegabilità dei sistemi intelligenti. Tra gli ambiti potenzialmente intersecati con questo nuovo campo di ricerca c'è il ramo della Computazione Neuro Simbolica (NSC), che corrisponde all'oggetto di analisi ed esplorazione di questa tesi. Il primo obiettivo della tesi è fornire una rassegna della letteratura in ambito NSC, finalizzata a valutarne l'impiego nella creazione di sistemi intelligenti *spiegabili*. Come secondo obiettivo, la tesi si occupa di discutere un possibile modello di integrazione, e della sua prototipazione e validazione, selezionando le tecnologie più adatte alla sua realizzazione nel campo della Computazione Neuro Simbolica.

You can't even understand machines motivation...

Ringraziamenti

Se hai trovato un minuto per cercare e leggere questa pagina sicuramente meriti i miei ringraziamenti.

Grazie:)

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

L'intelligenza artificiale (AI) [23][28] è un'ampia area dell'informatica mirata alla realizzazione di *macchine intelligenti*, capaci cioè di portare a termine dei compiti che tipicamente richiederebbero l'intelligenza umana. Le tecniche utilizzate in tale ambito sono diverse, ma principalmente possono essere racchiuse in due macro approcci: simbolico [33] e connessionista [20]. Il primo prevede la rappresentazione e la manipolazione delle informazioni tramite simboli. L'elaborazione avviene quindi su un piano logico, utilizzando simboli, connessioni astratte e relazioni logiche per creare nuove informazioni. Per tale motivo questo approccio viene definito *top-down* [32]: la conoscenza arriva dall'alto, da un piano puramente logico, prescindendo dai valori derivanti dall'esperienza. Il secondo approccio invece, definito *bottom-up*, è completamente opposto. Gli stimoli sono alla base dell'apprendimento, senza il bisogno di nessun modello mentale pregresso. La conoscenza è codificata tramite delle piccole unità funzionali, i neuroni, che attraverso un processo di apprendimento si organizzano in strutture sempre più complesse. La chiave è nella modalità di apprendimento: questa è basata completamente sull'esperienza. A partire da dei dati empirici è quindi possibile addestrare tali reti neuronali a riprodurne il comportamento.

Gli approcci logici, appartenenti alla corrente più classica dell'AI, hanno mostrato diverse lacune legate soprattutto all'estrazione dei dati a partire dai dati di contesto e alle scarse performance nella gestione di teoria ad alta complessità. Gli approcci sub-simbolici, al contrario, si dimostrano estremamente validi dal punto di vista dell'estrazione della conoscenza e delle performance, ma portano ad un altro grave problema: l'opacità dei sistemi risultanti. Infatti la codifica delle informazioni usata da tali tecniche non permette che la conoscenza approssimata venga esplorata e compresa dagli utilizzatori. Questo pone dei grossi limiti alla loro diffusione, non essendo al momento possibile verificarne con certezza il corretto comportamento. Tale problema ha portato alla nascita di tecniche apposite che mirano ad una *ExplainableAI* (XAI) [1], cioè dei sistemi che siano comprensibili

e verificabili da parte di esperti umani.

Queste due correnti insieme contengono tutti gli ingredienti per arrivare a una nuova classe di sistemi dove, grazie al funzionamento sinergico di tecniche simboliche e sub-simboliche, si arrivi ad un superamento dei limiti da loro mostrati se prese singolarmente. Tale idea ha ispirato la nascita di una nuova corrente, che mira all'unificazione di questi due approcci all'apparenza inconciliabili. Anche in questo caso sono state varie le metodologie adottate ma i risultati più promettenti arrivano dall'area di studio della *Computazione-Neuro-Simbolica* (*NSC*) [8].

Il lavoro presentato si pone al centro di queste problematiche. Partendo da una esplorazione della letteratura attuale, si procede a presentare un possibile sistema basato su tecniche di NSC mirato alla spiegabilità delle tecniche connessioniste. Nello specifico, nel capitolo 2 si offre un'introduzione al tema della computazione neuro simbolica e a quello della spiegabilità nel campo dell'intelligenza artificiale. Nel capitolo 3 i temi sopra trattati si uniscono in un unico modello concettuale mirato alla risoluzione del problema della spiegabilità in ottica NSC. Nel dettaglio, a partire dai contributi noti in letteratura sulla creazione di un sistema ibrido votato alla spiegabilità, si procede a presentarne un loro possibile raffinamento. Questo modello si basa su due strumenti fondamentali offerti dalle tecniche di computazione neuro-simbolica visti sopra: il condizionamento e l'induzione logica. In pratica grazie alla prima si vuole fare in modo che il sistema sub-simbolico, in questo caso delle reti neurali, venga influenzato nel suo comportamento da della conoscenza pregressa sotto forma di regole logiche. Con l'induzione, invece, si mira a estrapolare dai dati prodotti dalla rete una serie di regole che ne esplichino il comportamento. Grazie a questo uso combinato di estrazione e imposizione della conoscenza, si vuole ottenere una teoria logica che sia una finestra sul modo di ragionare della rete sottostante. Naturalmente i due punti più problematici risultano essere proprio le modalità di condizionamento e induzione. Per questo, nel capitolo 4 dal modello si passa alla realizzazione di un piccolo prototipo che possa servire da caso di studio, in modo da esaminare il problema più nel dettaglio. Per la sua realizzazione sono usati i framework DL2 [16] per il condizionamento e NTP [27] per l'induzione. Entrambi i lavori sono stati scelti fra le tecniche più promettenti e all'avanguardia nel panorama della computazione neuro-simbolica.

Nel capitolo 5 si procede a elencare e discutere nel dettaglio gli esperimenti fatti e i loro risultati. Sono utilizzati come casi di studio dei dataset generati appositamente, ma che presentano due problematiche note in letteratura: lo sbilanciamento dei dati di addestramento e una dimensionalità elevata del problema. Come si potrà vedere, la componente di condizionamento risulta estremamente valida e performante in tutte le prove effettuate. L'induzione invece mostra dei grossi limiti soprattutto al crescere della complessità del problema.

Nel Capitolo 6 si procede alla valutazione del lavoro effettuato e ad elencare i

possibili lavori futuri.

Capitolo 2

Stato dell'arte

Il campo dell'intelligenza artificiale ha visto nel tempo il delinearsi di due correnti opposte: quella simbolica e quella connessionista. La prima, basata sull'utilizzo di modelli simbolici in modo da rappresentare, esplorare e inferire conoscenza. Entrando più nel dettaglio, questi metodi, basati sulla logica e sulla definizione di un chiaro modello formale che sfrutta simboli, hanno la particolarità di riuscire ad offrire una rappresentazione della conoscenza human-readable. Da questa risulta semplice inferire nuova conoscenza sfruttando diversi approcci logici (deduzione, induzione, abduzione...). I principali vantaggi derivanti dall'utilizzo di tali approcci sono dichiarativitá, osservabilità, interpretabilità e affidabilità. Possiedono però due grossi svantaggi: la difficoltà nell'acquisizione delle regole e la scarsa scalabilità. La seconda corrente invece, fondata sull'utilizzo di reti neurali e metodi statistici, ha il grande vantaggio di poter acquisire la conoscenza da degli esempi, di essere altamente scalabile e di riuscire a rappresentare delle basi di conoscenza complesse e altamente imprecise.

Sebbene a prima vista inconciliabili, da uno studio delle due correnti, si può notare come in realtà i vantaggi e gli svantaggi dei due si compensino a vicenda. Questo ha portato alla nascita di svariati approcci per la fusione dei due sistemi in modo da ottenere un ibrido che mantenga i principali punti di forza di entrambi. Le metodologie utilizzate per la creazione di tale tipologia di sistemi sono vari, ma l'area di ricerca che al momento risulta essere la più promettente e preponderante è quella della *Computazione Neuro-Simbolica* (NSC) [17]. Dei lavori principali appartenenti a questa corrente si parlerà nella Sezione 2.1, per poi passare ad approfondire nella Sezione 2.2 i vantaggi ottenibili da questi approcci ibridi, con particolare riguardo per l'aspetto di spiegabilità dei sistemi sub-simbolici.

2.1 Computazione Neuro-Simbolica

L'insieme di metodi diversi aventi come obiettivo l'integrazione di queste due opposte metodologie è estremamente ampio e variegato. Un analisi completa delle principali caratteristiche dei principali lavori appartenenti a tale categoria è rintracciabile in [8]. Approfondendo lo studio di tali tecniche è possibile notare come non sia presente un'organicità nelle metodologie adottate e nei risultati ricercati. Infatti, probabilmente a causa dell'estrema freschezza di tale ambito di ricerca, sono state vagliate le più disparate modalità d'integrazione, ognuna con i suoi metodi e scopi.

In questo panorama estremamente eterogeneo è comunque possibile estrarre due macro categorie in cui è possibile denotare una comunanza di intenti fra le tecniche a esse appartenenti. Tali insiemi sono:

- *Logic As Constraint*, dove l'idea principale si basa sull'utilizzo di vincoli logici (espressi come regole simboliche che utilizzano diversi tipi di astrazione —Logica del Primo Ordine, etc.) a guidare ed arricchire il processo di apprendimento di una rete neurale, fungendo quindi da fattore di regolarizzazione. In questo modo si ottiene come risultato una rete che abbia incorporata implicitamente la conoscenza di alto livello instillatagli durante il training e ne faccia uso in fase di valutazione.

- *Differentiable Programming*, dove i lavori appartenenti a questo gruppo sono accomunati dall'idea di rendere i programmi logici differenziabili. Si vuole cioè ottenere un mapping del dominio simbolico in uno completamente numerico, in modo da poter utilizzare anche per tale categoria di problemi le metodologie e gli algoritmi di ottimizzazione appartenenti all'area del sub-simbolico. In pratica la logica guida la costruzione di una rete neurale equivalente nel funzionamento ai framework simbolici.

Verrà ora data una descrizione dei diversi lavori appartenenti a queste due categorie.

2.1.1 Logic As Constraint

A riprova dell'estrema varietà presente nel campo della NSC, anche all'interno di una stessa corrente i metodi utilizzati differiscono anche di molto nei modi e nelle tecniche utilizzate. Quelli descritti di seguito sono stati scelti in quanto maggiormente rappresentativi dell'intero insieme.

Real Logic

Nei lavori [30] e [13] viene definita la semantica di una logica di primo ordine fuzzy con dominio nell'insieme dei numeri reali. In questo modo si cerca di colmare il divario esistente fra il ragionamento simbolico e l'apprendimento empirico delle reti neurali, basato appunto su dati numerici. Il risultato viene chiamato *Real Logic*. Nello specifico questo linguaggio L è composto da un insieme di costanti C, un insieme di funzioni F e uno di predicati P. Come connettivi logici sono utilizzati costrutti presi dalla logica fuzzy (t-norm e derivati) in modo da lavorare nel dominio dei numeri reali nell'intervallo $[0, 1]$. L'operazione di mapping degli elementi simbolici nello spazio vettoriale è definita Grounding. Le costanti sono mappate in vettori (la lunghezza è un iperparametro del modello), le funzioni simboliche sono equivalenti a delle funzioni nello spazio vettoriale e i predicati sono mappati in funzioni con codominio in $[0, 1]$. Questo valore definisce appunto il grado di verità del predicato in un'ottica fuzzy. I quantificatori sono definiti come delle operazioni di aggregazione. Il processo di apprendimento è visto come un problema di soddisfacibilità. Si cerca di ottimizzare il grounding di atomi, predicati e funzioni in modo da massimizzare la soddisfacibilità di una formula data in input come vera. Per esempio dato il predicato *uomo(seneca)*, la rete cercherà di ottimizzare il grounding di *uomo* e *seneca* (ottimizzando i parametri visti sopra) in modo che il grado di verità stimato sia vicino a 1. Viene ricercato il grounding ottimo per ogni elemento del linguaggio (atomi, funzioni, predicati). Il processo di apprendimento viene valutato con un ponteggio nell'intervallo $[0, 1]$ (livello di soddisfacibilità). Per riassumere, intuitivamente le formule logiche vengono usate per comporre una loss function che miri ad addestrare una rete capace di approssimare il valore di verità (in un intervallo $[0, 1]$) delle formule date in input. Questo viene fatto cercando la migliore rappresentazione possibili per i costrutti simbolici nello spazio vettoriale (grounding di atomi, funzioni, predicati), in modo che la soddisfacibilità della rete sia il più vicino possibile a 1 sui dati di test. La rete risultante sarà capace di apprendere a partire da degli esempi reali etichettati nel modo corretto, ma manterrà in fase di valutazione l'impronta logica data in fase di addestramento.

Rules Injection

Nel lavoro [11], invece, viene presentato un metodo mirato a incorporare delle regole di implicazione logica nel processo di estrapolazione di rappresentazioni distribuite, atto alla creazione automatica di una base di conoscenza. Molto sinteticamente, vengono ricercati gli embedding delle tuple di cui si vuole costruire la rappresentazione distribuita e delle regole di implicazione che devono essere imposte sulla base di conoscenza. Questa rappresentazione è derivata dal modello di

fattorizzazione a matrice [26], secondo il quale ogni relazione è rappresentata da un vettore appartenente all'insieme dei reali di dimensione k. La funzione da ottimizzare è basata sul valore di compatibilità fra una tupla e una relazione. Viene infatti imposto che in una implicazione logica questo valore sia simile (<=) fra la relazione implicante e quella implicata, nei confronti di una tupla t. In questo modo l'obiettivo dell'ottimizzazione è nella minimizzazione della distanza fra questi due valori (si impone un ordinamento parziale fra le varie relazioni). Un'importante ottimizzazione sta nel considerare uno spazio degli embeddings delle tuple-entità ristretto (reali positivi) in modo che la funzione da minimizzare possa essere riscritta in modo da essere indipendente dalla tuple facenti parti il training set. In questo modo il costo per il calcolo della funzione di ottimizzazione è lineare rispetto al numero di relazioni logiche imposte e non rispetto a quello delle tuple interessate da queste implicazioni. Questo particolare differenzia il lavoro in esame da quelli precedenti, limitati nella scalabilità proprio da questo problema. Come risultato si avrà che gli embedding risultanti saranno influenzati delle regole di implicazione imposte sul modello.

Semantic Loss

Da un'analisi dei lavori precedenti l'autore di [34] fa notare come tutte si basino sul tentativo di riduzione dei vincoli in una variante numerica differenziabile. Come si è visto sopra, questo mapping avviene normalmente sostituendo gli operatori logici (congiunzioni) con la loro *t-norm* e le implicazioni con delle disuguaglianze. Il prezzo di questa scelta sta nell'ottenere una funzione che tenga conto della sintassi del vincolo e non della semantica (si ha così una perdita del preciso significato del vincolo). Il lavoro [34] mira appunto alla risoluzione di questo problema, creando una funzione di perdita *semantica* (Semantic Loss), che mantenga e rinforzi il significato del vincolo imposto sull'output della rete neurale. Il metodo creato è fatto in modo da poter essere compatibile con ogni rete *feedforward*. La funzione risultante dovrebbe avere la forma:

$$loss\ di\ partenza + w \times semantic\ loss$$

con w un qualsiasi peso. Facendo un esempio, nel caso di una rete che dovrebbe avere in output un vettore con codifica *one-hot*, questa proprietà dovrebbe essere imposta come vincolo. Quindi questa funzione semantica dovrebbe catturare quanto la predizione si discosti dall'avere uno solo degli output a 1 e tutti e rimanenti a 0. Intuitivamente questa funzione viene paragonata al logaritmo negativo della probabilità di generare uno stato che soddisfi il vincolo. La formula della *semantic loss* è descritta da una formula Booleana (viene fatto uso di concetti mutuati dalla logica proposizionale), e per questo motivo questa metodologia è applicabile a tutti

i domini per i quali i vincoli possano essere espressi in tale modalità. In caso di vincoli con complessità elevata, che sarebbero difficilmente trattabili, viene introdotto un framework apposito, basato sulla tecnica di *Weighted Model Counting* [29]. I risultati risultano ottimi sia in caso di apprendimento semi-supervisionato (i vincoli hanno un effetto di regolarizzazione sui dati non etichettati), sia in caso di apprendimento supervisionato, ma con dominio estremamente complesso.

Network Augmentation

I lavori visti fino ad ora, per quanto diversi fra loro, sono accomunati dall'utilizzo di una loss function modificata, che tenga conto dei vincoli logici che si vuole instillare durante il processo di apprendimento. Tale metodologia è utilizzata anche in [16]. Approccio completamente differente è quello riscontrabile in [18]. Tale lavoro si basa sull'idea di modellare i vincoli da imporre come una parte stessa della rete, in moda che i vincoli vengano rinforzati automaticamente durante il processo di apprendimento. In pratica una parte della rete viene addestrata in modo tale da approssimare l'influsso che i vincoli avrebbero sul risultato facendo in modo che dei neuroni *semantici* si attivino solo e soltanto se l'input è coerente con i vincoli imposti.

2.1.2 Differentiable Programming

Anche in questo caso, a causa della moltitudine di tecnologie appartenenti a tale categoria, ne sono state selezionate alcune ritenute maggiormente rappresentative.

Deep Logic

In [6] si cerca di costruire un motore generico per l'esecuzione di programmi logici utilizzando delle reti neurali (reti particolari basate sulla memoria, Memory Networks, con meccanismi di *Attention*). Il focus è sull'utilizzo di una architettura di rete unica anche per task diversi. Del contesto e della query viene calcolato l'embedding (tensore) e tutto viene dato in pasto alla rete (addestrata in backpropagation su un dataset contenente esempi di programmi logici con esempi di tutti i task esaminati). I risultati mostrano come non si arrivi ancora a una generalizzazione tale da permettere un confronto con dei motori tradizionali (Prolog). Una causa è nel limite dato dalla rappresentazione degli embedding (numero di dimensioni finite), che non permettono di mantenere tutta la conoscenza.

Differentiable Probabilistic Logic Networks

In [24], invece, viene preso in esame il modello ibrido di inferenza logica probabilistica utilizzato all'interno di OpenCog. Questo è basato su di un Unified Rule

Engine (URE) in cui la conoscenza viene modellata come un grafo. I nodi sono gli elementi della conoscenza (soggetti, oggetti), gli archi i predicati. Il processo di inferenza consiste nel matching di un sottografo che soddisfi i vincoli dati (possibile sia backward che forward chaining). Il grado di probabilità dell'output viene calcolato utilizzando delle funzioni scritte a mano, che presi i valori di probabilità dell'input calcolino quello in output. Fine del lavoro è utilizzare dei tensori come valori di verità in modo che sia possibile applicare degli algoritmi di ottimizzazione quali backpropagation per stimare i valori di verità corretti e ottenere un motore di inferenza dove le operazioni di stima del grado di verità siano generiche. Sono state fatte anche delle prove in merito all'inferenza di nuove regole in fase di training.

Neural Theorem Provers

Nel lavoro [27] viene presentato una modalità differenziabile di ragionamento su delle basi di conoscenza (*Neural Theorem Provers*). Il fine principale è quello di unire i vantaggi dei ragionatori simbolici (ragionamento in più passi, interpretabilità, uso della conoscenza sul dominio) con l'abilità di poter utilizzare una rappresentazione numerica dei simboli. Questo porterebbe alla capacità di poter ragionare in termini di similarità dei simboli nello spazio di rappresentazione numerica, ottenendo così una maggiore generalizzazione nel processo deduttivo. Come risultato ci si pone quattro obiettivi:

- ottenere che le rappresentazioni di simboli simili siano vicine nello spazio vettoriale;
- utilizzare queste similarità per provare le query sulla base di conoscenza;
- indurre delle nuove regole a partire dai dati;
- effettuare dei ragionamenti in più step.

Nella pratica quello che si fa è andare a mappare il *proof tree* ottenuto applicando una versione dell'algoritmo di backward chaining di Prolog in una rete neurale. Per far questo sono state identificate e mappate le tre operazioni primarie dell'algoritmo di chaining: *unificazione*, *and* e *or*. Questi moduli fungono da primitive per la definizione della rete completa, che può essere addestrata, attraverso un algoritmo di backpropagation, in modo da identificare la rappresentazione migliore dei simboli nello spazio vettoriale. Si fa uso di tutti gli elementi della logica del primo ordine escluse le funzioni.

Tensor Log

Ultimo fra i lavori proposti è [7]. In questo viene introdotta una modalità d'integrazione fra il ragionamento logico probabilistico e le metodologie sub-simboliche. L'idea principale è quella di utilizzare come motore d'inferenza l'algoritmo di *Belief Propagation* [35], applicato su un *factor graph* dove le variabili corrispondono a un possibile assegnamento ad una variabile logica (costanti del database) e i fattori corrispondono ai predicati della teoria in esame. Questi fattori hanno il ruolo di rendere gli assegnamenti consistenti con i predicati presenti nella teoria. Il processo di inferenza logica si riduce al calcolo di una funzione $f(p, a)$ che dato un predicato p e un vettore a, restituisca in output un vettore y contenente la distribuzione delle confidenze sulle costanti del database. Questa funzione è proprio il risultato dell'algoritmo di Belief Propagation (tempo di convergenza lineare su un grafo *Polytree*). Il processo di apprendimento (discesa del gradiente) è mirato alla stima dei pesi dei fatti contenuti nella teoria e necessari al calcolo della confidenza. TensorLog, ossia l'implementazione di tale modello teorico (realizzato su TensorFlow e Theano) è stato valutato nelle performance andando ad effettuare un confronto con ProbLog2 [15]. Sebbene i risultati siano estremamente positivi, va considerato che tale modello è al momento applicabile ad un piccolissimo sottoinsieme dei programmi valutabili da ProbLog2, rendendo il confronto alquanto impari.

2.1.3 Neural Program Induction

Fra le opere non assimilabili alle due categorie proposte sopra, ma di particolare rilevanza nel campo della NSC, vi sono quelle legate al problema della *Neural Program Induction*. Molto semplicemente, questa branca è focalizzata alla sintesi automatizzata di programmi per la risoluzione di task a partire da dati ed esempi.

Neuro Symbolic Program Synthesis

In [21] viene proposta una tecnica capace di costruire incrementalmente un programma in un dato DSL a partire da degli esempi. Gli altri lavori nel campo hanno il problema di richiedere un grosso lavoro manuale in modo da definire delle euristiche che guidino la ricerca nello spazio degli infiniti programmi possibili. In questo si distingue invece la *Neuro Symbolic Program Synthesis*, in grado appunto di imparare dagli esempi forniti durante l'addestramento, senza bisogno di ulteriori configurazioni manuali. Altri punti di forza di questa tecnica sono l'interpretabilità (alcune tecniche mirano ad approssimare il comportamento del programma desiderato tramite una rete neurale, perdendo quindi di comprensibilità) e la capacità di generalizzare su più task. Il metodo è basato su due moduli distinti:

- la rete di I/O Cross Correlation, capace di produrre una rappresentazione continua degli esempi di input/output dei programmi dati come esempio;

- la rete neurale Recursive-Reverse-Recursive (R3NN), che data la rappresentazione continua degli esempi, usati in modo da condizionare la costruzione del programma sulla loro base, effettua la sintetizzazione vera e propria del programma procedendo per espansione di programmi parziali. Iterativamente viene costruito un albero di parsing selezionando, in base a una certa probabilità, quale simbolo espandere utilizzando una delle regole del DSL (grammatica libera dal contesto).

I risultati sono molto buoni sia nella sintesi di programmi visti in fase di training, sia, in misura minore, nella sintesi di programmi di cui non erano stati dati esempi durante il training. Una delle possibili strade suggerite è la sostituzione dell'approccio supervisionato con delle tecniche di Reinforcement Learning.

Neural Guided Search

Il lavoro [36] presenta un approccio differente rispetto a quello visto sopra. Infatti, mentre quello viene posto nella categoria della sintesi diretta, insieme che raccoglie gli approcci accomunati dall'uso di particolari tipi di reti neurali in modo da generare direttamente l'albero rappresentante il programma cercato, questo si pone nella categoria della *Neural Guided Search*. Caratteristica principale di questi metodi è l'utilizzo di tecniche di machine learning per guidare una ricerca discreta nello spazio delle ipotesi. Nello specifico viene introdotta un'espansione del linguaggio miniKanren [5], linguaggio del paradigma relazionale, che permetta di valutare qual è il miglior candidato espandibile durante il processo di generazione del programma. In modo molto conciso, miniKanren rappresenta il problema come una serie di disgiunzioni fra i possibili programmi candidati, e di vincoli che devono essere soddisfatti perché la soluzione sia consistente con gli esempi. Un agente esterno, preso in input l'insieme dei vincoli, ed effettuatone l'embedding, esegue la valutazione sui singoli vincoli e quindi sui programmi candidati, andando a decretare quale di questi dovrà essere espanso. Intuitivamente il punteggio di ogni candidato può essere visto come la plausibilità che i vincoli a esso associato vengano soddisfatti. Sono state effettuate diverse prove con delle strategie di embedding differenti in modo da valutare quale fosse la più performante (RNN o GNN). I risultati mostrano come il sistema si sia dimostrato competitivo rispetto alle soluzioni più affermate (es. RobustFill [12]). Sono stati ottenuti dei buoni risultati anche in termini di generalizzazione, infatti il modello addestrato si è dimostrato valido anche per problemi più ampi rispetto a quelli visti in fase di training.

AlphaNPI

Anche in [22] fine ultimo è l'apprendimento automatico delle modalità di esecuzione di un task dato in input. Nello specifico, è proposto un algoritmo, AlphaNPI, che mira al training di un modello NPI [25] (*Neural Program Interpreter*) utilizzando delle tecniche di Reinforcement Learning, nello specifico un adattamento dell'algoritmo AlphaZero [31]. I modelli NPI mirano appunto all'apprendimento delle modalità di esecuzione di un task a partire dalle tracce di esecuzione di un programma reale. Questa particolarità rendeva questa tecnica inadatta al dominio del *Program Synthesis*, necessitando appunto di un programma reale come base da cui apprendere. Molto sinteticamente, tali modelli si basano su una rete LSTM che ad ogni istante t, dato in input una rappresentazione dello stato dell'ambiente e una dell'ultima azione eseguita, da in output un'indicazione di terminazione (raggiungimento del target) oppure un'azione da eseguire. L'esecuzione di tale azione porta ad un aggiornamento dello stato e quindi a un nuova valutazione. I programmi imparati possono far uso di iterazione e ricorsione. L'addestramento supervisionato prevede il confronto delle azioni decise dalla rete con quelle del programma reale e quindi la penalizzazione della NPI in caso di mancato match. L'estensione vista in questa pubblicazione mira proprio a superare il problema dell'apprendimento supervisionato. Viene utilizzata una variante di AlphaZero che fa uso di una ricerca ad albero Monte Carlo ricorsiva (*MCTS*). I dati generati da questa ricerca (nodi dell'albero di ricerca) vengono collezionati in un buffer apposito. Questi nodi/tuple vengono utilizzati per addestrare la rete AlphaNPI in modo da massimizzare la similarità nel comportamento fra i due agenti. La rete risultante ha dato in fase di test ottimi risultati in termini di generalizzazione, riuscendo ad ottenere gli stessi risultati degli approcci completamente supervisionati.

DeepProbLog

Altro lavoro di notevole interesse è quello proposto in [19]. Vengono estese le funzionalità del linguaggio logico ProbLog [10], andandolo ad integrare con una nuova categoria di predicati, quelli neurali, che incapsulano al loro interno il funzionamento di tali reti. È difficilmente categorizzabile perché è l'unico a mantenere tutte le funzionalità dei linguaggi logici simbolici (inferenza, facilità nella spiegazione) andando a guadagnare l'efficacia dei metodi sub-simbolici, quali l'apprendimento da dati di esempio. L'incertezza della conoscenza così derivata viene gestita grazie alle funzionalità probabilistiche già presenti in ProbLog. Ci sono esempi di applicazione di questo framework anche nel campo della *Logic Induction*, cosa che ha portato al suo collocamento in tale categoria. In particolare si sono fatte delle prove, ottenendo degli ottimi risultati, in task di completamento di programmi

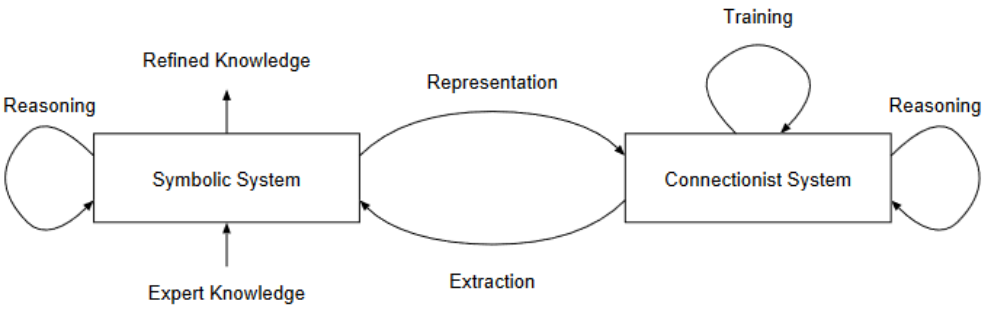

Figura 2.1: Ciclo di apprendimento neuro-simbolico [2].

Forth, dove le mancanze devono essere colmate con delle reti neurali addestrate su esempi di input/output dell'intero programma.

2.1.4 Tassonomia

Viene ora proposto uno schema riassuntivo delle caratteristiche dei vari approcci esaminati che cerchi di metterne in luce eventuali analogie o differenze. È usata come base la categorizzazione proposta nel lavoro [8]. Secondo tale categorizzazione i metodi appartenenti alla famiglia della *NSC* possono essere catalogati in base a come affrontano i seguenti problemi: *Knowledge Representation*, *Learning*, *Reasoning* e *Knowledge Extraction*.

Questa categorizzazione deriva della strutturazione standard data ai sistemi neuro-simbolici per rappresentarne le modalità di apprendimento e funzionamento (Figura 2.1).

La *Knowledge Representation* racchiude il problema fondamentale di come la conoscenza simbolica viene rappresentata e quindi di come è gestito il mapping fra questo tipo di rappresentazione e quella numerica. Sono previste due tipologie di massima, *Logica proposizionale* e *Logica del primo ordine*, ognuna con le sue tecniche specifiche.

La sfera del *Learning* prende invece in esame come le componenti simboliche e sub-simboliche sono integrate nel processo di apprendimento. Sono previste tre modalità:

- *Logic Induction*, in questo caso l'apprendimento è mirato alla costruzione di programmi logici a partire da degli esempi. In pratica si utilizza un motore basato su operazioni differenziabili per affinare delle regole logiche;

- *Ibrida Orizzontale*, i metodi di questo tipo mirano all'utilizzo della conoscenza simbolica durante il processo di addestramento di una rete neurale in modo da migliorarne l'efficienza. Tutti i lavori appartenenti alla categoria della *Logic as Constraint* appartengono a questo gruppo.

- *Ibrida Verticale*, in questo caso la componente simbolica e quella *neurale* sono tenute separate in moduli distinti. Il processo di apprendimento è però condiviso.

Con *Reasoning* si intende invece la capacità di una rete neurale di effettuare delle inferenze come nei motori logici tradizionali. Anche in questo caso sono state individuate tre categorie, basate sulle tecniche alla base del ragionamento: *Chaining*, forward o backward, *SAT*, dove in pratica si cerca di approssimare la soluzione al problema della migliore soddisfacibilità, *Reletionship Reasoning*, dove invece il focus è sulle relazioni presenti fra le entità della base di conoscenza.

Con *Knowledge Extraction*, invece, si intende la capacità di un sistema NSC di trasformare la conoscenza da un formato sub-simbolico a uno simbolico.

Tale categorizzazione verrà ora utilizzata per classificare le diverse tecniche viste nella Sezione 2.1. La Tabella 2.2 fa riferimento agli aspetti di *Knowledge Representation* e *Learning*, mentre la Tabella 2.1 a quelli di *Reasoning* e *Knowledge Extraction*.

Tabella 2.1:
Tassonomia 1

	Reasoning			Knowledge Extraction
	Chaining	Approssimazione SAT	Reletionship Reasoning	Logic Program Induction
Logic Tensor Networks[30]		✓		
Lifted Rules Injection[11]			✓	
Semantic Loss[34]		✓		
DeepLogic[6]				
DPLN[24]	✓			
Differentiable Proving[27]	✓*			
TensorLog[7]	✓			
NSPS[21]				✓
Neural Guided Synthesis[36]			✓	✓
AlphaNPI[22]				✓
DeepProbLog[19]	✓			✓

Tabella 2.2:
Tassonomia 2

	Knowledge Representation		Learning		
	Logica Proposizionale	Logica del Primo Ordine	Logic Induction	Ibrida Orizzontale	Ibrida Verticale
Logic Tensor Networks[30]		✓		✓	
Lifted Rules Injection[11]		✓		✓	
Semantic Loss[34]	✓			✓	
DeepLogic[6]		✓			
DPLN[24]		✓			
Differentiable Proving[27]		✓	✓		
TensorLog[7]		✓			
NSPS[21]					
Neural Guided Synthesis[36]					✓
AlphaNPI[22]					
DeepProbLog[19]		✓			✓

2.2 Explainability

Come si è visto nella Sezione 2.1 i metodi di unificazione proposti nel campo della computazione neuro-simbolica sono i più disparati. La spinta che potrebbe fornire un'organicità ai diversi approcci potrebbe essere quello dato dal bisogno di spiegare il comportamento dei motori sub-simbolici, *DNN* in particolare. Infatti, sebbene queste metodologie offrano delle prestazioni ottime da un punto di vista di scalabilità e capacità di apprendimento, si è ancora lontani dal riuscire a dare una spiegazione del perché di un dato risultato. Questo problema, in un contesto in cui questo tipo di metodi sono utilizzati per prendere delle decisioni che potenzialmente possono impattare sulla vita delle persone, risulta essere di primaria importanza. È infatti impossibile affidarsi completamente a questi meccanismi senza avere il controllo assoluto sul loro modo di operare e di ragionare. Questo risulta essere un campo di ricerca aperta, noto con il nome di *Explainable AI* (*XAI*), e che riceve sempre più attenzione da parte dalle diverse comunità scientifiche. A sostegno dell'estrema novità del problema vi è il fatto che ancora non sia disponibile una vera e propria definizione del concetto di spiegabile su cui i ricercatori concordino. Le proprietà che un sistema di tale categoria dovrebbe avere sono tuttora oggetto di discussione. Quelle che però sembrano avere una maggiore preponderanza sono:

- *affidabilità*, inteso come la capacità di dare sempre lo stesso risultato per lo stesso problema;
- *causalità*, cioè la capacità di correlare le variabili del problema in rapporti di causa-effetto;
- *trasferibilità*, la corretta comprensione di un modello è una condizione basilare nella ricerca di un sistema dove la conoscenza sia trasferibile. Infatti, sapere esattamente cosa un modello rappresenta e come ragiona è il primo passo per il riutilizzo di tale conoscenza in un altro frangente. Per tale motivo questa proprietà è stata inserita nella lista delle proprietà desiderabili di un sistema spiegabile;
- *informatività*, cioè la capacità di fornire informazioni sul problema gestito;
- *confidenza*, in questo genere di sistemi stabilità e robustezza sono dei componenti fondamentali. Si dovrebbe avere sempre la garanzia che le modalità di funzionamento siano tali da garantire l'affidabilità del responso;
- *equità*, cioè la capacità di fornire una garanzia di comportamento equo da un punto etico e morale;

- *accessibilità*, la trasparenza è vista come il primo passo verso dei sistemi realmente accessibili a tutti. Infatti l'incomprensibilità degli attuali modelli è l'ostacolo principale alla diffusione di tali tecniche verso gli utilizzatori meno esperti.

- *interattività*, la trasparenza garantirebbe un maggior grado di interattività con il sistema stesso da parte degli utilizzatori;

- *controllo del livello di privacy*, l'opacità dei sistemi non permette infatti l'analisi delle informazioni che sono state effettivamente acquisite. La realizzazione di un sistema spiegabile potrebbe garantire la totale trasparenza su queste informazioni, eliminando ogni rischio per la privacy.

Uno dei concetti basilari su cui si concorda è la differenza fra *spiegabilità* di un sistema e la sua *interpretabilità*. Questa distinzione risulta fondamentale per comprendere l'obiettivo delle tecniche appartenenti a quest'area di studio.

Intuitivamente, mentre con *interpretablità* si intende la capacità di capire il funzionamento di un modello da un punto di vista algoritmico, il termine *spiegabilità* rappresenta la capacità di comprendere quali sono le reali motivazioni che hanno portato a dare un certo valutazione. Questa differenza potrebbe sembrare poco importante, ma è invece di fondamentale importanza.

Per spiegarla meglio si può fare un esempio banale: se si fa cadere un oggetto, questo cadrà e probabilmente si romperà. L'interpretazione può dare le ragioni *meccaniche* della caduta ma senza coglierne veramente le cause: l'oggetto è caduto, per questo ha toccato terra e per tale ragione si è spezzato. La spiegazione dell'evento, invece, dovrebbe dare le vere ragioni alla base del fenomeno. In questo caso potrebbe essere: l'oggetto è caduto, a causa della gravità ha ottenuto una certa accelerazione che l'ha portato all'urto con il pavimento. L'energia accumulata nella caduta ha portato alla rottura dell'oggetto all'impatto. Riportato sui modelli d'apprendimento automatico si ha che, non si vuole ottenere una spiegazione meccanica dell'algoritmo che ha prodotto il risultato, ma le vere cause legate alla logica di dominio.

Quest'area di ricerca ha visto il proliferare di svariati metodi e tecniche, con unico fine quello di rendere trasparente e affidabile il funzionamento delle tecniche di deep e machine learning (ML). Mentre le tecniche tradizionali di ML godono di un certo grado di trasparenza innata, questo non è vero per le metodologie di deep learning. Nello schema in Figura 2.2 sono mostrate schematicamente alcune delle diverse tecniche *post-hoc* utilizzate per cercare di fornire una maggiore trasparenza dei diversi metodi di apprendimento automatico. Nello specifico abbiamo sei diversi approcci possibili:

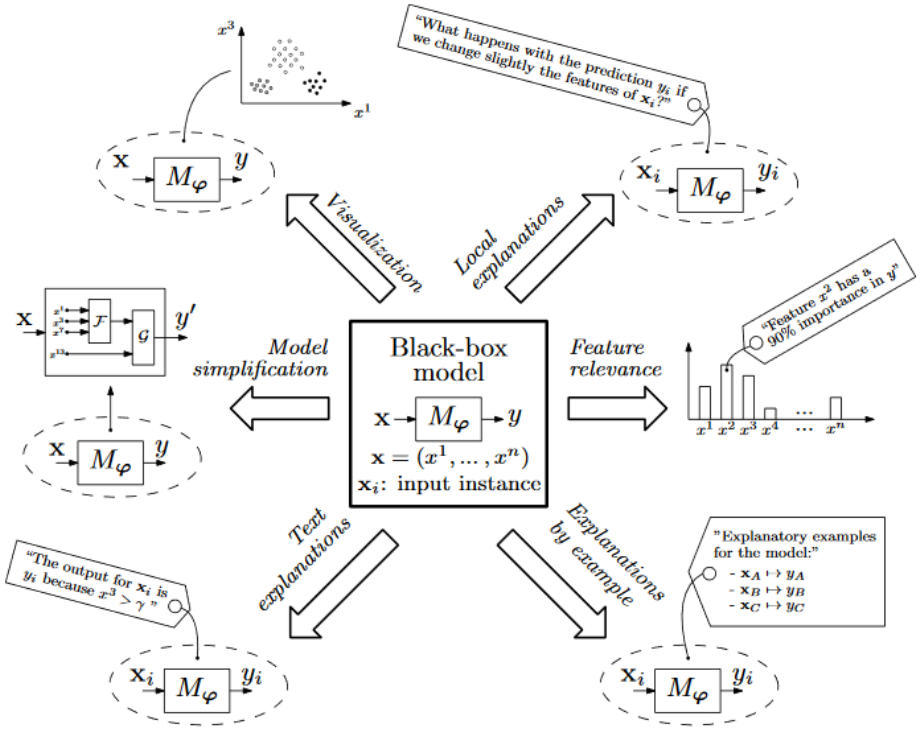

Figura 2.2: Diversi approcci disponibili per la giustificazione a posteriori di un modello di machine learning [1].

- *spiegazione testuale*, consiste nella generazione di spiegazioni testuali che aiutino a giustificare il comportamento del modello. La spiegazione potrebbe essere in linguaggio naturale, in un linguaggio logico, etc.;
- *spiegazione tramite esempi*, consiste nell'estrazione di esempi abbastanza rappresentativi da permettere la comprensione del modello;
- *semplificazione del modello*, insieme di tecniche mirate alla costruzione di un sistema meno complesso dell'originale ma che ne mimi il comportamento di massima. La minore complessità permette una migliore analisi.
- *visualizzazione*, mira alla spiegazione di un modello tramite la visualizzazione del suo comportamento. Per renderlo possibile è spesso utilizzato un processo di riduzione di dimensionalità;
- *spiegazione locale*, consiste nella riduzione della complessità del problema grazie alla ricerca di spiegazioni per una sotto-porzione del dominio approssimato dal modello.
- *attributi rilevanti*, consiste nella quantificazione di come un attributo del problema influenzi l'output finale. L'analisi comparata di questi attributi fornisce una spiegazione indiretta sul funzionamento del modello.

Un'alternativa a questa tipologia di tecniche è fornita nel Capitolo 3. Viene infatti presentata una soluzione al problema della spiegabilità basata sulla creazione di un sistema neuro-simbolico ibrido. Questo utilizza le proprietà di trasparenza dei metodi simbolici per potenziare e giustificare il comportamento di quelli sub-simbolici (opachi).

Capitolo 3

Modello concettuale: Neuro-Simbolico per l'Explainability

Uno dei vantaggi principali nell'utilizzo di tecniche simboliche per la risoluzione di problemi logici e di apprendimento è, come si è visto nella prima parte, legato alla garanzia di ottenere un sistema trasparente e comprensibile da parte dell'utente finale. Prendendo come esempio i motori *Prolog* e di deduzione logica si può notare come qualsiasi spiegazione sia corredata da una serie di fatti e di regole che hanno portato alla risposta finale. La trasparenza è una parte fondante nel meccanismo stesso di computazione. È infatti possibile esaminare tutto il percorso che ha portato a un dato responso, cosa che garantisce l'affidabilità e la riproducibilità del procedimento. La stessa cosa, come si è visto chiaramente nella sezione legata al problema della *spiegabilità*, non è applicabile alle tecniche sub-simboliche, basate su metodologie matematiche e statistiche che non garantiscono la corretta comprensione di procedimento e risultato.

L'unificazione fra le due correnti, quindi, tralasciando il resto dei benefici, è auspicabile anche solo per le potenzialità nel poter correggere questa grave lacuna dei sistemi numerici, che ne impediscono l'utilizzo in contesti dove affidabilità e garanzia di *imparzialità* sono requisiti primari. Di questa problematica, e di come i metodi neuro-simbolici possano essere una possibile soluzione, si tratterà in questo capitolo. Partendo dalla presentazione dei modelli più all'avanguardia noti in letteratura, si passerà poi a definirne un nuovo raffinamento, con l'obiettivo di superarne i limiti riscontrati.

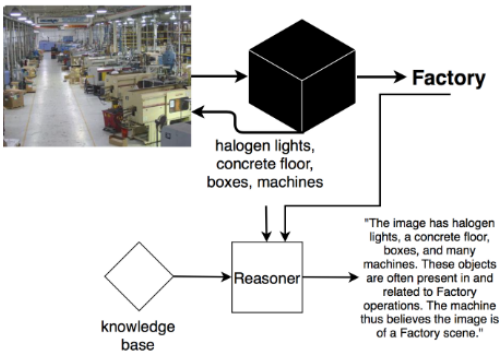

Figura 3.1: Modello di Doran [14] per la spiegabilità.

3.1 Fondamenta teoriche

Un primo passo per la risoluzione del problema della comprensibilità che preveda l'utilizzo di tecniche simboliche affiancate a quelle sub-simboliche è proposto nel lavoro [14]. Il modello proposto, rappresentato in Figura 3.1, prevede l'utilizzo di un ragionatore simbolico con l'obiettivo di dare delle spiegazioni sull'output di una rete neurale. L'autore sostiene che una vera giustificazione automatica dei motivi che hanno portato a un output di una rete possa essere data attraverso l'utilizzo di un motore logico. La conoscenza utilizzata da tale strumento dovrebbe essere codificata in una *Knowledge Base*(KB) compilata a priori da degli esperti di dominio, che in tal modo fornirebbero tutti gli strumenti per poter comprendere la decisione del sistema. Oggetto del ragionamento dovrebbero essere le caratteristiche estratte automaticamente dall'entità oggetto di classificazione. Per esempio, in un task di classificazione di immagini capace di riconoscere il contesto in cui una foto è stata presa, il procedimento di analisi e giustificazione del risultato consisterebbe in:

- estrazione del contenuto dell'immagine. Per esempio in un ambiente domestico si potrebbe trovare un frigorifero, un forno, etc.;
- utilizzo del ragionatore logico per giustificare la scelta del classificatore sulla base degli elementi estratti dall'immagine. Proseguendo con l'esempio precedente, prendiamo un sistema con una *KB* contenente una regola del tipo:

  ```
  cucina(X) :- forno(X), frigorifero(X).
  ```

  Tale regola, formalizzata nel linguaggio logico *Prolog*, codifica una relazione con il seguente significato: se in un ambiente X viene identificata la presenza

di un *forno* e di un *frigorifero*, allora tale ambiente è una *cucina*. Una conoscenza pregressa di questo tipo riuscirebbe a fornire un'indicazione di quali sono gli elementi dell'immagine che hanno portato all'assegnazione della classe finale.

Tale sistema, seppur nella sua semplicità, fornisce intuitivamente un'idea chiara di come il ragionamento simbolico possa essere non in competizione, ma piuttosto di supporto, alle tecniche sub-simboliche.

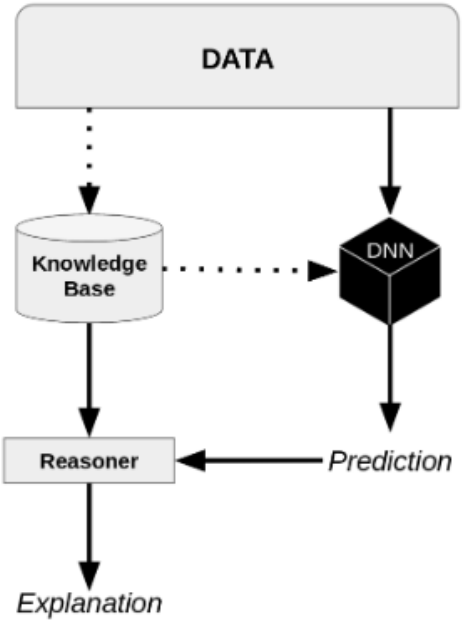

Figura 3.2: Modello di Bennetot [4] per la spiegabilità.

Tale modello è stato ulteriormente raffinato in un secondo modello [4], visibile in Figura 3.2, nel quale alcune tecniche derivate dal campo della *NSC* sono utilizzate per superare alcuni dei limiti identificati nel lavoro originale. In particolare viene fatto notare come le previsioni generate dalla rete siano in realtà completamente scorrelate dalla giustificazione data dal ragionatore, essendo i due moduli completamente distinti e basandosi su delle informazioni sconnesse fra loro. Esempi concreti nel caso del motore sub-simbolico e regole logiche nel ragionatore.

Per superare questo limite viene proposto un modello per il quale la base di conoscenza venga generata a partire dagli stessi dati su cui la rete neurale è addestrata, e queste stesse regole vengano utilizzate come fattori di regolarizzazione nel processo di apprendimento della rete stessa. In questo modo, oltre ad ottenere

una rete più performante grazie all'utilizzo della conoscenza simbolica, si otterrà un ragionatore che dia delle spiegazioni sulla base della stessa teoria che il modello numerico sta approssimando.

3.2 Verso un nuovo modello

Cercando di schematizzare, i contributi dei lavori di cui sopra al problema della *spiegabilità* nei sistemi sub-simbolici possono essere riassunti con:

- utilizzo di una KB su cui effettuare del ragionamento sul comportamento della rete. Di tale KB è prevista la compilazione in due modalità distinte. Completamente a priori da parte di esperti del dominio. Completamente a partire dai dati di addestramento grazie a delle tecniche di estrazione della conoscenza;
- condizionamento della rete per migliorarne l'affidabilità e correggerne i possibili bias;

Si vuole proporre quindi un raffinamento ulteriore di tali modelli, che miri a risolverne le debolezze e, al contempo, a sfruttare appieno le potenzialità dei meccanismi di integrazione fra simbolico e sub-simbolico visti nel Capitolo 2.

3.2.1 Proprietà richieste

Come prima cosa vanno identificate le debolezze dei lavori visti sopra. Questi fungeranno da punto di partenza per la definizione delle proprietà richieste al nuovo modello.

Il primo e più importante appunto che si può fare al modello di [4] è che le spiegazioni date risultano ancora scorrelate dal reale comportamento del sistema sub-simbolico. Infatti, sebbene l'utilizzo di una KB generata a partire dai dati di addestramento sia sicuramente un passo in avanti rispetto al lavoro precedente nel risolvere tale problema, questo risulta ancora insufficiente. Infatti, non è presente alcuna garanzia che la conoscenza estratta a partire dai dati di addestramento rifletta ciò che effettivamente la rete ha appreso. Ipoteticamente, nel caso in cui i dati riflettano esattamente la struttura del dominio e l'algoritmo di estrazione dati sia ottimale, il contenuto della KB sarebbe lo stesso di quello di una KB compilata a priori da una persona con un'ottima conoscenza del problema. In questo caso varrebbero così le stesse critiche che [4] muove a [14].

L'esclusione della conoscenza a priori in formato logico, inoltre, pone dei limiti al miglioramento nell'addestramento del modello sub-simbolico. Limitandosi ad effettuare la regolarizzazione a partire da delle regole ottenute a partire dai dati, si

limitano pesantemente le potenzialità delle metodologie di *condizionamento logico*. Queste, infatti, esprimono il loro maggiore potenziale nei casi in cui i dati di addestramento non forniscano una rappresentazione chiara del dominio, e quindi un intervento esterno possa fare la differenza.

Detto questo, il sistema ricercato dovrà riuscire a garantire il soddisfacimento delle seguenti proprietà:

1. recupero della teoria logica che approssimi al meglio il comportamento del classificatore (spiegazione globale).

2. garantire la massima convergenza possibile con il modello approssimato, di modo che i responsi del ragionatore simbolico possano garantire una spiegazione corretta del comportamento (spiegazione locale);

3. permettere la correzione e l'integrazione di tale teoria con dell'altra conoscenza (sotto forma di regole logiche) se conosciuta a priori. Tale conoscenza dovrà essere poi propagata al classificatore;

4. conseguentemente al punto precedente, fornire un meccanismo per la scoperta degli errori di apprendimento del classificatore in accordo con la conoscenza fornita dai dati d'esempio;

Fondamentalmente, grazie alla Voce 3 e alla Voce 4, si vuole ottenere un modello abbastanza flessibile da permettere, oltre che di giustificare il comportamento della componente sub-simbolica, di assistere gli utilizzatori nel processo di creazione e miglioramento della stessa. Si vuole cioè che la comprensibilità, non solo sia un valore aggiunto per gli utilizzatori finali, ma possa essere di supporto anche in fase di creazione e validazione.

3.2.2 Modello

Il modello risultante dalla valutazione dei requisiti in Sezione 3.2.1 è mostrato in Figura 3.3. Tale modello mantiene il principio di fondo dell'utilizzo di una base di conoscenza parallela al modello sub-simbolico in modo da permetterne l'analisi e la comprensione.

Prima di tutto è previsto che questa non venga compilata unicamente a partire dai dati, ma si possa procedere alla definizione iniziale di una base di conoscenza su cui basare il processo di apprendimento (Voce 3). In questo modo si andrebbe a risolvere il secondo dei problemi segnalati nella Sezione 3.1, lasciando che solo le nuove relazioni, quelle non previste a priori, vengano inferite.

La seconda particolarità risiede nella modalità con cui le regole latenti vengono desunte. È previsto l'utilizzo di un modulo con delle capacità di *Logic Program*

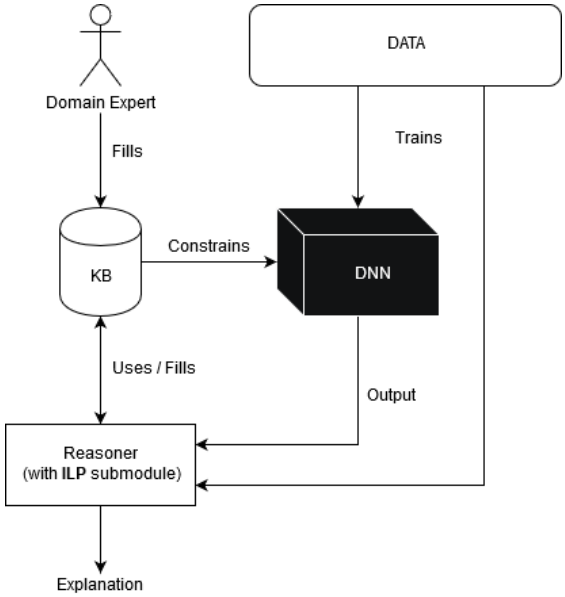

Figura 3.3: Modello proposto.

Induction che dovrebbe essere in grado di completare la KB grazie ai dati provenienti da degli esempi concreti. Tale modulo può essere utilizzata in due modalità: a partire dai dati di addestramento e a partire da quelli derivati dal classificatore stesso. La seconda dovrebbe fornire una rappresentazione comprensibile di ciò che questo ha effettivamente appreso (spiegazione globale) (Voce 1). Grazie alla prima, invece, se i dati rappresentano correttamente il dominio in esame, si dovrebbe arrivare ad avere una teoria che modelli correttamente il problema e che quindi possa essere utilizzata per analizzare l'errore nell'addestramento del classificatore (Voce 4).

Per quanto riguarda la proprietà alla Voce 2, la parte di condizionamento della rete sulla base della conoscenza simbolica viene mantenuta ma in un'ottica più ampia. Infatti, mentre in [4] questo condizionamento è mirato ad identificare e correggere i bias riscontrati nella rete, nel modello proposto si vogliono sfruttare tutte le caratteristiche positive dei metodi di *Logic as Constraint*. Infatti, anche se queste metodologie non garantiscano che la rete risultante integri completamente la conoscenza pregressa, è stato dimostrato come migliorino l'affidabilità del modello risultante potendo contare su delle informazioni potenzialmente non desumibili dai dati di training.

Il condizionamento, se utilizzata in congiunzione con l'inferenza logica, permetterebbe non solo di verificare lo stato di avanzamento nell'apprendimento, ma anche di intervenire andando ad applicare delle correzioni alle regole desunte.

Riassumendo, il modello in fig. 3.3 fornisce gli strumenti per:

1. a partire dai dati di addestramento indurre la teoria corretta;
2. recuperare la teoria che approssima al meglio il comportamento del classificatore;
3. correggere tale teoria se scorretta o integrarla con dell'altra conoscenza se conosciuta a priori;

Per esempio, riprendendo l'esempio del classificatore capace di riconoscere la tipologia di ambiente, la teoria corretta inferibile dai soli dati consisterebbe nelle regole:

```
cucina(X) :- forno(X), frigorifero(X).
   bagno(X) :- vasca(X), lavabo(X).
```

Queste regole Prolog codificano le seguenti relazioni. Se in un ambiente X sono presenti un forno e un frigorifero allora si tratta di una cucina. In caso siano presenti una vasca e un lavabo, ci si trova davanti a un bagno.

Nel caso il classificatore confondesse l'ambiente *cucina* con l'ambiente *bagno*, la teoria approssimante sarebbe:

```
bagno(X) :- forno(X), frigorifero(X).
   bagno(X) :- vasca(X), lavabo(X).
```

Cioè, anche nel caso siano presenti un forno e un frigorifero, l'ambiente X viene considerato un bagno.

A questo punto sarà possibile correggere la relazione sbagliata o aggiungere delle nuove regole, per poi procedere ad eseguire un riaddestramento del classificatore che tenga conto dei nuovi vincoli. Questo procedimento andrebbe ripetuto fino a quando la teoria corretta e quella approssimante non concordino.

3.3 Design del sistema

Partendo dal modello definito nella Sezione 3.2.2 si passa alla delineazione di un sistema basato su tali proprietà. Nel design non si vogliono fare assunzioni sulla tipologia di tecnologia su cui poi andrebbe applicata, che siano metodi di classificazione statistici o tecniche di apprendimento approfondito. Ciò che si sta cercando di ottenere è infatti un metodo agnostico rispetto alla tecnologia alla base, che con i giusti accorgimenti possa garantire dei risultati in uno dei qualsiasi campi di

applicazione. Per tale motivo in tale sezione si farà riferimento a un generico *componente per l'apprendimento* con la capacità di generare della conoscenza a partire da dei dati. Fine ultimo è quindi la modellazione di un sistema atto all'analisi e al miglioramento di tale componente.

Con un processo iterativo si vuole quindi ottenere uno strumento che, in fase di apprendimento, sia capace di:

- assimilare la conoscenza di dominio pregressa sotto forma di regole;
- assimilare la conoscenza proveniente dai dati di training;
- convertire tale conoscenza in delle regole simboliche comprensibili dall'utilizzatore;
- permettere valutazione e correzione delle regole inferite;
- imporre anche tali regole durante l'addestramento in modo da rafforzarne l'assimilazione e correggere il funzionamento del componente per l'apprendimento.

In fase di applicazione del prodotto finito sarà quindi possibile utilizzare il ragionatore per ottenere delle informazioni sulle regole, fra quelle all'interno della base di conoscenza, che hanno avuto un peso nella valutazione data.

Cercando di scomporre il sistema nelle sue componenti fondamentali, necessarie al suo corretto funzionamento, avremo:

1. un modulo adibito alla valutazione delle regole simboliche e quindi alla correzione del comportamento del *componente* sulla loro base;
2. un modulo adibito all'estrapolazione della teoria equivalente a partire dai dati, sia quelli di addestramento sia quelli prodotti dal *componente* stesso;
3. un modulo adibito alla valutazione della teoria equivalente in modo da garantire la trasparenza del procedimento.

Una volta definito il modello globale e create delle interfacce di comunicazione fra le parti coinvolte, dal punto di vista tecnologico ognuna di queste può essere analizzata e realizzata in modo distaccato. È chiaro come la scelta della tecnologia effettiva di applicazione vada ad impattare pesantemente la parte realizzativa ma, prendendo i giusti accorgimenti, non sul comportamento generale, che invece rimarrebbe valido.

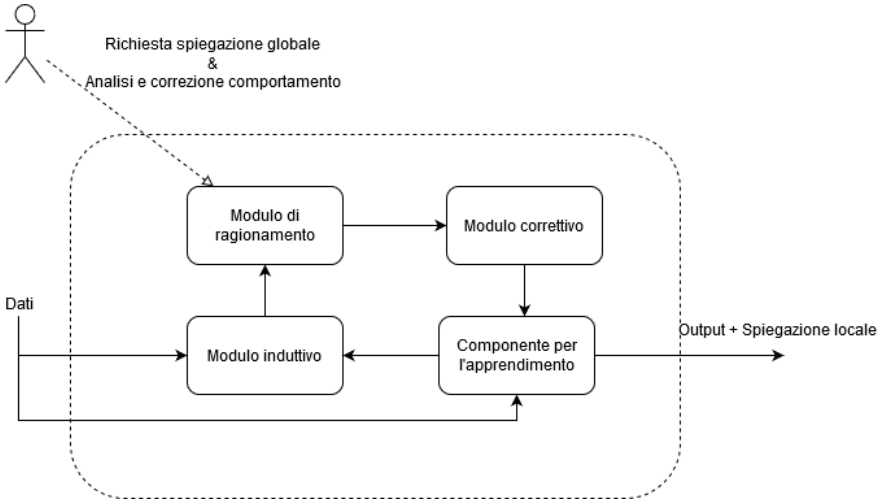

Figura 3.4: Schema di massima.

Prendendo per esempio il caso di una rete neurale che lavora su dei dati non strutturati (audio, immagini, video) si dovrebbe considerare una parte di pre-elaborazione dei dati capaci di estrarre a partire da questi delle informazioni strutturate. In questo modo si otterrebbe la compatibilità con i metodi di ragionamento simbolico fondamentali per il funzionamento del sistema. Naturalmente, il problema non si porrebbe nel caso di dati già strutturati. Infatti, in questo caso si renderebbe necessario solamente l'adattamento di tali dati agli standard richiesti dal sistema, che potrebbe essere per esempio la formulazione degli stessi come fatti di una teoria logica, etc. In quest'ottica assumono particolare importanza le interfacce di comunicazione fra i vari moduli e la loro standardizzazione.

Nel Capitolo 4 si procederà con la realizzazione di un piccolo prototipo che, sfruttando le tecnologie maggiormente all'avanguardia nel campo della computazione neuro-simbolica, serva a dimostrare l'effettiva valenza del modello sopra descritto.

Capitolo 4

Studio di fattibilità

Prima di iniziare con la delineazione del prototipo e delle sue caratteristiche viene data una rapida sintesi degli obiettivi e dei vincoli che ne hanno guidato la realizzazione, cercando di giustificare in tal modo le decisioni prese. Detto questo, il primo punto da affrontare nella definizione dei requisiti è l'area in cui si dovrà inserire tale prototipo. Nella Sezione 3.3 si è fatto notare come teoricamente il modello proposto sia applicabile a diverse tipologie di tecniche, dagli algoritmi di machine learning più tradizionali alle reti neurali. Si è altresì detto come però ognuna di queste tecniche richieda un diverso sforzo, da un punto di vista puramente implementativo, per la sua integrazione. Per tale motivo, essendo scopo di questo prototipo non il costruire un sistema effettivamente utilizzabile, ma fornire una conferma della validità del metodo presentato, le scelte fatte sono state volte a fornire un sistema il più esemplificativo possibile delle proprietà ricercate. Questo anche a scapito della complessità e completezza del prototipo stesso.

Come target dell'esperimento sono stati scelti i sistemi di reti neurali artificiali. Infatti, fra tutte, queste sono quelle che di più soffrono del problema della poca trasparenza nel loro funzionamento e quindi gioverebbero maggiormente di un avanzamento nel campo dell'Explainability. Si è scelto però di trattare dei problemi estremamente semplici, con un grado di complessità di molto inferiore a quelli che normalmente sono oggetto di tale categoria di tecniche, ma che presentano delle problematiche note in letteratura (dati sbilanciati, elevata dimensionalità, etc). Questi problemi sarebbero stati risolvibili facilmente anche con degli algoritmi di machine learning più tradizionali (decision tree, KNN, etc). Tale scelta è stata fatta perché il grado di difficoltà minore permette di avere più controllo sulla conformazione del problema da risolvere, e quindi, di avere un riscontro maggiore sulla reale efficacia delle soluzioni adottate. Una volta verificato il funzionamento del modello, utilizzando i giusti accorgimenti, sarà verosimilmente possibile scalare la soluzione ottenuta a una classe di problemi con complessità reale.

4.1 Realizzazione del prototipo

Come visto nella Sezione 3.3 le componenti principali necessarie alla realizzazione, e quindi al funzionamento, del prototipo sono:

1. un modulo adibito alla valutazione delle regole simboliche e quindi alla correzione del comportamento della rete neurale;
2. un modulo adibito all'estrapolazione della teoria equivalente a partire dai dati, sia quelli di addestramento sia quelli prodotti dalla rete;
3. un modulo adibito alla valutazione della teoria equivalente.

Come si è potuto constatare nella Sezione 2.1, il panorama della *NSC* è molto vasto e comprende tantissimi lavori in ambiti anche molto diversi. Fra queste diverse categorie di soluzioni sono da ricercare delle metodologie che possano andare a soddisfare le necessità del modello sopra proposto.

4.1.1 Tecnologie

Per quanto riguarda il punto 1 sono state valutate principalmente due alternative: quella proposta in [16] (DL2) e quella proposta in [18] (Augmentation). La prima appartiene alla categoria di tecniche fondate sulla modifica della funzione di perdita (loss function) sulla base dei vincoli imposti. Per riassumere nuovamente, le formule simboliche vengono convertite in una loro equivalente differenziabile, che viene utilizzata per modificare l'andamento della loss function. Grazie a tale modifica si ottiene una modifica del processo di apprendimento mirata a favorire l'assimilazione dei vincoli imposti. La seconda, invece, si basa sull'idea di modificare la struttura della rete stessa, andando a definirne una parte affinché mimi il comportamento dei vincoli sui dati. In pratica si vuole che i neuroni che la compongono si attivino se il vincolo è soddisfatto. La scelta è ricaduta su DL2 a causa di due motivazioni principali. Prima di tutto, l'idea su cui si basa tale lavoro è probabilmente la più diffusa in tale categoria e per questo risulta essere la più solida e provata. L'altra, invece, utilizzando un metodo molto più recente e sperimentale, non possiede ancora gli stessi riscontri nei risultati. La seconda motivazione che ha guidato la scelta è la semplicità del metodo in sé. Infatti, i principi su cui si basa la traduzione delle regole simboliche in formato differenziabile sono essenzialmente semplici e poco invasivi rispetto al normale funzionamento di una rete neurale. Risulta quindi semplice adattarli alle proprie esigenze o estendere il meccanismo di condizionamento, come effettivamente è stato fatto.

Per quanto riguarda il punto 2, anche in questo caso, sono diversi i metodi che mirano a risolvere il problema della *Logic Program Induction*. Fra i diversi approcci quello più interessante è risultato essere quello proposto nel lavoro [27] e [9],

appartenente invece ad un'altra delle categorie principali nel panorama della NSC, il *Differentiable Programming*. Il framework proposto (NTP) ha come obiettivo base quello di creare una modalità differenziabili di valutazione di affermazioni logiche. Riassumendo, si cerca di eseguire un mapping fra regole e fatti in formato simbolico in uno spazio numerico n-dimensionale. La veridicità di un'affermazione è basata sulla distanza dei fatti che la compongono in tale spazio. Naturalmente questo framework presenta anche un'altra funzionalità fondamentale, indispensabile per il task al punto 2: la capacità di inferire delle relazioni latenti fra le istanze della teoria esaminata. Anche se questo risulta essere il punto di interesse principale, anche altre caratteristiche, fra cui la capacità di fare dei ragionamenti in termini di similarità, e l'utilizzo di una sintassi Prolog come standard di definizione delle teorie logiche, si sono rivelate determinanti nell'orientare la scelta verso tale tecnologia.

Per quanto riguarda il punto 3 invece si è scelto di adottare un tradizionale motore Prolog per la valutazione e analisi delle teorie logiche. Nello specifico è stato utilizzato il motore tuProlog, un framework leggero e open-source con tutte le caratteristiche fondamentali necessarie a questa parte del sistema, come l'esplorazione e la visualizzazione dell'albero di derivazione. Sarebbe stato interessante sperimentare anche per questo modulo l'utilizzo del framework *NTP*. Questo motore Prolog non avrebbe portato dei vantaggi solamente da un punto di vista delle performance, ma anche ad una maggiore flessibilità del meccanismo di unificazione rispetto ai framework simbolici tradizionali. Infatti, grazie al suo modello di ragionamento per similarità, non è necessario che due termini siano esattamente identici per essere unificati, ma piuttosto che siano vicini nello spazio vettoriale di appartenenza. In pratica si da più importanza alla semantica di un termine che alla forma con cui è espresso. La maturità di questa tecnologia non è ancora tale però da poterla porre come alternativa seria ai motori completamente simbolici, e per tale motivo è stata scartata.

4.1.2 Funzionamento

Dopo aver definito finalità, requisiti e strumenti tecnologici disponibili non resta che passare al dettaglio di come il prototipo in esame è stato realizzato e di quali siano le sue caratteristiche e funzionalità.

Per i dettagli puramente implementativi si rimanda al codice sorgente[1]. In tale sede ci si limiterà ad elencare quali funzionalità sono state effettivamente realizzate e come le tecnologie elencate nella Sezione 4.1.1 siano state utilizzate.

Per la parte di addestramento/condizionamento sono presenti tre modalità di funzionamento:

[1] https://github.com/Gilbocc/NSC4ExplainableAI

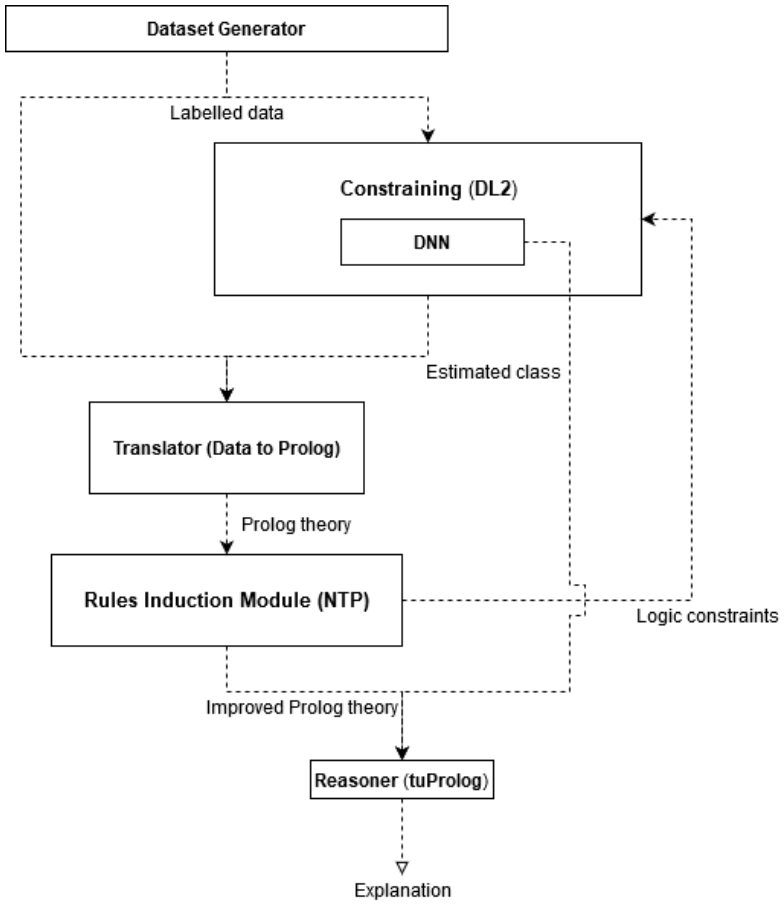

Figura 4.1: Schema tecnologico del sistema.

1. *standard*, in cui nessun condizionamento viene applicato. I risultati ottenuti in tal modo possono essere utilizzati come baseline;

2. *condizionamento locale*, in tale modalità viene aggiustato il valore restituito dalla funzione di perdita sulla base della valutazione sul training set dei vincoli simbolici imposti. L'influsso di tale condizionamento è pesato sulla base di un peso W preso come parametro del modello;

3. *condizionamento globale*, in tale modalità le relazioni simboliche vengono sfruttate per la generazione di esempi che possano aiutare a correggere il comportamento della rete nel caso non si attenga al vincolo imposto. In pratica, a seguito della valutazione della loss function legata alle regole simboliche, vengono selezionate le istanze del training set la cui classificazione ha portato a una violazione di tali regole. A questo punto le regole guidano la generazione di un nuovo batch di esempi che si trovano nell'intorno di quello inesatto. La rete viene così addestrata utilizzando questi dati sintetici in modo da aggiustarne il comportamento. Intuitivamente, si vuole cercare di imporre il vincolo grazie ad una serie di esempi che lo supportino.

Come detto sopra, come base per il modulo di condizionamento si è fatto uso del lavoro [16].

Il modello ottenuto a seguito della fase di addestramento viene salvato e dato in input ad un componente incaricato di effettuarne la valutazione. I dati utilizzati per la valutazione, insieme al responso della rete, vengono utilizzati per la creazione di una teoria Prolog equivalente. In pratica questo componente funge da connettore fra la componente sub-simbolica del sistema (classificatore) e quella simbolica (reasoner, motore di induzione).

La teoria ottenuta viene poi passata al sistema di inferenza che cerca di estrapolarne le relazioni latenti. Questa parte è, come già visto, basata sul lavoro [9].

Teoria e relazioni inferite insieme formano la teoria equivalente alla rete neurale. Attraverso il motore tuProlog è possibile esplorare tale teoria e ottenere così le giustificazioni al comportamento della rete.

Per ottenere la teoria di riferimento si applica la stessa pipeline di conversione in Prolog e arricchimento a partire, però, dai dati di training.

4.2 Generazione dei dati

I dati sono uno degli aspetti fondamentali di qualsiasi algoritmo di apprendimento automatico. La loro natura, forma, dimensionalità, etc. influiscono ampiamente sulla valutazione della tecniche da utilizzare e quindi sulla buona riuscita del task

in oggetto. Essendo il prototipo in esame costruito per funzionare con delle tecniche di *apprendimento approfondito*, sarebbe stato naturale pensare a dei dati non strutturati (immagini, audio, video, etc.) come dominio degli esperimenti effettuati. L'utilizzo di tale tipologia di dati avrebbe però comportato un aumento non indifferente nella complessità degli strumenti necessari alla realizzazione del prototipo stesso, senza apportare nessun reale beneficio dal punto di vista dello scopo principale di tale lavoro: dimostrare l'effettiva efficacia della metodologia adottata. Per tale motivo si è scelto di utilizzare tali tecniche con dei dataset completamente strutturati, per i quali normalmente verrebbero utilizzati degli algoritmi di machine learning più tradizionali. Si è inoltre scelto di non utilizzare dei set di dati reali ma di crearne di appositi. In tal modo è infatti possibile mantenere il più assoluto controllo sulla forma dei dati e sulle logiche che ne governano l'andamento.

In un contesto come quello di tali esperimenti, dove l'obiettivo principale è il riuscire a verificare come il processo di apprendimento segua una certa logica e poter così esplorare il perché di certi risultati, l'assoluto controllo sulle regole che governano la distribuzione dei dati e sulla struttura che essi hanno si sono rivelate proprietà imprescindibili. È infatti stato possibile generare dei dataset con delle proprietà specifiche, che permettessero di testare il prototipo con la massima trasparenza e controllo.

Da un punto di vista puramente pratico, la procedura seguita per la generazione dei dataset è la seguente:

1. definizione del numero di attributi e della loro tipologia;
2. definizione delle relazioni logiche su cui si basa la classificazione di tali dati;
3. generazione randomica delle istanze del dataset;
4. assegnazione della classe di appartenenza utilizzando le logiche di cui sopra.

Capitolo 5

Analisi dei risultati

Alla luce di quanto detto fino ad ora, si procede ad elencare dettagliatamente quali sono stati gli esperimenti effettuati con il prototipo in esame. Come anticipato ci si è concentrati su dei problemi noti in letteratura che solitamente minano il processo di apprendimento automatico. In casi come questi, un framework come quello proposto, capace non solo di dare un sostegno all'analisi e comprensione di una rete neurale, ma anche al suo miglioramento, potrebbero rivelarsi di fondamentale importanza.

Per ogni caso di studio si presenta il problema alla sua base, la forma e le logiche del dataset generato ad hoc, la forma della teoria logica associata e il risultato desiderato. I risultati effettivamente ottenuti verranno esposti e discussi nella Sezione 5.3.

5.1 Sbilanciameto del dataset

La prima categoria di problemi che si è cercato di trattare è quella legata all'addestramento in presenza di dati sbilanciati. Con questo termine vengono identificati i casi in cui il numero di istanze di una classe risulta nettamente minora rispetto a quelle delle altre classi all'interno del training set. Questo solitamente causa una cattiva approssimazione del dominio da parte della rete, che tendenzialmente ignorerà la porzione con pochi esempi.

Solitamente, le tecniche utilizzate per risolvere tale tipologia di problemi sono legate principalmente all'utilizzo di tecniche di pre-elaborazione sul dataset (rebalancing, sampling, etc.). Questo lavoro viene effettuato da parte di esperti che, dopo un attento studio del problema e dei dati, elaborano una strategia di pulizia e miglioramento delle istanze di addestramento.

Quello che si vuole ottenere è una sorta di miglioramento/automatizzazione di questo processo. Grazie alla parte di inferenza logica sui dati di training dovrebbe

essere possibile ottenerne le logiche contenute all'interno. Dall'esplorazione di tale e teoria e dal confronto con quella ottenuta effettuando il lavoro di inferenza sui dati prodotti dalla rete stessa (si ricorda come l'esplorazione sia facilitata dall'utilizzo di un linguaggio come Prolog), sarà possibile identificare le lacune nel comportamento della stessa, e quindi, andare a rafforzarle attraverso la componente di condizionamento (globale o locale).

Formato dei dati. Il dataset generato ha una struttura del tipo *["Name", "Surname", "Class"]*, dove Name e Surname sono due attributi categorici e Class è la categorizzazione assegnata. Il numero di istanze totali del dataset è di 10000 elementi. Di questi, 1000 hanno come valore di Name *Achille*, mentre i restanti hanno tale attributo, con una pari probabilità, uguale a *Zenio* o *Pino*. L'attributo Surname conterrà invece una stringa generata casualmente, e quindi differente per ogni elemento del dataset.

La categoria, con dominio in $[0 - 1]$, è assegnata con la seguente logica: se l'attributo Name è uguale a *Achille* allora la classe sarà 1, 0 altrimenti.

Teoria Prolog. Da ogni elemento del dataset viene estratta una rappresentazione equivalente in linguaggio Prolog. Facendo un esempio, a partire dalla tupla *(Achille, Orual, 1)* si otterrà:

```
class(1).
name(achille).
surname(orual).
nameWithClass(achille,1).
surnameWithClass(orual,1).
person(achille,orual).
classification(pino,orual,1).
```

In pratica si vanno ad inserire nella *KB* una serie di fatti che mantengano le informazioni sull'elemento con diversi gradi di specificità.

Desiderata. Date le poche istanze con valore *Achille* per l'attributo Name ci si aspetta che queste non siano sufficienti a fornire un esempio valido da cui il modello possa apprendere. Questo anche a causa della variabilità dell'attributo Surname. Si vuole quindi verificare se grazie al prototipo creato sia possibile:

1. recuperare la relazione corretta a partire dai dati di training. Con corretta si intende una regola che rappresenti la correlazione fra la classificazione e il nome;

2. recuperare la teoria equivalente dal modello errato in moda da poterlo analizzare;

3. correggere il comportamento del modello grazie al meccanismo di condizionamento e alla conoscenza ottenuta al punto 1.

5.2 Dataset ad elevata dimensionalità

L'addestramento di modelli in presenza di dataset con un alto numero di attributi è uno degli altri problemi noti nell'ambito dell'addestramento automatico. Ci si riferisce a tale problema con la frase *Curse of dimensionality* [3]. Coniata inizialmente per esprimere la difficoltà nell'ottimizzazione di una funzione con un elevato numero di parametri, questa frase è stata mutuata nel campo dell'apprendimento con un diverso significato. Piuttosto che ad un problema di ottimizzazione si fa riferimento a due problematiche principali nel caso di dataset ad elevata dimensionalità. Il primo, che si presenta nel caso il numero di attributi sia molto alto e quello di istanze di training sia non adeguato, è di creare un modello che sovradatti i dati, e che quindi non scali su delle istanze non viste in fase di addestramento. Il secondo è più legato ad un punto di vista tecnico. Gli algoritmi di apprendimento solitamente ragionano in termini di similarità dei dati in uno spazio n-dimensionale. In pratica le istanze di una stessa classe risulteranno più vicine in tale spazio, dando modo così di isolare la regione legata ad una certa categoria e quindi poterla associare anche a dei dati non visti in fase di addestramento. All'aumentare del numero di dimensioni la distanza fra le singole istanze aumenta vertiginosamente rendendo difficile effettuare questa sorta di clustering alla base del processo di classificazione. Anche in questo caso sono state sviluppate delle tecniche apposite per cercare di diminuire la dimensionalità dei dati e quindi risolvere alla radice tale problema.

Finalità di questo esperimento è verificare se la metodologia proposta possa essere di ausilio anche per tale categoria di problemi.

Formato dei dati. Il dataset generato contiene trenta attributi binari (con valore 0 o 1) numerati da 1 a 30, più una colonna Class contenente la categorizzazione. Il numero di istanze totali del dataset è di 1000 elementi. Tutti i valori dei 30 attributi sono stati assegnati completamente casualmente.

La categoria, con dominio in $[0 - 1]$, è assegnata con la seguente logica: se l'attributo 10 è pari a 1 allora la classe sarà 1, 0 altrimenti.

Teoria Prolog. La *KB* è popolata con delle regole pregresse della forma:

```
ok(true).
has1(X1,...,XN) :- ok(X1).
notOk(false).
hasNot1(X1,...,XN) :- notOk(X1).
```

Queste regole, presenti per ognuno dei 30 attributi, sono necessari a fornire gli strumenti logici per verificare se un attributo X1, data in input la lista degli attributi X1,..,XN, abbia valore positivo o negativo.

Da ogni elemento del dataset viene estratta una rappresentazione equivalente in linguaggio Prolog e aggiunta alla KB di partenza. Da un elemento con attributi $X1, ..., XN$ di classe 0 si avrà:

```
isA(X1,...,XN).
```

In caso di classe 1 invece:

```
isB(X1,...,XN).
```

Desiderata. Anche in questo caso gli obiettivi sono riconducibili a quelli visti nell'esperimento in Sezione 5.1.

5.3 Risultati

Si procede ora a mostrare i risultati ottenuti nei due esperimenti riportati nella Sezione 5.1 e nella Sezione 5.2. Entrambe le configurazioni sono state testate con le stesse modalità, cercando di valutarne le performance nei seguenti task:

1. recupero delle teoria ottima a partire dai dati di addestramento (Sezione 5.3.1);

2. addestramento della rete neurale e recupero della teoria equivalente (Sezione 5.3.2);

3. analisi e correzione della teoria equivalente, con conseguente riaddestramento della rete neurale (Sezione 5.3.3). A questo scopo sono state valutate sia le modalità di condizionamento globali, che quelle locali.

5.3.1 Recupero teoria ottima

Il primo test effettuato su entrambi i casi di studio è la generazione di una teoria logica a partire dai dati etichettati utilizzati per l'addestramento. Si è già detto come, teoricamente, se la tecnica di inferenza fosse ottima si otterrebbe un insieme di regole logiche *ottime*. Verrebbero cioè alla luce le relazioni e i vincoli che hanno portato al manifestarsi di tali occorrenze.

Naturalmente però il processo di inferenza è altamente complesso e tutt'altro che vicino al concetto di *ottimo*. Lo spazio delle possibilità è infatti così vasto da non permettere, in tempi computazionali accettabili, di effettuare una ricerca esaustiva delle corrette relazioni latenti. Per tale ragione si è deciso di testare questo aspetto: il modello proposto è fondato sulla componente di inferenza. È importante constatare se tale via, anche con gli strumenti più all'avanguardia e con dei casi di studio aventi una ridotta complessità, risulti ancora impercorribile.

Per quanto riguarda il caso di studio in Sezione 5.2 l'obiettivo è, come visto, l'ottenimento di una regola che esprima la correlazione fra il nome e la classificazione. Effettivamente, partendo dai dati corretti, il framework *NTP* è riuscito a produrre una formula nella forma:

```
classification(X,Y,Z) :- nameWithClass(X,Z).
```

In pratica, se interrogata sulla classificazione Z di un individuo con nome X e cognome Y, il motore logico risponderà verificando la presenza nella KB di un individuo con lo stesso nome. Questo risulta in accordo con la logica di creazione del dataset, basata appunto sul nome assegnato all'individuo. In un'ottica in cui i dati siano tutti corretti, questa strategia risulta ottima anche per assegnare una classificazione agli individui non apparsi fra quelli d'esempio. È necessario solamente che il nome sia nell'insieme noto.

La stessa fortuna non si è avuta con il secondo dei casi di studio, quello legato alla dimensionalità. La regola risultante ha assunto infatti la forma:

```
has1(X1,...,XN) :- has1(X1,...,XN).
```

Il risultato del processo di inferenza, per quanto corretto, non ha nessun reale valore. Si tratta infatti di una tautologia. Si limita cioè ad esprimere questo concetto: se la tupla $(X1, ..., XN)$ ha l'attributo 1 vero, allora l'attributo 1 di tale tupla è vero.

I problemi nell'inferenza della logica è sicuramente da ricondursi allo stesso problema che si cercava di analizzare in partenza: quello della dimensionalità. Infatti il funzionamento di *NTP* è basato sul principio di distanza in un dominio multidimensionale. Purtroppo questo, come visto nella Sezione 5.2, è un problema noto quando si utilizzano dei dataset ad elevata dimensionalità. Il risultato risulta essere quindi in accordo con le aspettative.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1979
1	0.00	0.00	0.00	21
accuracy			0.99	2000
macro avg	0.49	0.50	0.50	2000
weighted avg	0.98	0.99	0.98	2000

Tabella 5.1: Risultati esperimento sbilanciamento senza condizionamento

	precision	recall	f1-score	support
0	0.68	0.86	0.76	106
1	0.78	0.55	0.65	94
accuracy			0.71	200
macro avg	0.73	0.71	0.70	200
weighted avg	0.73	0.71	0.71	200

Tabella 5.2: Risultati esperimento dimensionalità senza condizionamento

5.3.2 Recupero teoria equivalente

Si esaminano ora i risultati dell'addestramento delle reti a partire dai soli dati d'esempio. Per quanto riguarda il primo dei casi di studio si procede anche al tentativo di estrazione della teoria equivalente a partire dalle valutazioni della rete. Per quanto riguarda il secondo caso di studio, vista l'inattuabilità del meccanismo di inferenza in Sezione 5.3.1, questa parte è omessa. I test sono stati eseguiti utilizzando il 20% dei dati disponibili come insieme di test. Su tutti gli altri dettagli implementativi, quali struttura della rete, funzione di loss utilizzata, etc., si rimanda al repository GitHub[1].

Nel caso del primo caso di studio, i risultati dell'addestramento (50 epoche) sono mostrati in Tabella 5.1. I risultati dell'addestramento (100 epoche) nel caso del secondo caso di studio sono mostrati in Tabella 5.2.

Come si può notare, in entrambi i casi sono riscontrabili le criticità legate ai problemi affrontati. Nel caso del dataset fortemente sbilanciato, il modello risultante risulta anch'esso sbilanciato verso la classe predominante. Infatti benché

[1]https://github.com/Gilbocc/NSC4ExplainableAI

l'accuratezza del 98% sembri suggerire un comportamento corretto, guardando i dati di *precision* e *recall* ci si accorge di come il modello stimi tutti gli elementi come appartenenti alla classe predominante. In questo modo l'errore si diminuisce, ma solamente perché gli elementi mal classificati sono in scarsissimo numero.

Anche nel caso del secondo dei problemi affrontati si può notare nel modello risultante un comportamento decisamente mediocre. Infatti presenta un accuratezza media di circa l'80% (calcolata su 5 addestramenti differenti). Questo valore, seppure a prima vista accettabile, risulta comunque eccessivamente basso una volta considerati sia la durata dell'addestramento (100 epoche), che la semplicità del dominio. Si ricorda infatti che l'unico discriminante per una corretta classificazione è il valore del decimo attributo: se questo è a 1 la classe associata è 1, se è 0 la classe è anch'essa 0. Questo esempio mostra in modo lampante come la dimensionalità sia un aspetto decisamente problematico nell'addestramento di reti neurali.

Per quanto riguarda la teoria associata al primo modello, il processo di estrazione ha prodotto solamente regole con una accuratezza estremamente bassa. Tale accuratezza è calcolata da NTP sulla base di quanto la regola inferita sia in accordo con i fatti della teoria base. Tale mancanza nel dare un responso netto mostra quindi un'assenza di logica nel comportamento del classificatore, almeno secondo l'impostazione data alla teoria estratta dai responsi. Si ricorda infatti che questo si limita infatti a dare la stessa valutazione (0) ad ogni istanza considerata.

5.3.3 Correzione e Condizionamento

Una volta ottenute la teoria ottima (Sezione 5.3.1) e la quella equivalente al modello (Sezione 5.3.2), seguendo il flusso descritto nella Sezione 3.2.2, si può procedere con la parte di analisi e correzione dello stesso. Si è detto infatti come, grazie alla valutazione della due teorie di cui sopra, sia possibile analizzare il comportamento della rete e identificarne le lacune. Tradotto nei termini dei due problemi affrontati in questo capitolo, si deve ricercare una modalità per correggere i problemi di sbilanciamento e dimensionalità.

Nel primo caso, tale correzione è ricercata grazie all'iniezione del fatto:

```
classification(achille, X, 1).
```

Il significato è molto semplice: qualsiasi persona con nome *Achille*, indipendentemente dal cognome, deve avere classificazione 1.

Il condizionamento è stato testato utilizzando entrambe le modalità del prototipo realizzato: globale e locale. Nel primo caso, si è ottenuta un'accuratezza media (su 5 addestramenti del modello) pari al 99.53%. Anche il condizionamento globale ha ottenuto degli ottimi risultati: l'accuratezza media è del 99.28%.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1980
1	1.00	0.60	0.75	20
accuracy			1.00	2000
macro avg	1.00	0.80	0.87	2000
weighted avg	1.00	1.00	1.00	2000

Tabella 5.3: Risultati esperimento sbilanciamento con condizionamento locale

	precision	recall	f1-score	support
0	0.99	1.00	1.00	1980
1	1.00	0.50	0.67	20
accuracy			0.99	2000
macro avg	1.00	0.75	0.83	2000
weighted avg	1.00	0.99	0.99	2000

Tabella 5.4: Risultati esperimento sbilanciamento con condizionamento globale

	precision	recall	f1-score	support
0	0.97	0.29	0.44	105
1	0.56	0.99	0.71	95
accuracy			0.62	200
macro avg	0.76	0.64	0.58	200
weighted avg	0.77	0.62	0.57	200

Tabella 5.5: Risultati esperimento dimensionalità con condizionamento locale

I dettagli di una di queste esecuzioni per entrambe le modalità di condizionamento si possono trovare rispettivamente nella Tabella 5.3 per quello locale e nella Tabella 5.4 per quello globale. Mettendo da parte il valore di accuratezza, già alto in partenza ma inefficace nel valutare le reale capacità della rete, si può notare come il modello, in entrambi i casi, sia migliorato di parecchio nell'individuazione delle istanze della classe 1.

Per il problema legato alla dimensionalità, invece, la relazione iniettata è:

```
isB(X1,...,XN) :- has10(X1,...,XN).
```

In altre parole, se un'istanza ha l'attributo 10 a 1, allora può essere attribuita la classe 1.

Anche in questo caso il condizionamento ha portato a degli ottimi risultati. Però, a differenza del caso precedente, il solo condizionamento locale (Tabella 5.5) non ha sortito gli stessi effetti di quello globale che ha ottenuto il 93% di accuratezza media (Tabella 5.6). Il secondo metodo si è rivelato maggiormente efficace proprio grazie alla sua modalità di funzionamento. Infatti, generando degli esempi nell'intorno delle istanze che non seguono la logica imposta, va a porre rimedio al problema della dimensionalità nella maniera più semplice: provvedendo un insieme maggiore di dati.

Riassumendo, si può comunque dire che, in entrambi i problemi, il condizionamento ha portato ad un netto aumento delle performance della rete.

5.3.4 Valutazioni globali

Nella Sezione 3.3 sono proposti cinque punti che esprimono le proprietà ricercate nel sistema in fase di studio. I test, realizzati proprio per poter valutare l'effettiva valenza di tali punti, hanno portato alle seguenti valutazioni:

	precision	recall	f1-score	support
0	0.97	0.66	0.78	105
1	0.72	0.98	0.83	95
accuracy			0.81	200
macro avg	0.85	0.82	0.81	200
weighted avg	0.85	0.81	0.81	200

Tabella 5.6: Risultati esperimento dimensionalità con condizionamento globale

- *assimilazione della conoscenza di dominio pregressa sotto forma di regole.* Questo compito è stato correttamente portato a buon fine. Si pensi per esempio a:

```
classification(achille, X, 1).
isB(X1,...,XN) :- has10(X1,...,XN).
```

  Tali regole, utilizzate nella Sezione 5.3.3 per migliorare il comportamento del modello, derivano completamente dalla conoscenza di dominio pregressa, e non dai dati di addestramento.

- *assimilare la conoscenza proveniente dai dati di traini.* Tale comportamento è garantito dall'utilizzo della tecniche di apprendimento automatico (in questo caso delle reti neurali).

- *conversione della conoscenza presente all'interno del modello di machine learning in regole simboliche.* Questo punto si è rivelato possibile nel caso di un problema semplice (primo caso di studio), dove è stata inferita correttamente la correlazione fra la classificazione e il nome. Nel secondo caso, invece, la complessità del problema non ha permesso di arrivare a delle regole con una qualche rilevanza.

- *permettere valutazione e correzione delle regole inferite.* Nel caso di studio legato alla dimensionalità le regole inferite non hanno portato a nessun beneficio. Infatti la relazione che ha portato effettivamente ad un miglioramento deriva completamente da delle informazioni date dall'utente. Nel primo caso invece, dall'esplorazione della teoria ottima inferita, e dal confronto con quella equivalente, si è potuto arrivare in modo semplice alla relazione chiave. Questo non può che essere un fatto a sostegno di come la valutazione di una

rete sotto forma di regole logiche possa essere sicuramente uno strumento molto utile ai fini della sua comprensione e del suo miglioramento.

- *imposizione di regole durante l'addestramento* Anche questo punto ha dato degli ottimi risultati (Sezione 5.3.3), dimostrandone l'effettiva attuabilità.

In definitiva, sebbene le tecniche mostrino in alcuni casi di essere ancora acerbe, i risultati ottenuti fanno sperare come questa sia una via percorribile per arrivare ad una piena trasparenza dei motori sub-simbolici.

Capitolo 6

Conclusioni

In questo lavoro si è visto come i lavori afferenti all'area della *NSC* possano costituire un'ottima base per la costruzione di un sistema ibrido votato alla trasparenza e alla comprensione della parte sub-simbolica. Partendo dall'esplorazione dei lavori appartenenti allo stato dell'arte e dalle metodologie al momento oggetto di ricerca (Capitolo 2), si è analizzata una possibile via per la creazione di sistemi comprensibili (Capitolo 3). Sebbene il metodo possa sembrare teoricamente valido, la creazione di un prototipo (Capitolo 3) ha permesso di mettere in luce quali sono i limiti realizzativi dati dalla tecnologia al momento disponibile. Infatti, per quanto la parte di condizionamento, necessaria ad allineare il comportamento della rete neuronale alla teoria logica che la descrive, si sia dimostrata estremamente valida, così non è stato per la componente di induzione logica. Questa parte, centrale per la realizzazione del modello proposto, sebbene realizzata con una delle tecniche più promettenti, ha dimostrato subito i suoi limiti all'aumentare della complessità dei dati. È stato però interessante notare come, in un caso di studio più semplice (quello legato allo sbilanciamento dei dati), il comportamento globale si sia dimostrato all'altezza delle aspettative. Questo dimostra che, anche se al momento esistono dei forti limiti dal punto di vista tecnologico, la strada intrapresa possa portare a dei progressi nella ricerca della *XAI*.

Per quanto riguarda i lavori futuri, sarebbe interessante mettere alla prova il modello con un caso di studio reale. Sarebbe inoltre prioritaria la ricerca di un metodo alternativo mirato all'estrazione della logica a partire dai dati.

Bibliografia

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019.

[2] Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration - a structured survey, 2005.

[3] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[4] Adrien Bennetot, Jean-Luc Laurent, Raja Chatila, and Natalia Díaz-Rodríguez. Towards explainable neural-symbolic visual reasoning, 2019.

[5] William Byrd. Relational programming in minikanren: Techniques, applications, and implementations. 09 2009.

[6] Nuri Cingillioglu and Alessandra Russo. Deeplogic: Towards end-to-end differentiable logical reasoning, 2018.

[7] William W. Cohen, Fan Yang, and Kathryn Rivard Mazaitis. Tensorlog: Deep learning meets probabilistic dbs, 2017.

[8] Artur d'Avila Garcez, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning, 2019.

[9] Michiel de Jong and Fei Sha. Neural theorem provers do not learn rules without exploration, 2019.

[10] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2468–2473, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[11] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings, 2016.

[12] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel rahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy i/o, 2017.

[13] Ivan Donadello, Luciano Serafini, and Artur d'Avila Garcez. Logic tensor networks for semantic image interpretation, 2017.

[14] Derek Doran, Sarah Schulz, and Tarek R. Besold. What does explainable ai really mean? a new conceptualization of perspectives, 2017.

[15] Anton Dries, Angelika Kimmig, Wannes Meert, Joris Renkens, Guy Van den Broeck, Jonas Vlasselaer, and Luc De Raedt. Problog2: Probabilistic logic programming. In Albert Bifet, Michael May, Bianca Zadrozny, Ricard Gavalda, Dino Pedreschi, Francesco Bonchi, Jaime Cardoso, and Myra Spiliopoulou, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 312–315, Cham, 2015. Springer International Publishing.

[16] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1931–1941, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[17] Ioannis Hatzilygeroudis and Jim Prentzas. Neuro-symbolic approaches for knowledge representation in expert systems. *Int. J. Hybrid Intell. Syst.*, 1:111–126, 01 2004.

[18] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic, 2019.

[19] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming, 2018.

[20] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[21] Emilio Parisotto, Abdel rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis, 2016.

[22] Thomas Pierrot, Guillaume Ligner, Scott Reed, Olivier Sigaud, Nicolas Perrin, Alexandre Laterre, David Kas, Karim Beguir, and Nando de Freitas. Learning compositional neural programs with recursive tree search and planning, 2019.

[23] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence: A Logical Approach*. 01 1998.

[24] Alexey Potapov, Anatoly Belikov, Vitaly Bogdanov, and Alexander Scherbatiy. Differentiable probabilistic logic networks, 2019.

[25] Scott Reed and Nando de Freitas. Neural programmer-interpreters, 2015.

[26] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[27] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving, 2017.

[28] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[29] Tian Sang, Paul Beame, and Henry Kautz. Performing bayesian inference by weighted model counting. volume 1, pages 475–482, 01 2005.

[30] Luciano Serafini and Artur S. d'Avila Garcez. Learning and reasoning with logic tensor networks. In Giovanni Adorni, Stefano Cagnoni, Marco Gori, and Marco Maratea, editors, *AI*IA 2016 Advances in Artificial Intelligence*, pages 334–348, Cham, 2016. Springer International Publishing.

[31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis.

Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

[32] A. M. Turing. Intelligent machinery. Report, 1948.

[33] Andre Vellino. Artificial intelligence: The very idea: J. haugeland, (mit press, cambridge, ma, 1985); 287 pp. *Artificial Intelligence*, 29:349–353, 09 1986.

[34] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge, 2017.

[35] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Exploring artificial intelligence in the new millennium. chapter Understanding Belief Propagation and Its Generalizations, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[36] Lisa Zhang, Gregory Rosenblatt, Ethan Fetaya, Renjie Liao, William E. Byrd, Matthew Might, Raquel Urtasun, and Richard Zemel. Neural guided constraint logic programming for program synthesis, 2018.