UNIVERSITA' DEGLI STUDI DI BARI

CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA

IN

INGEGNERIA DELLA CONOSCENZA

APPLICAZIONE DI EM A UN MODELLO DI CLASSIFICAZIONE PROBABILISTICO PER GRAFI DI CONOSCENZA

Relatori:

Prof. Nicola Fanizzi

Prof.ssa Claudia d'Amato

Laureando:

Christian Riefolo

ANNO ACCADEMICO 2021-2022

Indice

Sommario

Molti sistemi sviluppati nel campo del machine learning sono *rule-based*, ovvero rappresentano la conoscenza acquisita in forma di regole, ereditando diversi vantaggi: le regole risultano essere compatte, modulari ed esplicite dando modo agli esperti di dominio di analizzarle e controllare la veridicità delle stesse. Negli ultimi anni le reti neurali artificiali vengono sempre più utilizzate nei sistemi di apprendimento e la conoscenza acquisita viene rappresentata implicitamente all'interno dell'architettura della rete e nei valori attribuiti ai *pesi*. In generale però risulta complicato ottenere la completa comprensione del processo di apprendimento laddove in molti casi vi è necessario. Si possono definire fondamentalmente due approcci per capire il funzionamento della rete: basandosi sui valori dei pesi e il cambiamento di essi durante il processo di apprendimento, oppure adottare un approccio che ha l'obiettivo di mettere insieme le statistiche degli output della rete. Nel primo caso giungeremmo ad una serie di valutazioni molto generali e nel secondo approccio saremmo costretti a ripercorrere tutto il processo di apprendimento. Nel corso di questa tesi sarà presentato un approccio atto a superare alcuni limiti dei sistemi allo stato dell'arte (i.e. sistemi basati su embedding), quindi un modello che incorpora una nuova metodologia con l'obiettivo di rendere più *interpretabile* il processo di *estrazione* della conoscenza.

L'abilità di imparare da dati con un certo grado di incertezza, o informazioni mancanti, è fondamentale nei sistemi di apprendimento. Nel *mondo reale*, le informazioni possono essere mancanti in accordo con diversi fattori, come per

esempio rumore durante le misurazioni. Ci sono due motivi per cui si desidera modellare in maniera esplicita la trattazione degli input incerti: in primo luogo potremmo essere interessati alle relazioni tra gli input reali e gli output, in secondo luogo il problema potrebbe essere non-stazionario, nel senso che per diversi esempi input differenti possono essere mancanti o il livello di incertezza può essere vario. A tal proposito verrà definita una strategia per trattare il problema basata sull'algoritmo EM.

Il lavoro che segue è organizzato in capitoli ed ognuno di essi ha il compito di guidare, gradualmente, il lettore attraverso i concetti utili alla comprensione degli obiettivi preposti.

- Il primo capitolo fornisce un panoramica generale sui sistemi basati su conoscenza, fornendo anche alcuni cenni storici nella prima parte e concentrandosi, nella seconda parte, sulla rappresentazione ontologica della stessa e gli strumenti disponibili per tale rappresentazione, come OWL e le logiche descrittive.

- Nel secondo capitolo si introdurranno i modelli generativi, puntando più a definire un approccio diverso a quello discriminativo e, successivamente, verrà descritto lo stato dell'arte con l'obiettivo di concentrarsi sul presentare una nuova idea di modello che superi i diversi aspetti negativi dell'attuale trend. Si definirà, quindi, il modello in questione sia in forma basilare che gerarchica.

- Nel terzo capitolo verrà introdotto il concetto di incertezza e come può impattare sui risultati, concentrandosi più su un aspetto legato ai dati e all'apprendimento basato su di essi. Verranno definite le diverse categorie, sia di dati mancanti o incompleti, sia i diversi approcci per gestire gli stessi. Verrà definito, quindi, uno di questi approcci, l'algoritmo

Expectation-Maximization e la sua implementazione sul modello in forma basilare descritto nella seconda parte del secondo capitolo.

- Il quarto capitolo sarà dedicato alla parte sperimentale, definendo gli aspetti tecnici ed implementativi del progetto su cui si basa il lavoro e, quindi, i risultati ottenuti.

- Nel quinto ed ultimo capitolo, infine, seguiranno le conclusioni e verranno elencati i possibili sviluppi futuri.

Capitolo 1

Sistemi Basati su Conoscenza

I sistemi basati su conoscenza (KBS, *Knowledge Based System*) sono stati definiti come sistemi informatici che utilizzano la conoscenza riguardo un dominio al fine di agire o supportare la risoluzione di problemi complessi [PM17] all'interno del dominio stesso. La conoscenza viene codificata sotto forma di regole, fatti o relazioni, e viene utilizzata per supportare il processo decisionale automatizzato o per fornire informazioni agli utenti. Quest'ultima può essere acquisita da fonti esterne, come libri di testo o esperti del settore, o può essere sviluppata, mediante l'utilizzo di dati, internamente attraverso l'apprendimento automatico [Mit97]. I sistemi basati su conoscenza sono importanti perché consentono di trasferire la conoscenza esperta in una forma che può essere utilizzata in modo efficiente e preciso dai computer. Inoltre, essi consentono di gestire grandi quantità di informazioni in modo più rapido ed accurato rispetto ad un essere umano, consentendo di prendere decisioni basate sulla conoscenza in modo rapido e affidabile. I sistemi basati su conoscenza possono considerarsi un tipologia di computer di quinta generazione [AS09] cercando di differenziarsi dai tradizionali sistemi informativi *Computer-Based* e pongono diversi scopi [Tur90], tra i più importanti:

- offrono una grande quantità di conoscenza in diverse aree;
- acquisiscono nuove *percezioni* simulando situazioni non conosciute;

- offrono un importante miglioramento nel campo della produttività dei software;
- riducono costi e tempi legati allo sviluppo di sistemi computerizzati.

1.1 Struttura e Categorie

Il tema che accomuna tutti i sistemi basati su conoscenza è il tentativo di rappresentare conoscenza in forma esplicita con una componente di ragionamento che gli consenta di derivare conoscenza. Pertanto un dato sistema basato su conoscenza si contraddistingue per due caratteristiche alla base della struttura: base di conoscenza e motore inferenziale.

- La base di conoscenza rappresenta fatti e regole riguardanti il mondo, ossia il dominio che si desidera modellare. Spesso il dominio è descritto sotto forma di ontologia, ovvero concetti organizzati gerarchicamente, in contrapposizione a quella in forma procedurale, volta ad incorporare il dominio implicitamente, cioè all'interno del codice. Forme aggiuntive rispetto a quella ontologica includono frame, grafi concettuali e asserzioni logiche [Sow00]. Questa può essere usata in due momenti distinti dal sistema: *online* quando vengono utilizzate, oltre la base di conoscenza, le osservazioni del mondo corrente, i suoi obiettivi e le sue abilità per scegliere cosa fare e aggiornare la base di conoscenza stessa e *offline*, quando, grazie alla conoscenza di fondo o esperienze passate si costruisce la base di conoscenza utilizzata, quindi, per agire online [PM17].
- Il motore inferenziale, invece, si può definire come il *cuore* di un KBS. L'idea dietro questo componente è quella di prendere in input la base di conoscenza e utilizzare algoritmi di ragionamento per dedurre informazioni che possono essere utilizzate per rispondere alle domande dell'utente, risolvere problemi e prendere decisioni [RN99].

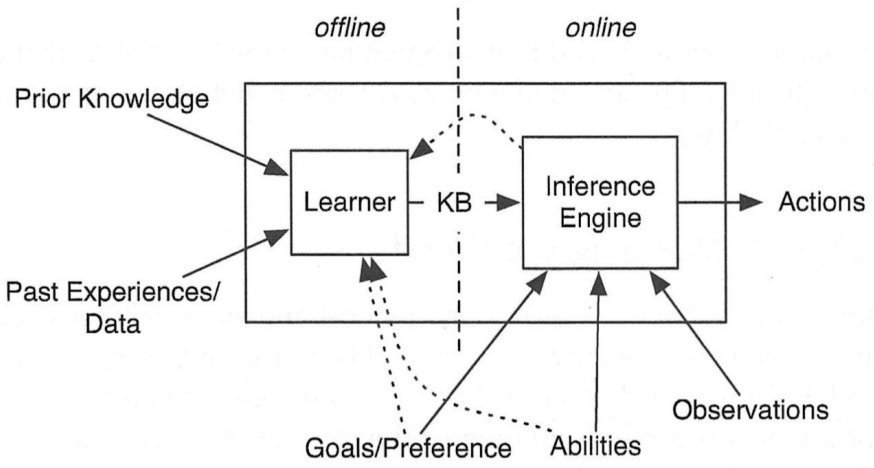


Figura 1.1: Decomposizione offline/online di un sistema basato su conoscenza [PM17]

In sintesi, la base di conoscenza rappresenta ciò che il sistema sa, mentre il motore inferenziale utilizza questa conoscenza per risolvere problemi.

Questi sistemi sono stati descritti per la prima volta da McCarthy nel 1956, che li ha definiti come programmi che possono ragionare e utilizzare la conoscenza per risolvere problemi [McC59]. Nel corso del tempo si sono susseguite diverse evoluzioni che hanno determinato categorie diverse di KBS [TL91], nello specifico le principali sono:

- Sistemi Esperti
- *Linked Systems*
- *Case-Based Systems*
- Database in congiunzione con interfacce intelligenti
- Sistemi di tutoring intelligenti

Sistemi Esperti: il Primo Approccio

I primi KBS sviluppati sono chiamati Sistemi Esperti, questi consistono nel primo approccio alla sostituzione di esperti di dominio nella risoluzione di problemi. Si considerano, quindi, i pionieri dei sistemi basati su conoscenza [AS09] e spesso i termini vengono scambiati erroneamente: laddove col termine *Sistema Basato su Conoscenza* ci si riferisce all'architettura del sistema, col termine *Sistema Esperto* ci si riferisce al tipo di compito per cui il sistema stesso è creato, quindi, in cosa sostituirà l'esperto di dominio.

L'esempio più noto di questa classe di sistemi è *MYCIN* [SB75; BS84], un sistema progettato nei primi anni '70 all'Università di Stanford per diagnosi mediche. Nello specifico il suo compito era identificare batteri che causavano infezioni, come la meningite e raccomandare antibiotici e dosaggi degli stessi in base al peso corporeo del paziente. In questo sistema si rappresentavano *fatti* presenti in un database attraverso asserzioni su cui si ragionava tramite l'uso di regole. Seppure MYCIN non sia mai stato utilizzato per i suoi scopi originari, principalmente per una questione di responsabilità da parte dei medici di fornire diagnosi errate [Cha14], questo sistema ha dimostrato di possedere diverse potenzialità, infatti negli anni a seguire sono stati sviluppati ulteriori sistemi *Rule-Based* anche in ambito non medico.

1.2 Ontologie e Knowledge Graph

Come detto precedentemente una base di conoscenza può definirsi anche in forma ontologica. In filosofia, con il termine ontologia ci si riferisce allo studio dell'esistenza, più precisamente riguardo ciò che esiste, o cosa dovrebbe esistere in un particolare dominio [PM17]. In ambito di *Intelligenza Artificiale* (IA), un'ontologia è un modello formale che descrive la struttura concettuale di un determinato dominio. In altre parole, definisce le categorie di oggetti, concetti e relazioni

esistenti in un particolare campo di conoscenza e la loro organizzazione logica. Questo modello serve come base per lo sviluppo di sistemi di intelligenza artificiale per il *Ragionamento Automatico* o per il *Knowledge Management*.

Il termine *grafo di conoscenza* è stato introdotto negli anni '70 e '80 del secolo scorso, in ambito di IA e di rappresentazione della conoscenza. In particolare, la rappresentazione della conoscenza mediante grafi è stata sviluppata da ricercatori come Sowa [Sow83] e Brachman [BL04], il quale hanno proposto l'idea di sfruttare le potenzialità intrinseche dei grafi di conoscenza per basi di conoscenza, nello specifico ontologie, e sistemi di intelligenza artificiale. Un grafo di conoscenza o anche rete semantica [Sow92] è, quindi, un modello di rappresentazione delle relazioni tra entità in un particolare dominio di conoscenza. Viene definito come un grafo, dove i nodi si riferiscono alle entità e gli archi le relazioni tra di loro. Questo tipo di incorporamento offre diversi vantaggi:

- Interpretabilità: i grafi di conoscenza definiscono le relazioni tra i concetti in modo intuitivo;
- Scalabilità: grazie a questi si possono rappresentare relazioni complesse tra elementi, anche in presenza di grandi moli di informazioni;
- Ricerca efficiente: i grafi di conoscenza rendono più veloci e precisi i processi di ricerca e recupero delle informazioni [SCL22];
- Integrazione di fonti diverse: l'utilizzo di questo modello di rappresentazione rende più semplice integrare informazioni provenienti da diverse fonti, questo perché forniscono un modello comune per rappresentare le relazioni tra le informazioni.

Pertanto si intuisce che le ontologie vengono spesso definite attraverso grafi di conoscenza. Ad esempio, in un'ontologia del mondo animale, un nodo potrebbe rappresentare la classe `Vertebrati` e un altro nodo potrebbe rappresentare la

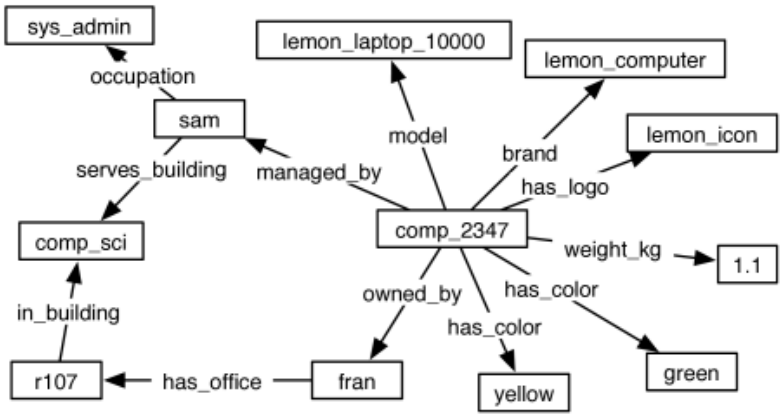


Figura 1.2: Un esempio di rete semantica [PM17]

classe `Mammifero`. Un arco che va da `Mammifero` a `Vertebrati` potrebbe rappresentare la relazione `è una classe di`. In questo modo, i grafi di conoscenza permettono di modellare e rappresentare in modo visivo la struttura e le relazioni tra le diverse classi e concetti in un'ontologia. Questo rende più facile per i sistemi comprendere e utilizzare le informazioni contenute nell'ontologia per rispondere a domande o effettuare inferenze.

1.3 Assunzioni sulla Conoscenza

In un sistema logico formale utilizzato per la rappresentazione della conoscenza è opportuno fare delle assunzioni in merito al dominio di applicazione, precisamente sulla quantità di conoscenza che si suppone di avere a disposizione. Questa scelta ha un impatto importante sulla capacità del sistema di adattarsi a situazioni incerte o in continua evoluzione e sulla sua capacità di acquisire nuove conoscenze. Generalmente si possono fare due tipi di assunzioni, l'assunzione di mondo aperto *Open World Assumption* (OWA) e di mondo chiuso *Closed World Assumption* (CWA). Le due si distinguono per come il sistema reagisce nel momento in cui cerca di provare il valore di verità di un determinato concetto di cui non dispone

di informazioni complete. Nel caso di OWA, un sistema deve essere in grado di ragionare in un ambiente incerto e in continua evoluzione, dove non si dispone di una conoscenza completa del mondo che lo circonda. Ciò significa che l'IA deve essere in grado di adattarsi a nuove situazioni e di acquisire nuove conoscenze senza essere programmata in modo specifico per ogni scenario possibile. Si contrappone al concetto di CWA che intende un ambiente specifico e ben definito in cui un sistema di intelligenza artificiale deve funzionare, quindi, con una conoscenza completa del mondo che lo circonda. In questo caso, il sistema è limitato a funzionare solo in quel determinato ambiente e non è in grado di adattarsi a situazioni incerte o a nuove conoscenze.

Si deduce, quindi, che l'assunzione di mondo aperto risulta essere una sfida per i ricercatori, poiché richiede una flessibilità e una capacità di apprendimento continuo che non è tipica dei sistemi IA esistenti. L'assunzione del mondo aperto è tipica dei linguaggi per il Web semantico, come l'*Web Ontology Language* (OWL); al contrario, in generale la CWA è tipica dei linguaggi relazionali [Ber09].

1.4 Rappresentazione della Conoscenza nel Web Semantico

Il web semantico costituisce una via che permette ai sistemi basati su conoscenza di interpretare le informazioni distribuite sul World Wide Web. Invece di proporre solo pagine HTML leggibili e comprensibili dagli umani, i siti web forniscono informazioni utili anche ai calcolatori [PM17]. La semantica è una parte specifica del linguaggio che permette lo studio e quindi la comprensione del significato preciso delle parole o delle frasi. Quando si parla di web semantico, quindi, si parla di un Web in grado di comprendere in pieno quanto gli viene richiesto e questo viene fatto attraverso l'utilizzo di ontologie, grafi di conoscenza e altre tecnologie di intelligenza artificiale per descrivere le relazioni tra i concetti e

le informazioni sul Web. Seppur possa sembrare intuibile quindi l'utilizzo del metalinguaggio XML per la costruzione e definizione di un particolare documento questo offre non poche limitazioni nel contesto di utilizzo sia sintatticamente che semanticamente:

- la sintassi XML non definisce nessun meccanismo esplicito per qualificare le relazioni tra documenti;
- la semantica XML costituisce il limite di non essere in grado di capire la correlazione di uno stesso soggetto di cui si parla in documenti diversi.

A seguito di ciò e della continua evoluzione del web semantico, il *World Wide Web Consortium* (W3C) definisce un nuovo standard sulla base di XML, chiamato *Resource Description Framework* (RDF) standardizzando così la definizione di relazioni tra informazioni ispirandosi alla logica dei predicati, esprimendo così le asserzioni della logica di primo ordine come triplette formate da soggetto, predicato e oggetto. Così è, quindi, possibile identificare una risorsa attraverso uno *Unique Resource Identifier* (URI). Una risorsa è tutto ciò che è possibile identificare univocamente, inclusi individui, classi e proprietà [PM17].

Graficamente è possibile rappresentare asserzioni sempre tramite grafi dove i nodi costituiscono soggetto e oggetto e gli archi (orientati da soggetto a oggetto) per i predicati. Sebbene RDF come strumento si ispiri alla logica del primo ordine, quest'ultima risulta troppo complessa. RDF è limitato nella sua capacità di rappresentare relazioni complesse e inferenze tra le informazioni, rendendo difficile la creazione di ontologie e la gestione delle conoscenze sul Web.

OWL: Web Ontology Language

OWL è stato sviluppato per fornire una forma di linguaggio più espressivo ed ad alto livello che consente di descrivere ontologie più complesse e di eseguire inferenze logiche sulle informazioni presenti sul Web. Tramite questo linguaggio,

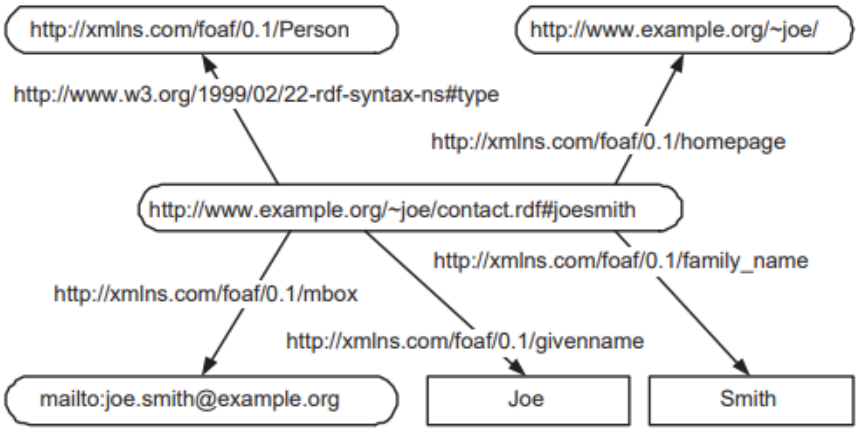


Figura 1.3: Esempio di una rappresentazione grafica di RDF: gli URI sono identificati dagli ellissi e i rettangoli contengono semplici stringhe [Obi07]

gli sviluppatori possono creare o modellare ontologie consentendo un maggior grado di dettaglio per descrivere significati e relazioni tra le informazioni sul Web, quindi, i sistemi possono utilizzare queste informazioni per integrare e combinare i dati da fonti diverse in modo più efficace.

L'idea di OWL nasce dal consorzio W3C per superare i limiti espressivi di RDF [AH09] e fornire un linguaggio più potente per descrivere e gestire le informazioni sul Web espandendo così le opportunità per OWL e il Web Semantico [DPJ17]. In particolare, se RDF ed RDFS consentono di definire la struttura delle triple e le classi che rappresentano i concetti di soggetto, predicato ed oggetto, OWL aggiunge la semantica a questo schema permettendo di descrivere più dettagliatamente proprietà e classi, esprimendosi sempre in forma di triple.

Le ontologie OWL sono costituite da classi, proprietà e individui:

- le classi sono insiemi di individui che condividono una certa proprietà o caratteristica;
- le proprietà rappresentano le relazioni tra le classi e gli individui.

Queste ultime possono essere di due tipi:

- *Object Properties*, che definiscono una relazione tra due individui;
- *Data Properties*, che definiscono una relazione tra un individuo e il tipo di valore da attribuire ai dati.

Gli individui sono oggetti specifici che appartengono a una classe.

OWL mette a disposizione un insieme di strumenti per definire ontologie in modo strutturato, consentendo di specificare le proprietà delle classi e degli individui, le relazioni tra le classi, le restrizioni sulle proprietà e le classi, e molto altro.

Ci sono tre versioni, o profili, di OWL: OWL Lite, OWL DL e OWL Full, che differiscono per la complessità e le funzionalità supportate:

- OWL Lite si concentra sulle funzionalità di base necessarie per descrivere ontologie semplici, come le classi e le proprietà. OWL Lite è sufficiente per molte applicazioni semplici, come la categorizzazione di documenti e la descrizione di oggetti e concetti comuni, tuttavia presenta alcune limitazioni, come il non supportare la negazione di classi, le restrizioni sui nomi e altre funzionalità avanzate;
- OWL DL, acronimo di *Description Logic*, supporta funzionalità più avanzate rispetto a OWL Lite, come la negazione di classi, le restrizioni sui nomi e le classi di equivalenza, inoltre, supporta anche il ragionamento automatizzato, che consente di inferire nuove conoscenze a partire dalle ontologie esistenti. Anche questo profilo presenta alcune restrizioni sulla complessità dell'ontologia, che ne limitano la potenza;
- OWL Full è la versione più potente di OWL, che supporta tutte le funzionalità del linguaggio. Risulta essere molto più espressivo di OWL Lite e OWL DL, ma non supporta il ragionamento automatizzato, quindi, si ritiene particolarmente adatto adatto per ontologie molto complesse che richiedono la massima espressività possibile.

OWL consente di creare ontologie ben strutturate, che possono essere utilizzate per la ricerca semantica, la gestione delle informazioni e l'integrazione di dati provenienti da diverse fonti. Grazie a questo strumento è, quindi, possibile rappresentare in modo formale la conoscenza di un dominio specifico e condividerla in modo interoperabile tra diversi sistemi.

OWL 2 è la versione successiva di OWL, sviluppata con l'obiettivo di migliorare il linguaggio e renderlo più espressivo e utile. Nonostante ciò, mantiene la compatibilità con le ontologie esistenti scritte nella sua versione precedente e introduce nuove funzionalità e miglioramenti significativi.

OWL 2 è composto da tre ulteriori profili:

- OWL 2 EL (*Expressive Profile*) è un profilo semplificato di OWL 2, il cui utilizzo risulta familiare a OWL 2 DL. Viene spesso impiegato per ontologie di tipo biosanitario che prevedono descrizioni strutturate complesse o un grande numero di classi permettendo quindi l'assenza di restrizioni sugli assiomi;

- OWL 2 QL (*Query and Reasoning Language*) è un profilo che supporta la costruzione di ontologie per la query e il ragionamento su grandi quantità di dati. Questo profilo è stato progettato per consentire una rapida elaborazione di query su ontologie con molte istanze;

- OWL 2 RL (*Rule Language*) è un profilo creato per applicazioni che richiedono scalabilità senza sacrificare espressività, adattandosi a quelle applicazioni OWL 2 che richiedono uno scambio di espressività per l'efficienza ed applicazioni RDF(S) che al contrario richiedono maggiore espressività, tipica di OWL 2.

OWL 2 supporta anche nuove funzionalità come l'annotazione di ontologie, l'uso di punteggiature Unicode e il supporto per le espressioni regolari.

Logiche Descrittive

Le *logiche descrittive* (*Description Logics*, DL) sono una famiglia di formalismi *logic-based* atti a rappresentare la conoscenza, precisamente la parte terminologica di un dominio applicativo in modo strutturato e formale [BHS08].

La logica descrittiva si basa su un insieme di costrutti sintattici, chiamati *concetti* e *ruoli*, che sono utilizzati per descrivere le classi e le proprietà dell'ontologia. I concetti possono essere *primitivi* o *definiti* in termini di altri concetti, utilizzando operatori logici come l'intersezione, l'unione e la negazione. I ruoli, invece, rappresentano le proprietà degli oggetti come, ad esempio relazioni con altre entità, e possono essere definiti in termini di altri ruoli.

Le logiche descrittive [Baa+07] devono il loro successo per la loro adozione in rappresentazioni per il web semantico come OWL. I principali motivi per cui è impiegato nell'ambito dei linguaggi ontologici sono: una semantica priva di ambiguità basata sulla logica dei predicati di primo ordine; l'alto livello di espressività che offre per costruire concetti e regole, quindi istanziare individui sulla base di questi; un procedura ben ottimizzata che consente di inferire nuove conoscenze a partire dalle ontologie esistenti, come l'identificazione di inconsistenze o di informazioni mancanti.

L'idea alla base di OWL-DL è data dall'impiego della logica descrittiva nel dominio dei linguaggi ontologici, ereditando la caratteristica di suddividere l'ontologia in due parti principali [Baa+07]: la *Terminology Box* (TBox) e la *Assertion Box* (ABox). La TBox rappresenta il vocabolario, ovvero la terminologia usata per descrivere i concetti e le proprietà. La ABox, invece, rappresenta, il "mondo reale" descritto dall'ontologia e contiene le informazioni specifiche riguardanti gli oggetti e le loro relazioni, in particolare cosa è vero in un determinato momento.

Ad esempio, nella TBox si può descrivere il concetto di `umano`, mentre nella ABox si possono definire le istanze di `umano` attraverso asserzioni, per esempio `Ulisse:umano`. In questo modo, la TBox rappresenta il vocabolario con cui si

definiscono le istanze, mentre la ABox rappresenta le istanze stesse.

La separazione tra TBox e ABox è utile in quanto consente di separare la definizione dei concetti e proprietà dal loro utilizzo effettivo, fornendo un'organizzazione logica e semantica all'ontologia. Questa separazione consente anche di separare il processo di costruzione dell'ontologia, in cui si definiscono i concetti e le proprietà, dal processo di utilizzo dell'ontologia, in cui si aggiungono le istanze specifiche. Sulla base della *prospettiva* del Web Semantico si può sfruttare l'infrastruttura del Web per creare un grafo globale di sorgenti *distribuite* di dati collegati, ovvero la conoscenza espressa in OWL. Così, la semantica dei dati può essere disponibile anche su più server, come *triple store* o DB NOSQL. I dati, così *collegati*, seguono i principi dei *Linked Data* [HB11]:

- L'uso di URI come nomi per le *cose*;
- L'uso di URI HTTP per ricercare i nomi sull'infrastruttura del Web;
- Fornire le informazioni usando standard, come RDF o SPARQL;
- Includere link ad altri URI, così da permettere di scoprire più *cose*.

Capitolo 2

Modelli Generativi

I modelli generativi sono un tipo di modelli di intelligenza artificiale, sviluppati per generare nuove informazioni in base a un insieme di dati di addestramento. Più specificatamente il funzionamento di un modello generativo è basato su una rappresentazione probabilistica dei dati ed il suo obiettivo è quello di descrivere la relazione tra le variabili che descrivono i dati cercando di imparare la distribuzione di probabilità che li ha generati [Goo+14]. Il modello viene quindi utilizzato per generare nuovi dati facendo uso di questa rappresentazione probabilistica. Più nello specifico, la stesura di questo lavoro vuole marcare quest'ultima caratteristica contrapposta a modelli complementari, ovvero i modelli discriminativi. Da questi due approcci si possono derivare analogamente classificatori generativi e discriminativi, quindi i rispettivi modelli che ne ereditano i nomi.

2.1 Reti Bayesiane e Modelli di Classificazione

Le *reti bayesiane* (o *belief networks*) sono modelli grafici probabilistici comunemente utilizzati per rappresentare la distribuzione di probabilità su un insieme di variabili casuali. Esse possono essere costruite / apprese e impiegate ai fini del ragionamento probabilistico e la generazione di dati e quindi più specificamente

per la predizione, ad es. ai fini della classificazione, rappresentando dipendenze arbitrarie tra le variabili con modelli facilmente estendibili con nuove variabili. Esse consentono, inoltre, anche di incorporare conoscenze pregresse, come ad esempio specifici vincoli tra le variabili.

Una rete bayesiana per un set di variabili $\mathbf{X} = \{X_1, ..., X_n\}$ consiste in una rete strutturata S che incorpora una serie di indipendenze condizionali sulle variabili $\mathbf{X}$ e un set P corrispondente alle distribuzioni di probabilità associate ad ogni variabile. Tutte queste componenti messe insieme definiscono la distribuzione di probabilità congiunta per $\mathbf{X}$.

Definizione 2.1.1 *Dato un insieme di variabili* $\boldsymbol{X} = \{X_1, ..., X_n\}$ *e denotato con* $\boldsymbol{Pa}$ *il nodo genitore di* X_i*, la probabilità congiunta della variabile* $\boldsymbol{X}$ *è definita come segue [Hec22]*

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa_i})$$

Il *Markov Blanket* è un concetto utilizzato nella teoria dei grafi probabilistici, che include sia i modelli di Markov che le reti bayesiane. In particolare, il Markov Blanket si riferisce all'insieme di nodi direttamente connessi a un nodo target in un grafo probabilistico, ovvero l'insieme di genitori, figli e nodi che condividono un genitore comune con il nodo target.

Il concetto di Markov Blanket è importante poiché permette di definire le dipendenze condizionali tra variabili in un grafo probabilistico. Conoscendo il Markov Blanket di un nodo, quindi, si può calcolare la distribuzione di probabilità condizionale di quel nodo dato il resto del grafo.

Come detto, sulla base delle reti bayesiane si possono definire specifici modelli di classificazione. Tali modelli, infatti, si possono distinguere in due tipologie, i classificatori basati sulla discriminazione (e.g. lineare o logistica), ovvero:

Definizione 2.1.2 (Modelli Discriminativi) *Un* modello discriminativo *è un modello che si basa sulla distribuzione di probabilità a posteriori della classe-obiettivo Y condizionata dalle variabili osservate X: $P(Y \mid X)$.*

In generale essi modellano il *confine decisionale* per le diverse classi di esempi.

Definizione 2.1.3 (Modelli Generativi) *Un modello generativo è un modello che si basa sulla probabilità congiunta $p(X, Y)$ delle variabili osservate X e della classe-obiettivo Y.*

Tali modelli sono più complessi e computazionalmente più costosi rispetto a quelli discriminativi.

2.2 Modelli Generativi e OWA

I modelli generativi possono essere utilizzati per supportare l'assunzione di conoscenza del mondo aperto poiché permettono di generare nuove informazioni in base a quelle esistenti. Ad esempio, un modello generativo addestrato su un insieme di immagini di animali può essere utilizzato per generare nuove immagini di animali che non sono presenti nel training set originale. Questo significa che il modello è in grado di gestire situazioni in cui non esiste una conoscenza precedente o dove la conoscenza disponibile non è sufficiente.

I modelli generativi possono essere utilizzati soprattutto per generare nuove informazioni che possono essere utilizzate per estendere o completare le conoscenze esistenti [Zha+18]. Ad esempio, un modello generativo addestrato su un insieme di descrizioni di paesaggi può essere utilizzato per generare nuove descrizioni di paesaggi che non sono presenti nel set di allenamento originale. Questo significa che il modello è in grado di gestire situazioni in cui la conoscenza disponibile è incompleta o insufficiente. Quanto appena affermato rende questi modelli particolarmente utili per sistemi di intelligenza artificiale che devono gestire situazioni imprevedibili e in continua evoluzione.

In generale va detto però che non esiste una categoria di modelli che sia universalmente considerata la migliore da utilizzare all'interno dell'assunzione di conoscenza del mondo aperto. La scelta del modello più appropriato dipende dalle specifiche esigenze e obiettivi del problema che si sta affrontando.

I modelli generativi possono essere particolarmente utili in contesti in cui si desidera generare nuovi dati o in contesti in cui si desidera modellare la distribuzione sottostante dei dati. Tuttavia, non sono necessariamente i modelli più adatti a tutti i problemi che si incontrano nell'assunzione di conoscenza del mondo aperto. Ad esempio, in un contesto in cui si desidera semplicemente classificare i dati in diverse categorie, un modello discriminativo potrebbe essere più adatto rispetto a un modello generativo. In altri contesti, potrebbe essere necessario utilizzare una combinazione di modelli generativi e discriminativi per ottenere risultati ottimali.

2.3 Modelli Generativi Basati su Embedding

In generale, la rappresentazione dei dati è uno dei principali obiettivi del machine learning, in quanto la qualità della rappresentazione dei dati ha un forte impatto sulla capacità del modello di generalizzare e di raggiungere prestazioni elevate su nuovi dati. L'*embedding* è una tecnica di *representation learning* in cui viene appreso uno spazio di rappresentazione a partire dai dati di input, in modo da esprimere le informazioni essenziali contenute nei dati in una forma più compatta e interpretabile.

I modelli generativi basati su embedding rappresentano una classe di modelli di apprendimento automatico che utilizzano tecniche di embedding per generare nuovi dati.

Il funzionamento di questo tipo di modelli comprende una rete neurale che viene addestrata per generare dati che sembrano simili ai dati di input. Questa

rete neurale può essere addestrata utilizzando diversi approcci, come il Generative Adversarial Network (GAN) [Goo+14], il Variational Autoencoder (VAE) o il Boltzmann Machine. I modelli generativi basati su embedding sono in grado di generare nuovi dati che hanno caratteristiche simili a quelle dei dati di input, aprendo la strada a diverse applicazioni come la generazione di testo, immagini, suoni e molto altro ancora.

L'"utilizzo di tecniche di representation learning, tra cui anche l'embedding, è in costante crescita negli ultimi anni nel campo del machine learning [Sch15; BCV14]. Ciò è dovuto principalmente alla loro efficacia nel migliorare le prestazioni dei modelli su una vasta gamma di compiti di apprendimento automatico e alla loro capacità di estrarre rappresentazioni informative e di bassa dimensionalità dai dati.

Tuttavia questo tipo di modelli presenta alcuni difetti, infatti, se da un lato allo stato dell'arte la classificazione si basa su un trend legato al representation learning, quindi mirando a mappare entità e relazioni e ridurre il problema dell'effettuare predizioni a problemi di algebra lineare [Wan+17], dall'altro questo assume non poche difficoltà, tra cui:

- incorporare all'interno dei modelli la conoscenza estratta dai grafi attraverso solo il ragionamento deduttivo;
- la difficile interpretabilità dei modelli stessi la quale rappresentazione degli spazi vettoriali non si riconduce facilmente alle caratteristiche originali e quindi durante il processo di classificazione porta questi modelli ad essere delle visti come delle *black-box* [Fd22].

2.4 Modelli Grafici Probabilistici

I modelli grafici sono una tipologia di modelli probabilistici che utilizzano grafi per rappresentare le relazioni tra le variabili in un sistema. In particolare, quando

il grafo è diretto aciclico (DAG, Directed Acyclic Graph) diremo che il modello si chiama rete bayesiana [Mur22] [Figura 2.1 (a)].

Esiste una forte correlazione tra i modelli generativi e i modelli grafici, poiché i modelli generativi possono essere rappresentati utilizzando un grafo probabilistico. Pertanto si sottolinea come sia possibile non solo avvalersi di manipolazioni algebriche, per risolvere complessi modelli probabilistici, ma come possa essere vantaggioso migliorare l'analisi degli stessi attraverso la rappresentazione diagrammatica delle distribuzioni di probabilità, ottenendo così modelli probabilisti grafici. Questi offrono diversi vantaggi [Bis06]:

- permettono in una maniera semplicistica di visualizzare la struttura dei modelli probabilistici e possono essere usati per progettare e motivare nuovi modelli;
- approfondimenti rispetto alle proprietà del modello, incluse le proprietà di indipendenza condizionata, possono essere ottenute guardando il grafo;
- calcoli complessi, richiesti per l'inferenza e l'apprendimento in modelli complessi, si possono esprimere in termini di manipolazioni grafiche, portando avanti implicitamente le espressioni matematiche che ne conseguono.

I modelli grafici catturano i processi *causali* da cui i dati osservati vengono generati. Per questo motivo, questi modelli sono spesso chiamati *modelli generativi*.

2.5 Interpretabilità dei Modelli

Sfruttare le grandi basi di conoscenza, in particolar modo la loro rappresentazione come knowledge graph [Hog+21], permette di aumentare il dominio di applicazioni e la complessità delle stesse, nello specifico parliamo di basi di conoscenza

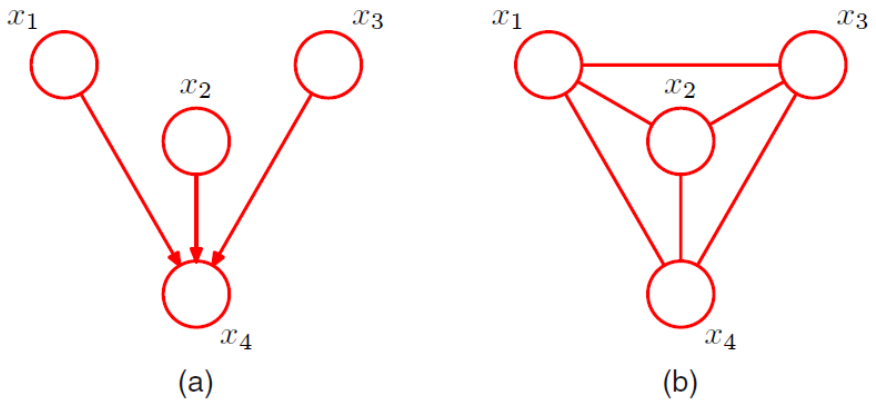


Figura 2.1: Esempi di un DAG (a) e il suo corrispondente *moral graph*, ciclico e indiretto (b) [Bis06]

presenti su infrastrutture web (ontologie web). La classificazione, in questo contesto, potrebbe risultare un compito complesso, soprattutto quando è richiesto che la semantica incorporata all'interno debba essere preservata il più possibile.

Con l'obiettivo di migliorare l'interpretabilità, si potrebbe ricorrere ad un approccio che ha lo scopo di semplificare i modelli probabilistici facendo affidamento sulle funzioni di variabili discrete che fanno leva sulle caratteristiche logiche di base. Analizzando un'idea nel contesto dell'apprendimento neurale [THA97; JJ93], è stato visto come è possibile adattare modelli grafici e, quindi, convertirli in regole probabilistiche partendo proprio dalla rappresentazione delle istanze nei grafi di conoscenza per produrre assiomi logici. I modelli probabilistici descritti di seguito hanno l'obiettivo di essere più interpretabili e verificabili dagli esperti di dominio. Quindi i punti cardine su cui si basano sono:

- l'assunzione di indipendenza, tipica dei modelli Naive Bayes [DP97];
- le prestazioni anche in presenza di conoscenza incerta, quindi dati mancanti, correlata a quella di OWA.

In primo luogo richiamiamo i concetti già descritti nella sezione 1.4 concentrandoci, quindi, su i grafi di conoscenza che possono essere rappresentati attraverso le logiche descrittive mediante OWL-DL, come descritto sempre nella suddetta sezione. Nello specifico definiamo con $\mathcal{K}$ la base di conoscenza descritta in DL, quindi:

$$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$$

dove $\mathcal{T}$ e $\mathcal{A}$ si riferiscono rispettivamente alla TBox e ABox. Con $Ind(\mathcal{K})$ ci riferiamo agli individui facenti parte della base di conoscenza $\mathcal{K}$.

Si definisce una codifica [Fd22] basata su una rappresentazione semplificata degli individui, come già proposto in studi precedenti basati sulle *distanze* o sulle funzioni-kernel [dFE08; Ret+12]. Più specificamente, definiti l'insieme $\mathbb{B} = \{0, 1\}$ ed un insieme di *feature di base*, altrove denominato anche *contesto*, $\mathcal{C} = \{F_i \mid i = \ 1, ..., D\}$, un individuo $\mathtt{a} \in Ind(\mathcal{K})$ può essere rappresentato da un vettore booleano $\mathbf{a} \in \mathbb{B}^D$ definito come segue:

$$a_i = \begin{cases} 1 & \mathcal{K} \vdash F_i(\mathtt{a}) \\ 0 & \mathcal{K} \vdash \neg F_i(\mathtt{a}) \end{cases}$$

dove $\vdash$ indica la procedura di dimostrazione[1] usata per decidere dell'appartenenza di un dato individuo $\mathtt{a}$ ad una feature F_i. Si osservi, però, che non sempre è possibile determinare tali appartenenze in quanto non si assume la *conoscenza completa* del dominio e, come detto inizialmente, sarà questo uno dei problemi affrontati nel seguito.

2.5.1 Modello Bernoulli Multivariato

Data la rappresentazione degli individui come tuple di dimensione D e valori binari, e indicando con y l'appartenenza ad una classe, la distribuzione

[1] Corrispondente (se completa e consistente) alla relazione di *conseguenza logica* della *Teoria dei modelli*, che è indicata in genere con $\models$.

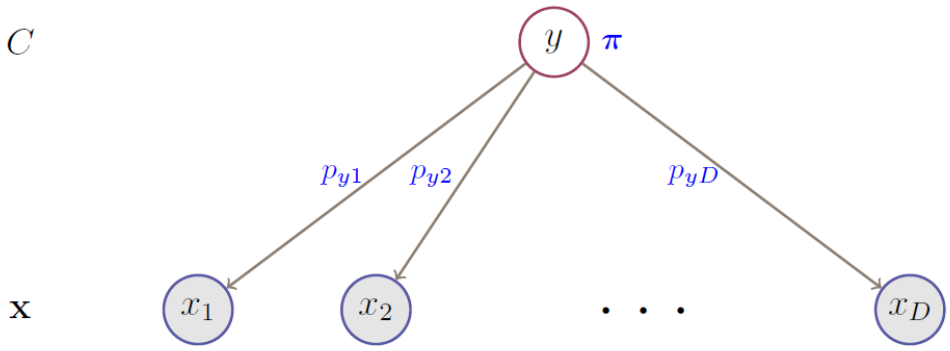


Figura 2.2: Grafo relativo a un MBM

di ogni individuo può essere descritta attraverso un particolare modello *Naive Bayes* (NB) [Mur22], precisamente come *Modello Bernoulli Multivariato* (MBM) [Fd22] e la probabilità condizionata è così definita:

$$P(\mathbf{x}|y) = Ber_D(\mathbf{x}|\mathbf{p}_y) = \prod_{i=1}^{D} Ber(x_i|p_{yi}) = \prod_{i=1}^{D} (p_{yi})^{x_i}(1-p_{yi})^{1-x_i}$$

Laddove $\mathbf{x}$ è considerato un vettore definito come segue:

$$\mathbf{x} = (x_1, x_2, ..., x_D)^T$$

Il problema della classificazione consiste nello stimare l'output di y in base ai dati di input rappresentati da tutti i vettori $\mathbf{x}$ e, quindi, dati i parametri del modello di probabilità a priori $P(y)$ e quella condizionata $P(\mathbf{x} \mid y)$, la distribuzione viene calcolata usando Bayes:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{\pi^y Ber_D(\mathbf{x};\mathbf{p}_y)}{P(\mathbf{x})}$$

Infine, l'appartenenza dell'individuo viene predetta massimizzando $P(y \mid \mathbf{x})$:

$$\hat{y}(\mathbf{x}) = \arg\max_{y \in \mathbb{B}} P(y|\mathbf{x})$$

che equivale ad una procedura di decisione di questo tipo:

$$\mathcal{K} \mid\sim \begin{cases} C(x) & if\ P(y=1 \mid \mathbf{x}) > 1/2 \\ \neg C(x) & altrimenti \end{cases}$$

Dove con $|\sim$ si intende come procedura di classificazione. Inoltre è possibile definire soglie più accurate rispetto a $1/2$.

Ciò che davvero rende il modello più interpretabile è come le regole per la classificazione vengono estratte. Più specificatamente, le regole congiunte di classificazione di un individuo $\mathtt{x}$ per una determinata classe C sono definite dai valori di probabilità p_{yD} di ogni x_D, più nello specifico tali regole possono essere così definite:

IF $\mathcal{K} \mid\sim C(\mathtt{x})$ con probab. a priori π THEN
(le x_i sono bernoulliane indipendentemente distribuite)
AND $x_1 = 1$ con probab. p_{11}
AND $x_1 = 0$ con probab. $1 - p_{11}$
AND $x_2 = 1$ con probab. p_{12}
AND $x_2 = 0$ con probab. $1 - p_{12}$
...
AND $x_D = 1$ con probab. p_{1D}
AND $x_D = 0$ con probab. $1 - p_{1D}$

dove si possono considerare per ogni $i = 1, ..., D$ solo le congiunte più probabili, in altre parole con i valori delle x_i corrispondenti a $\mathtt{max}(p_{1i}, 1 - p_{1i})$. Analogamente, le regole per il complemento $\neg C$ si possono definire in termini dei parametri p_{0i} e $\pi^0 = 1 - \pi$

Definita una soglia $\theta \in [0.5, 1]$ si può estrarre una definizione logica per un concetto target, corrispondente al problema di classificazione, quindi, partendo da un insieme $F^+ = \{i \in \{1, ..., D\} \mid p_{1i} > \theta\}$ corrispondente alle feature correlate con l'appartenenza *positiva* al concetto target ed analogamente un insieme $F^- = \{i \in \{1, ..., D\} \mid 1 - p_{1i} > \theta\}$ di indici di complementi di feature correlate con C, si contribuisce a definire il concetto-obiettivo attraverso la non-appartenenza, o

appartenenza *negativa*. Da questo si può definire definire l'assioma:

$$C \sqsubseteq \bigsqcap_{i \in F^+} F_i \sqcap \bigsqcap_{j \in F^-} \neg F_j$$

Questo risulta più interpretabile da parte di un esperto di dominio e permette, quindi, un maggior controllo e possibili modifiche per definire la classe obiettivo C. Brevemente, si definisce con $\bigsqcap$ l'operatore di intersezione tra insiemi, per fare un esempio:

$$\bigsqcap_{i \in I} A_i$$

rappresenta l'intersezione di tutti gli insiemi A_i, con i che varia tra tutti gli elementi dell'insieme I, ovvero, si prendono tutti gli elementi che sono contenuti in ogni insieme A_i, e si formano un nuovo insieme che rappresenta l'intersezione di tutti questi insiemi. Con $\sqsubseteq$, invece, definiamo l'operatore di sottorelazione o di inclusione parziale tra due insiemi, un esempio:

$$R \sqsubseteq S$$

si dice che R è sottorelazione di S, se ogni coppia ordinata (a,b) che soddisfa la relazione R soddisfa anche la relazione S.

Si assume la disponibilità di un *training set* completo:

$$\mathbf{T} = \langle \mathbf{X}, \mathbf{y} \rangle = \{(\mathbf{x}^t, y^t)\}_{t=1}^N$$

dove $\mathbf{x}^t$ indica la codifica di un individuo $\mathtt{x}^t \in Ind(\mathcal{K})$ e $y^t \in \mathbb{B}$ indica l'appartenenza alla classe obiettivo C. Il modello NB lavora essenzialmente ottimizzando *lo stimatore di massima verosimiglianza* (MLE, *maximum likelihood estimation*) [Mur22] per i parametri $\mathbf{\Pi} = \{\boldsymbol{\pi}, \mathbf{p_y}\}$, che definiscono rispettivamente la probabilità a priori delle classi e la distribuzione di probabilità delle feature per classe. Il MLE per la probabilità a priori viene stimato come segue:

$$\hat{\pi}^y = \frac{N_y}{N}$$

Per quanto riguarda la distribuzione di probabilità di ogni feature:

$$\hat{p}_{yi} = \frac{N_{yi}}{N} \qquad y = 0, 1; i = 1, ..., D$$

Un'ulteriore caratteristica del modello è la sua capacità a prestarsi in maniera positiva anche ad una situazione dove l'indipendenza su cui si basa non risulta sempre vera, rivelandosi anche poco propenso al *sovradattamento* [DP97; Mur22], infatti può essere mantenuto l'approccio bayesiano considerando distribuzioni di probabilità a priori come quella Beta e di Dirichlet in quanto la prima risulta adatta proprio alle variabili binarie e la seconda una generalizzazione della seconda, per variabili con più valori/stati possibili [Mur22]. Si possono applicare anche metodi basati sulla minimizzazione dell'errore quadratico

$$E(\mathbf{T}) = \sum_{t=1}^{N} [y^t - \hat{y}(\mathbf{x}^t)]^2$$

per correggere i parametri implementando strategie di ottimizzazione basate sulla *discesa del gradiente*. Il metodo della discesa è particolarmente utile in situazioni in cui la funzione da minimizzare è complessa e non ha una soluzione analitica, o quando il calcolo della soluzione analitica richiede troppo tempo o risorse computazionali. In questo caso, l'algoritmo cerca il minimo attraverso un processo iterativo, che parte da un punto iniziale e, ad ogni passo, si sposta in direzione opposta al gradiente della funzione, fino a raggiungere il minimo.

Per quanto riguarda casi di dati incompleti o mancanti si può pensare ad approcci anche qui bayesiani, ignorando questi e facendo leva sull'assunzione di indipendenza, mediando, quindi, sui valori disponibili. Diversamente si possono adottare strategie più *complete*, come un approccio generativo [GJ93] basato sull'algoritmo *Expectation-Maximization*, trattato più nello specifico nella sezione 3.3.

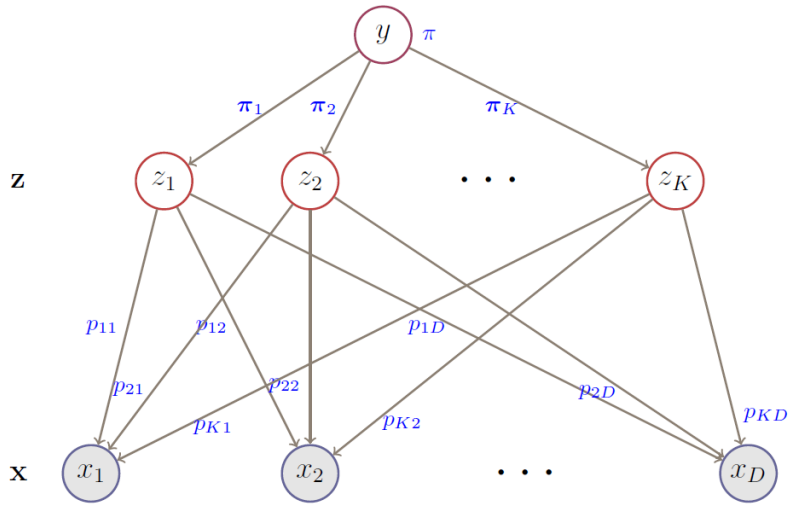


Figura 2.3: Una rete di tipo Mixture of Multivariate Bernoulli Model [Fd22]

2.5.2 Modello Misto Bernoulli Multivariato

Un'ulteriore evoluzione a questo modello è possibile considerando una gerarchia di variabili bernoulliane miste e rilassando la proprietà di indipendenza condizionata [Fd22]. Nella fattispecie, stimare $P(\mathbf{x} \mid y)$ attraverso MBM quando non è possibile l'assunzione di indipendenza su cui si basa il modello, farebbe risultare lo stesso impreciso. Una possibile soluzione potrebbe essere aumentare la dimensionalità del modello stesso aggiungendo ulteriori livelli *nascosti* che fungono da variabili indicatrici binarie $\mathbf{z}$, quindi $\{z_k\}_{k=1}^{K}$, definendo più distribuzioni di appartenenza per ogni individuo $P_k(\mathbf{x} \mid z_k)$. In questo caso possiamo definire un *Modello Misto Bernoulli Multivariato* (MMBM) [Figura 2.3], i cui parametri da stimare sono $\mathbf{\Pi} = \{\pi, \boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_K, \mathbf{p}_1, ..., \mathbf{p}_K\}$, quindi:

$$P(\mathbf{x} \mid \mathbf{\Pi}) = \sum_{k=1}^{K} P(\mathbf{x}, z_k \mid \mathbf{\Pi}) = \sum_{k=1}^{K} P(z_k \mid \mathbf{\Pi}) P_k(\mathbf{x} \mid z_k, \mathbf{\Pi}) = \sum_{k} \pi_k \mathbf{Ber}_D(\mathbf{x}; \mathbf{p}_k)$$

Si rimanda alla letteratura [Fd22] per ulteriori approfondimenti in merito al lavoro in questione.

Capitolo 3

Conoscenza Incompleta: Logica e Applicazione di EM

Il ragionamento in presenza di incertezza, conosciuto anche come ragionamento probabilistico [AM19], è un campo di studi che si occupa di come prendere decisioni o trarre conclusioni quando non si ha la certezza assoluta riguardo alle informazioni disponibili. Questo è un problema che si presenta in molte situazioni della vita reale, sia in contesti personali che professionali, per esempio in campo medico [Mye00], quando si valuta l'efficacia di un farmaco o di un trattamento, non si ha sempre la certezza assoluta riguardo ai risultati attesi.

L'incertezza può essere causata principalmente da due ragioni [Mur22]:

- la difficoltà nel comprendere i meccanismi generatori dei dati, chiamata *incertezza epistemologica* o *incertezza del modello*;
- l'incertezza intrinseca legata ai dati, definita come *incertezza aleatoria* o *incertezza dei dati*.

L'incertezza può avere un impatto significativo sulle prestazioni dei sistemi di intelligenza artificiale (IA) e sulla loro capacità di prendere decisioni affidabili. In particolare, gli effetti dell'incertezza nell'IA possono includere:

- prestazioni sub-ottimali, ovvero l'incertezza può influire negativamente sulle prestazioni dei modelli di IA, rendendoli meno precisi e affidabili nella loro capacità di effettuare previsioni e prendere decisioni;

- risultati imprecisi, in quanto l'incertezza può portare a risultati imprecisi e inaccurati. Ad esempio, se un modello di IA è incerto su una determinata previsione, potrebbe produrre un risultato che non rispecchia completamente la realtà;

- difficoltà di interpretazione, ossia l'incertezza può rendere difficile interpretare i risultati dei modelli di IA e capire il motivo per cui un modello ha effettuato una determinata previsione o preso una determinata decisione.

In questo capitolo vogliamo focalizzarci sulla fase di apprendimento dei modelli, nello specifico delle problematiche che legano questa ai dati presenti nel dataset. A tal proposito, in fase di progettazione bisogna avere un occhio di riguardo nei confronti dei dati disponibili, in quanto ricoprono un ruolo fondamentale. Di conseguenza si deve tener conto di diversi *fattori* riguardanti la qualità del dataset rispetto al modello, alcuni dei quali sono:

- Dimensione del dataset: se il dataset è troppo piccolo, il modello potrebbe non essere in grado di apprendere abbastanza informazioni per generalizzare correttamente. D'altra parte, se il dataset è troppo grande, il modello potrebbe richiedere troppo tempo e risorse per l'addestramento;

- Sovradattamento/sottoadattamento: il sovradattamento (*overfitting*) si verifica quando un modello, particolarmente complesso, si adatta troppo ai dati all'interno di un dataset e quindi non è in grado di generalizzare correttamente su nuovi dati. Il sottoadattamento (*underfitting*), d'altra parte, si verifica quando il modello non è abbastanza complesso per rappresentare correttamente i dati di addestramento;

- Dati mancanti: i dati mancanti possono essere un problema se non sono gestiti correttamente. Ciò potrebbe comportare una perdita di informazioni importanti per l'addestramento del modello.

3.1 Dati Mancanti e Metodi per la Loro Gestione

L'incertezza può sorgere quando non si dispone di informazioni complete o accurate su una determinata situazione o fenomeno [Rub76].

Gli effetti dei dati mancanti possono variare a seconda del tipo e della quantità di dati mancanti e del metodo di gestione utilizzato. Alcuni effetti possono includere una perdita di precisione [Bod08] o impattare su varianza e bias [Ayi+19].

I motivi per cui i dati sono mancanti (o incompleti) giocano un ruolo importante su come, poi, vengono trattati [How07], e per questo si definiscono tre tipologie:

- *Missing Completely at Random* (MCAR): come si può dedurre dal nome stesso, non esiste nessuna correlazione con nessun valore di altre variabili per cui i dati sono mancanti;

- *Missing at Random* (MAR): in questo caso i dati sono mancanti se la probabilità che un dato sia mancante, per una determinata variabile Y, non è funzione del suo stesso valore rispetto le altre variabili. Formalmente [How07]: $p(Ymissing \mid Y, X) = p(Ymissing \mid X)$;

- *Missing Not a Random* (MNAR): questi sono dati che non sono definibili ne come MCAR ne come MAR, quindi si definisce una relazione tra il valore mancante della variabile e il motivo per cui questo è mancante.

Nel corso degli anni sono stati sviluppati diversi metodi per trattare questo problema che generalmente possono essere divisi in quattro gruppi [LR02]:

- *Metodi sulle sole unità osservate*: questo metodo consiste nel non utilizzare i dati mancanti nell'analisi e quindi limitarsi ad esaminare solo i dati completi. Questo metodo può portare a una perdita di potere statistico, in quanto si sta utilizzando solo una parte del campione disponibile. Inoltre, l'omissione dei dati mancanti può influenzare il bias se la mancanza di dati è correlata con le variabili di interesse. Sarebbe quindi opportuno utilizzare questo approccio solo quando i dati sono MCAR.

- *Imputazione dei dati mancanti*: questo metodo consiste nell'assegnare dei valori ai dati mancanti sulla base delle informazioni disponibili. Esistono diverse tecniche di imputazione, tra cui l'imputazione *esplicita* e l'imputazione *implicita*.

- *Procedure di ponderazione*: Le procedure di ponderazione sono un metodo di gestione dei dati mancanti che prevede l'assegnazione di un peso ai dati completi in modo da compensare la mancanza di dati mancanti. In pratica, si utilizzano i dati completi per stimare i pesi che vengono poi utilizzati per *riscalare* i dati completi e ridurre il bias dovuto alla mancanza di dati mancanti. È importante notare che le procedure di ponderazione possono essere efficaci solo se la mancanza di dati è casuale. Se la mancanza di dati è correlata con le variabili di interesse, allora le procedure di ponderazione possono portare a un bias ancora maggiore.

- *Metodi basati su modelli*: In questo caso, si utilizzano le informazioni disponibili per prevedere i valori mancanti e imputarli nel dataset. Uno dei metodi più comuni basati su modelli è il metodo di massima verosimiglianza [Lit19], che cerca di trovare i valori mancanti che massimizzano la verosimiglianza del modello costruito sui dati completi.

La scelta del metodo di gestione dei dati mancanti dipende dalla natura dei dati mancanti, dal tipo di analisi che si vuole eseguire e dal livello di precisione

richiesto. È importante ricordare che non esiste un metodo di gestione dei dati mancanti che sia sempre il migliore, ma piuttosto è necessario scegliere il metodo più adatto al caso specifico.

3.2 Algoritmo Expectation-Maximization

L'algoritmo *Expectation-Maximization* (EM) è un metodo iterativo per stimare i parametri di un modello di probabilità, quando alcuni dei dati necessari per l'analisi sono mancanti o incompleti, definendosi, così, come un approccio generativo basato sui modelli per il trattamento di questi ultimi.

L'idea conduttrice dell'algoritmo EM è stata introdotta inizialmente da Dempster, Laird e Rubin nel 1977 [DLR77], e da allora tale metodo è stato applicato con successo in una vasta gamma di domini, tra cui il riconoscimento del parlato, la classificazione delle immagini, l'analisi dei dati biologici e molte altre applicazioni.

Combinato con un classificatore di tipo *Naive Bayes* è possibile compiere *soft clustering*, quindi, a partire dai dati un classificatore di questo tipo viene costruito avendo una variabile per ogni feature nei dati e *variabili nascoste* per ogni classe [PM17]. L'algoritmo EM consiste in una procedura iterativa in cui si alternano due fasi, la fase di *Expectation* e la fase di *Maximization*. Possiamo riferirci alla decomposizione di cui sopra per dimostrare come questa possa essere utile nella massimizzazione della log-verosimiglianza.

- Nella fase di *Expectation*, l'algoritmo cerca di stimare il valore delle distribuzioni di probabilità delle variabili latenti mancanti a partire dai dati noti e dai parametri del modello, quindi, viene calcolata la verosimiglianza data la stima attuale dei parametri del modello.

- Nella fase di *Maximization*, l'algoritmo stima i parametri del modello a partire dalle stime delle variabili latenti calcolate nella fase precedente.

Questi due passaggi vengono eseguiti in modo iterativo fino a quando la verosimiglianza non converge a un massimo locale, restituendo il modello probabilistico utile alla classificazione dei nuovi esempi.

L'algoritmo EM è particolarmente utile quando i dati mancanti sono MAR o MCAR, ovvero quando la probabilità di avere dati mancanti non dipende dal valore mancante stesso, ma solo da altre variabili osservate.

3.2.1 Formalizzazione dell'Algoritmo

Considerando un modello probabilistico dove definiamo le variabili osservate come X e le variabili *nascoste* con Z, la probabilità congiunta $p(\mathbf{X}, \mathbf{Z} \mid \theta)$ viene *controllata* dal parametro θ [Bis06]. L'obiettivo dell'algoritmo è la massimizzazione della funzione di verosimiglianza [DLR77], data da:

$$p(\mathbf{X} \mid \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \theta)$$

Si ritiene che stimare direttamente $p(\mathbf{X} \mid \theta)$, quindi l'ottimizzazione del MLE sui soli dati osservati, sia un'operazione troppo laboriosa, differentemente dallo stimare la funzione di massima verosimiglianza sui dati completi, ovvero $p(\mathbf{X}, \mathbf{Z} \mid \theta)$. Fatta questa premessa possiamo introdurre il concetto di distribuzione sui dati nascosti, ovvero $q(\mathbf{Z})$ e quindi notare che per ogni scelta di quest'ultima è possibile effettuare la seguente decomposizione per la funzione di verosimiglianza:

$$\log p(\mathbf{X} \mid \theta) = \mathcal{L}(q, \theta) + \mathrm{KL}(q \| p)$$

denotando con KL la distribuzione di Kullback-Leibler, quindi:

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{q(\mathbf{Z})} \right\}$$

$$\mathrm{KL}(q \| p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{Z} \mid X, \theta)}{q(\mathbf{Z})} \right\}$$

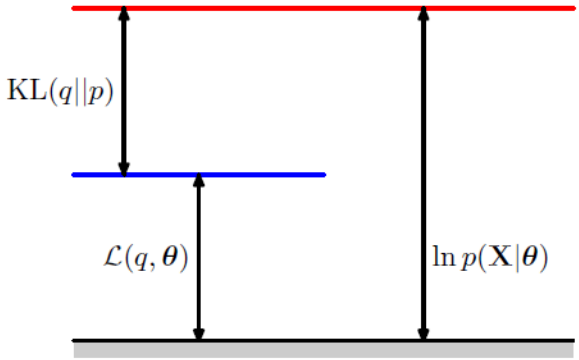


Figura 3.1: Illustrazione della decomposizione dove si nota che se $\mathrm{KL}(q\|p) \geq 0$ allora $\mathcal{L}(q, \theta)$ è limite inferiore di $\log p(\mathbf{X} \mid \theta)$ [Bis06]

Dove consideriamo $\mathcal{L}(q, \theta)$ un funzionale su $q(\mathbf{Z})$ e θ. Per la regola del prodotto delle probabilità, semplifichiamo $\log p(\mathbf{X}, \mathbf{Z} \mid \theta)$:

$$\log p(\mathbf{X}, \mathbf{Z} \mid \theta) = \log p(\mathbf{Z} \mid \mathbf{X}, \theta) + \log p(\mathbf{X} \mid \theta)$$

dunque lo si può sostituire in $\mathcal{L}(q, \theta)$ che ci porta ad un risultato tale che ci permette di eliminare $\mathrm{KL}(q\|p)$ ed in aggiunta ci fornisce il logaritmo della verosimiglianza $\log p(\mathbf{X} \mid \theta)$. Importante è che la distribuzione $q(\mathbf{Z})$ sia normalizzata in modo tale che la somma ci dia come risultato 1.

Definiamo un valore iniziale per il parametro θ^{old}, nel passo E, il limite inferiore $\mathcal{L}(q \mid \theta^{old})$ viene massimizzato rispetto $q(\mathbf{Z})$. La soluzione a questa massimizzazione proviene dal fatto che $\log p(\mathbf{X} \mid \theta^{old})$ non dipende da $q(\mathbf{Z})$, quindi, il valore di $\mathcal{L}(q \mid \theta^{old})$ crescerà quanto più la divergenza di Kullback-Leibler diminuirà come mostrato in figura 3.2.

Successivamente, al passo M, il limite inferiore $\mathcal{L}(q \mid \theta)$ sarà massimizzato rispetto θ ottenendo così un nuovo valore θ^{new}. Di conseguenza il limite inferiore aumenterà e, quindi, anche la funzione di log-verosimiglianza.

Va notato che, dato che si stima la distribuzione q usando θ^{old} poi massimizzato in M come θ^{new}, la divergenza di KL non sarà mai 0 e di conseguenza q mai

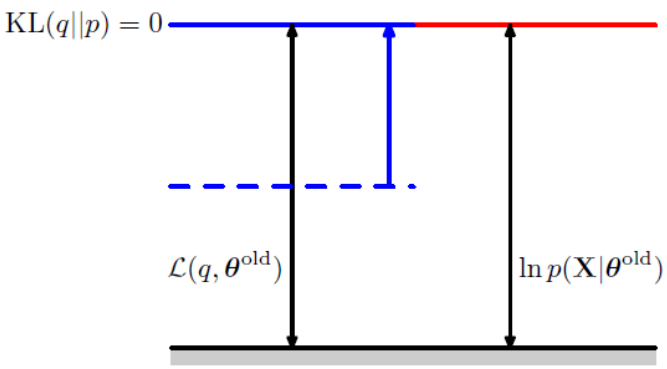


Figura 3.2: Passo E. Si può notare che ponendo la distribuzione q uguale alla distribuzione a posteriori per θ^{old} porterà il limite inferiore ad essere uguale alla log-verosimiglianza. Notare KL = 0 [Bis06]

uguale a $p(\mathbf{Z} \mid X, \theta^{new})$.

3.2.2 EM per Dataset di Esempi Indipendenti e Identicamente Distribuiti

Consideriamo $\mathbf{X}$ un dataset di N istanze $\{\mathbf{x}^t\}$ indipendenti e identicamente distribuite (i.i.d.) e $\mathbf{Z}$ corrispondente a N variabili latenti $\{\mathbf{z}^t\}$ dove $t = 1, ..., N$. Per l'assunzione di i.i.d. abbiamo che

$$p(\mathbf{X}, \mathbf{Z}) = \prod_t p(\mathbf{X}, \mathbf{z}^t)$$

marginalizzando su $\{\mathbf{z}_n\}$:

$$p(\mathbf{X}) = \prod_t p(\mathbf{x}^t)$$

Semplificando attraverso la regola della somma e prodotto di probabilità, otteniamo la probabilità a posteriori, o responsabilità, del passo E:

$$p(\mathbf{Z} \mid \mathbf{X}, \theta) = \frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{\sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \theta)} = \frac{\prod_{t=1}^{N} p(\mathbf{x}^t, \mathbf{z}^t \mid \theta)}{\sum_Z \prod_{t=1}^{N} p(x^t, z^t \mid \theta)} = \prod_{t=1}^{N} p(\mathbf{z}^t \mid \mathbf{x}^t, \theta)$$

Inoltre, la probabilità congiunta $p(\mathbf{X}, \mathbf{Z} \mid \theta)$ si fattorizza su tutti i dati, quindi, sfruttando questa proprietà possiamo stimare ad ogni ciclo iterativo i parametri

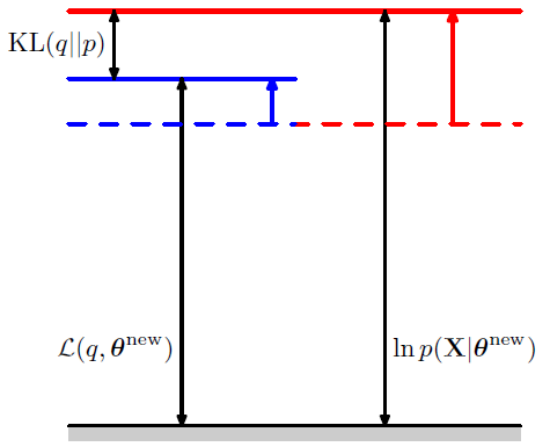


Figura 3.3: Fissato $q(\mathbf{Z})$ il limite inferiore è massimizzato rispetto θ producendo un nuovo θ^{new}. Dato che KL è un non-negativo, la log-verosimiglianza cresce più di quanto possa fare il limite inferiore [Bis06]

su un solo punto alla volta piuttosto che su tutto il dataset. Prendendo in esempio il caso gaussiano, dove t denota un singolo punto e i valori delle corrispondenti responsabilità precedenti e successive sono definite come $\gamma^{old}(z_k^t)$ e $\gamma^{new}(z_k^t)$, il passo M stima le medie in questo modo:

$$\mu_k^{new} = \mu_k^{old} + \left(\frac{\gamma^{new}(z_k^t) - \gamma^{old}(z_k^t)}{N_k^{new}} \right) (\mathbf{x}^t - \mu_k^{old})$$

dove

$$N_k^{new} = N_k^{old} + \gamma^{new}(z_k^t) - \gamma^{old}(z_k^t)$$

rende questa versione *incrementale* [Bis06], quindi assicura una convergenza più rapida.

3.3 EM: Stima dei Parametri di Mixture Model

Come già accennato nel capitolo 2, uno degli scopi di questo lavoro è ricercare uno strumento a supporto del modello proposto nel momento in cui ci si trovi in presenza di dati mancanti o incompleti. Questo è utile in quanto aiuta a sfruttare

tutti i dati presenti con l'obiettivo, quindi, di migliorare la fase di addestramento. Rispetto all'approccio bayesiano, già menzionato, questo approccio è più orientato alla stima delle probabilità dei valori mancanti, *generando* nuove informazioni partendo da quelle presenti.

La struttura principale dell'algoritmo rimane invariata, quindi successivamente all'inizializzazione, succederanno ricorsivamente i due passi di ricalcolo delle responsabilità e massimizzazione dei parametri [THA97]. In seguito alla fase di addestramento verranno effettuate le predizioni attraverso l'utilizzo dei parametri massimizzati.

Riferendoci alla letteratura [THA97; Mur22] possiamo adottare un approccio che si basa su misture di modelli gaussiani gerarchici [THA97] e quindi adattarlo al modello MBM.

In maniera preliminare si considera $G(x; c_i, \sigma_i)$ la densità normale centrata su c_i e con un vettore di parametri scalari σ_i. Assumendo di avere N dati di training $\{\mathbf{x^t}, z^t\}_{t=1}^N$ e z^t la nostra variabile nascosta, quindi, non conosciuta, partiamo da una *log-likelihood* calcolata come segue:

$$\mathcal{L} = \sum_{t=1}^{N} \log \left[\sum_{k=1}^{K} \hat{P}(z = k) G(\mathbf{x^t}; \hat{c}_k, \hat{\sigma}_k) \right]$$

Il passo E:

$$\hat{P}(z = k \mid \mathbf{x^t}) = \frac{\hat{P}(z = k) G(\mathbf{x^t}; \hat{c}_k, \hat{\sigma}_k)}{\sum_{l=1}^{K} \hat{P}(z = l) G(\mathbf{x^t}; \hat{c}_l, \hat{\sigma}_l)}$$

Il passo M che massimizza i parametri di probabilità a priori $\hat{P}(z = k)$ e le componenti (della dimensione di ogni feature x_i) $\hat{c}$, $\hat{\sigma}$ rispettivamente *media* e *deviazione standard*:

$$\hat{P}(z = k) = \frac{1}{N} \sum_{t=1}^{N} \hat{P}(z = k \mid \mathbf{x^t}),$$

$$\hat{c}_{ki} = \frac{\sum_{t=1}^{N} \hat{P}(y = k \mid \mathbf{x^t}) x_i^t}{\sum_{t=1}^{N} \hat{P}(y = k \mid \mathbf{x^t})},$$

$$\hat{\sigma}_{ki}^2 = \frac{\sum_{t=1}^{N} \hat{P}(z = k \mid \mathbf{x^t})(\hat{c}_{ki} - x_i^t)^2}{\sum_{t=1}^{K} \hat{P}(z = k \mid \mathbf{x^t})}$$

Passando a MBM, si assume la disponibilità di un *training set* completo $\mathbf{T} = \langle \mathbf{X}, \mathbf{y} \rangle = \{(\mathbf{x}^t, y^t)\}_{t=1}^N$ dove y^t definisce la classe non conosciuta, otteniamo la *log-likelihood* così:

$$\mathcal{L} = \log p(T \mid \Pi) = \sum_{t=1}^{N} \log[\hat{\pi}\mathbf{Ber}_D(\mathbf{x}^t; \hat{p}_1) + (1 - \hat{\pi})\mathbf{Ber}_D(\mathbf{x}^t; \hat{p}_0)]$$

Il passo E calcola le *responsabilità* attraverso:

$$\hat{P}(y \mid \mathbf{x}^t) = \frac{\hat{\pi}\mathbf{Ber}_D(\mathbf{x}^t; \hat{p}_y)}{\hat{\pi}\mathbf{Ber}_D(\mathbf{x}^t; \hat{p}_1) + (1 - \hat{\pi})\mathbf{Ber}_D(\mathbf{x}^t; \hat{p}_0)}$$

E nel passo M si calcolano i parametri in questo modo:

$$\hat{\pi}^y = \hat{P}(y) = \frac{1}{N} \sum_{t=1}^{N} \hat{P}(y \mid \mathbf{x}^t)$$

$$\hat{p}_{yi} = \frac{\sum_t \hat{P}(y \mid \mathbf{x}^t) x_i^t}{\sum_t \hat{P}(y \mid \mathbf{x}^t)}$$

Decisioni di progetto riguardanti inizializzazione e criterio di stop verranno affrontate nel prossimo capitolo.

3.4 EM e Incompletezza del Dataset

Come già detto in precedenza, i casi di incertezza dei dati possono riguardare in generale la mancanza di informazioni che possono risultare utili durante la fase di apprendimento. Più nello specifico, rispetto a quanto detto nella precedente sezione, l'incertezza può colpire non solo la mancanza di informazioni riguardo l'appartenenza di alcuni individui, ma anche gli individui stessi e le informazioni che racchiudono al loro interno. In questa sezione dimostreremo come l'algoritmo

EM risulta utile anche in un contesto in cui i vettori $\mathbf{x}$, corrispondenti ai singoli individui, presentano al loro interno valori incerti riferiti alle feature. Definiamo con $\mathbf{x}^+$ la proiezione di $\mathbf{x}$ su un sottoinsieme di variabili di input $X^+ \subseteq \{x_1, ..., x_D\}$ con valori conosciuti, rispettivamente $X^- = \{x_1, ..., x_D\} \setminus X^+$. Formalmente diremo che è possibile predire il valore di ognuno di questi valori sconosciuti $x_u \in X^-$ a partire da quelli conosciuti $x_i \in X^+$, definendo così la distribuzione di probabilità condizionata di x_u dati $\mathbf{x}^+$:

$$P(x_u \mid \mathbf{x}^+) = \sum_{y \in \mathbb{B}} P(x_u, y \mid \mathbf{x}^+) = \frac{\sum_{y \in \mathbb{B}} \mathbf{Ber}(x_u; p_{yu}) \pi^y P(\mathbf{x}^+ \mid y)}{\sum_{y' \in \mathbb{B}} \pi^{y'} P(\mathbf{x}^+ \mid y')}$$

da qui, il valore atteso:

$$\mathbf{E}[x_u \mid \mathbf{x}^+] = \frac{\sum_{y \in \mathbb{B}} p_{yu} \pi^y P(\mathbf{x}^+ \mid y)}{\sum_{y' \in \mathbb{B}} \pi^{y'} P(\mathbf{x}^+ \mid y')}$$

Per concludere, un'applicazione di EM in questo senso verrà mostrata nel capitolo 4, riguardante il caso di studio.

Capitolo 4

Esperimenti

In questo capitolo saranno presentati rispettivamente le librerie utilizzate per costruire e testare i modelli e quindi i dataset definiti attraverso le ontologie coinvolte. Successivamente saranno descritti il progetto e l'implementazione della valutazione empirica di tali modelli su tali dataset. Infine saranno discussi i risultati ottenuti.

4.1 Implementazione dei Modelli e Ontologie

I modelli descritti nei precedenti capitoli fanno leva su librerie già presenti sotto forma di API, definite in linguaggio Python. Parliamo nello specifico di `Owlready2` [Lam17], una libreria utilizzata per gestire grafi di conoscenza ed effettuare ragionamento in quanto è incorporato al suo interno una versione del *reasoner* Pellet, responsabile della codifica di cui nel capitolo 2. Un'altra libreria utilizzata è `Scikit-learn`, ampiamente utilizzata in ambito Data Science, in quanto offre, fra gli altri, anche una serie di modelli basati su feature bernoulliane. Oltre a questo, essa fornisce anche una serie di strumenti utili alla valutazione dei risultati e misure e funzionalità di partizionamento dei dati per diversi tipi di cross-validation che facilitano il disegno sperimentale.

Tabella 4.1: Numeri di classi e proprietà, feature generate ed estratte per ciascuna ontologia

grafi di conoscenza	classi	obj.prop.	generate	selezionate
Financial	59	16	75	13
NTNames	47	27	82	19
LUBM	43	25	75	13
KRKZeroOne	7	37	44	28

Le ontologie impiegate, definite tutte in linguaggio OWL, sono le seguenti: Financial, New Testament Names (NTNames), KRKZeroOne e LUBM, quest'ultima generata usando il LeHeigh University Benchmark[1]. Nel seguito verranno descritte le fasi di pre-processing, feature selection e codifica booleana.

4.2 Set-up Sperimentale

Lo scopo di questo lavoro, come detto in precedenza, è verificare i benefici dell'adozione dell'algoritmo EM nell'apprendimento di modelli bernoulliani. Nello specifico verranno testati due modelli descritti nel capitolo 2, rispettivamente il modello MBM e una sua estensione che include l'implementazione di tale algoritmo (MBM-EM) per addestrare il modello anche su dati per i quali non sono noti i valori per alcune feature (*appartenenza*) senza pre-imputazione degli stessi. Oltre questi verrà testato anche un modello di tipo regressione logistica come base generale di confronto rispetto i modelli di cui sopra. Ontologie, concetti target e codice sorgente degli esperimenti è pubblicamente disponibile [2]

Per ogni ontologia è prevista una fase di estrazione delle feature, che coinvolge classi e proprietà degli oggetti, ovvero le relazioni tra classi. Per quanto riguarda

[1] http://swat.cse.lehigh.edu/projects/lubm/

[2] https://github.com/THESCREAMINGMONKEY/MBM-EM.git

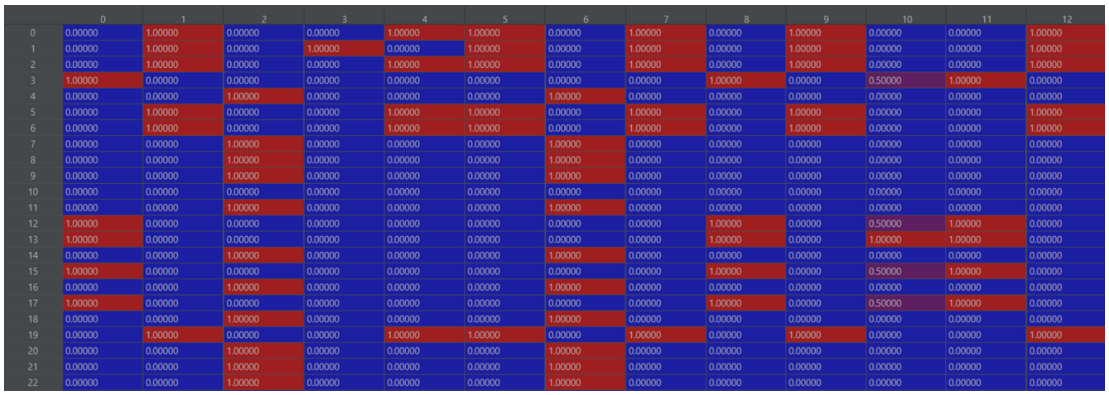


Figura 4.1: Matrice composta da individui sulle righe e le corrispondenti feature sulle colonne dopo la feature selection (Estratto da Financial)

la codifica booleana alla base dei modelli testati, viene effettuata una fase di feature selection delle feature più *significative* ottenute tramite il confronto rispetto ad una tolleranza sulla varianza, determinando quelle più informative. Queste corrispondo ai nodi al livello di input di ogni rete. Nella Tabella 4.1 vengono mostrati gli effetti di queste fasi su ognuno dei grafi di conoscenza considerati.

A seguito della codifica, i valori di ogni individuo saranno rappresentati da valori discreti, precisamente booleani $[1, 0]$, che determinano rispettivamente l'appartenenza o la non appartenenza alla classe [Figura 4.1] o la presenza di una feature per un individuo [Figura 4.2]. I valori mancanti o incompleti verranno definiti rispettivamente con -1 per l'appartenenza e 0.5 per le singole feature sugli individui, valori che determinano l'incertezza.

Per quanto riguarda i problemi di classificazione, in maniera preliminare vengono generati 10 concetti target in maniera casuale assicurandosi che per ognuno di essi ci siano almeno 10 individui la cui appartenenza risulta *positiva* e altri 10 *negativa*. Per ogni concetto target viene effettuata casualmente una cross-validation con 10 split stratificata, dividendo individui positivi, negativi e *incerti*.

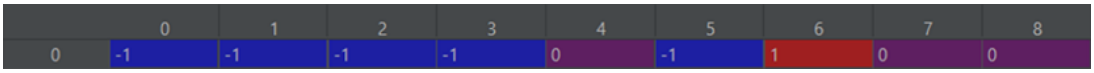


Figura 4.2: Vettore contenente l'appartenenza alla classe per ogni individuo (Estratto da Financial)

4.2.1 MBM-EM: Dati Mancanti

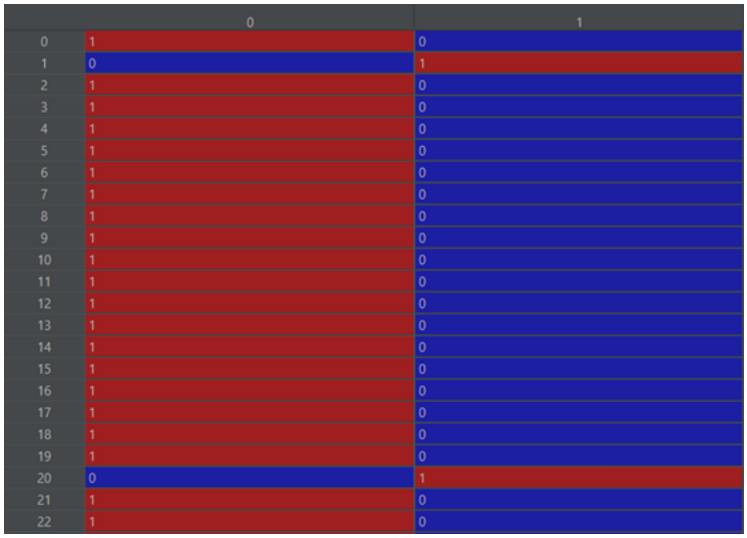
L'algoritmo calcola per ogni individuo la distribuzione di appartenenza ad una determinata classe, in particolar modo di quelli la cui etichetta y non è nota. Durante la costruzione del modello sono state prese diverse decisioni in merito all'inizializzazione dei parametri e alla trattazione dei dati incompleti o mancanti rispetto a quelli noti.

Inizializzazione

Per una scelta di progetto si è deciso di sfruttare i dati che sono già noti per calcolare inizialmente i parametri, ovvero la probabilità a priori e quella condizionata attraverso il calcolo delle responsabilità. Nello specifico, dato che siamo in presenza di variabili bernoulliane, ci si è concentrati sul determinare l'appartenenza ($C = 1$) o la non appartenenza ($C = 0$) ad una determinata classe in base ai diversi problemi target affrontati. Ragion per cui nel caso del calcolo delle responsabilità degli individui, vengono utilizzate già le etichette disponibili (quindi non calcolate), così da produrre una matrice contenente tutte le distribuzioni note di appartenenza individui-classe [Figura 4.3]. Successivamente vengono utilizzate queste per la massimizzazione dei parametri di cui sopra, così da assicurare risultati migliori rispetto alla totale incertezza.

Iterazioni

Il modello presenta i concetti base dell'algoritmo EM, utilizzando nel ciclo di apprendimento sia i dati noti che non (o incompleti). In particolare, nel passo E,



	0	1
0	1	0
1	0	1
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0
10	1	0
11	1	0
12	1	0
13	1	0
14	1	0
15	1	0
16	1	0
17	1	0
18	1	0
19	1	0
20	0	1
21	1	0
22	1	0

Figura 4.3: Matrice delle responsabilità di ogni individuo la cui classe è nota in fase di inizializzazione (Estratto da Financial)

nel caso dei dati le cui appartenenze sono note viene imposto il valore 1 di appartenenza alla classe, per tutti gli altri sono calcolate le responsabilità utilizzando i parametri calcolati al passo M. Notiamo alcuni valori indicare precisamente 1 che risultano essere le appartenenze già note imposte in fase di costruzione della matrice [Figura 4.4]. Come criterio di stop viene utilizzata la distanza euclidea ottenuta tramite la differenza delle sommatorie sulla matrice delle responsabilità, oltre che un numero massimo di iterazioni possibili nel caso non ci sia convergenza. Si è deciso di utilizzare questo criterio per due ragioni: la prima consiste nel fatto che usando la log-likelihood si otteneva una convergenza troppo veloce e da questo dei risultati meno accurati in quanto quest'ultima cresceva troppo velocemente, la seconda è data dalla disponibilità di dati già normalizzati, fondamentale per l'utilizzo di metriche basate sulla distanza vettoriale come in questo caso.

	0	1
0	0.83117	0.16883
1	0.83303	0.16697
2	0.83117	0.16883
3	0.93864	0.06136
4	1.00000	0.00000
5	0.83117	0.16883
6	0.00000	1.00000
7	1.00000	0.00000
8	1.00000	0.00000
9	1.00000	0.00000
10	0.92222	0.07778
11	1.00000	0.00000
12	0.93864	0.06136
13	0.93980	0.06020
14	1.00000	0.00000
15	0.93864	0.06136
16	1.00000	0.00000
17	0.93864	0.06136
18	1.00000	0.00000
19	0.83117	0.16883
20	1.00000	0.00000
21	1.00000	0.00000
22	1.00000	0.00000

Figura 4.4: Matrice delle responsabilità di ogni individuo per ogni classe, nota e no, in fase di training (Estratto da Financial)

4.2.2 MBM-EM: Dati Incompleti

Un ulteriore applicazione dell'algoritmo EM in questo contesto è possibile anche sui dati incompleti, come già detto nella sezione 3.4. L'obiettivo è quello di stimare per ogni individuo $\mathbf{x}$ le feature mancanti con lo scopo di migliorare anche i risultati dell'applicazione di EM descritta nella precedente sezione.

Dettagli Implementativi

L'applicazione di questa logica è racchiusa all'interno di un solo metodo che verrà richiamato all'inizio del modello, quindi precedentemente anche all'applicazione di EM per stimare le classi degli individui incerti.

Inizialmente tutti i dati incompleti rappresentati dal valore 0.5 saranno convertiti nel valore NaN (Not a Number) [Figura 4.5], questa decisione è stata presa per migliorare la gestione degli stessi dal punto di vista identificativo. Più nello specifico, alcune librerie implementano metodi adatti proprio a questi valori.

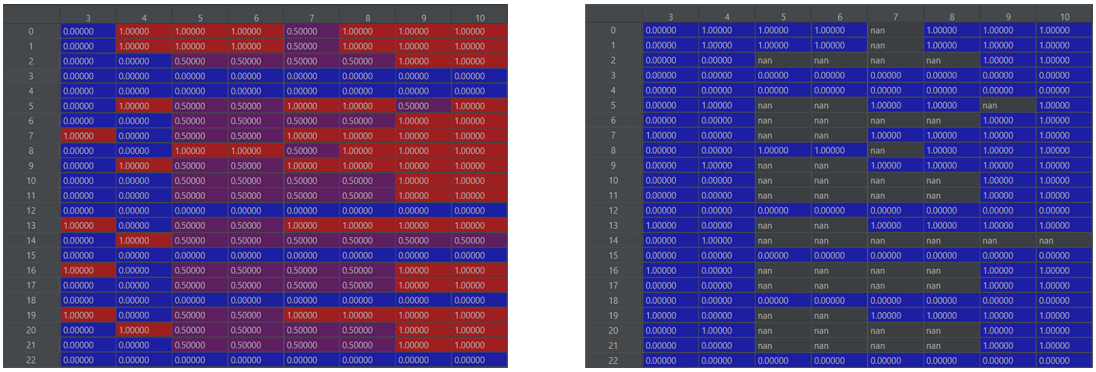
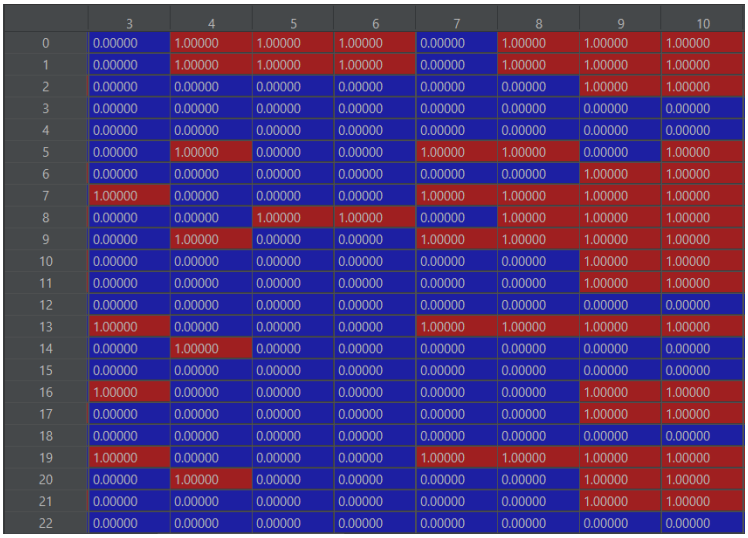


Figura 4.5: Trasformazione dei valori incerti (0.5) in valori NaN (Estratto da KRKZeroOne)

Successivamente nel passo E vengono calcolati i valori attesi per i dati mancanti. Nello specifico, per ogni colonna della matrice (per ogni feature) viene controllato se ci sono dei valori mancanti. Se ci sono dei valori mancanti, allora si esegue il calcolo della probabilità di avere quel valore mancante pari a 1, considerando la probabilità empirica della feature i (che rappresenta la probabilità di avere un valore pari a 1 per quella feature), e il numero di valori mancanti nella colonna i. Questa probabilità viene quindi utilizzata per calcolare il valore atteso dei valori mancanti utilizzando i valori noti della stessa feature. In particolare, il valore atteso viene calcolato come prodotto della probabilità di avere il valore mancante pari a 1 per quella feature, per il rapporto tra la somma dei valori noti per quella feature e il numero di valori noti (in modo da normalizzare i valori).

Nel passo M inizialmente si calcola la media dei valori attesi per ogni feature, ovvero la probabilità di ogni feature di essere 1.

In seguito si aggiornano i valori mancanti. In particolare, per ogni cella il cui valore è mancante, cioè si genera un valore casuale con distribuzione binomiale con parametro di probabilità pari al valore atteso corrispondente alla cella mancante. In questo modo si stima il valore mancante con la probabilità stimata durante il passo E.



	3	4	5	6	7	8	9	10
0	0.00000	1.00000	1.00000	1.00000	0.00000	1.00000	1.00000	1.00000
1	0.00000	1.00000	1.00000	1.00000	0.00000	1.00000	1.00000	1.00000
2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	0.00000	1.00000	0.00000	0.00000	1.00000	1.00000	0.00000	1.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
7	1.00000	0.00000	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000
8	0.00000	0.00000	1.00000	1.00000	0.00000	1.00000	1.00000	1.00000
9	0.00000	1.00000	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000
10	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
11	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
12	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
13	1.00000	0.00000	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000
14	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
15	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
16	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
17	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
18	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
19	1.00000	0.00000	0.00000	0.00000	1.00000	1.00000	1.00000	1.00000
20	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
21	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000
22	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Figura 4.6: Risultato dell'imputazione dei valori mancanti mediante l'algoritmo EM (Estratto da KRKZEROONE)

L'utilizzo di questo algoritmo anche in questo contesto porta a dei risultati migliori in fatto di velocità di convergenza durante l'esecuzione di EM (descritta nella sezione precedente) ed anche di risultati.

4.3 Risultati

Le metriche che sono state prese in considerazione sono *precision*, *recall* e F_1-measure, considerando i problemi come binari (tuttavia i set di test non contenevano esempi non etichettati, ovvero individui la quale appartenenza al concetto target non può essere determinata tramite ragionamento).

La Tabella 4.2 riporta i risultati medi sui dieci problemi di classificazione per ontologia considerati. In questa troviamo le ontologie nella colonna corrispondente ai grafi di conoscenza e per ognuna di esse sono mostrate le tre misure prese in considerazione. Troviamo successivamente le colonne corrispondenti ai tre modelli presi in considerazione ed in esse i risultati in forma medie $\pm$ deviazioni

grafi di conoscenza	*misure*	**modelli**		
		MBM	**MBM-EM**	**LREG**
Financial	P	0.993 ± 0.009	0.993 ± 0.009	0.900 ± 0.067
	R	0.975 ± 0.041	0.981 ± 0.022	0.873 ± 0.094
	F_1	0.981 ± 0.029	0.983 ± 0.025	0.822 ± 0.129
NTNames	P	0.976 ± 0.024	0.981 ± 0.019	0.941 ± 0.074
	R	0.950 ± 0.060	0.978 ± 0.020	0.524 ± 0.436
	F_1	0.958 ± 0.045	0.973 ± 0.028	0.491 ± 0.417
LUBM	P	1.000 ± 0.000	1.000 ± 0.000	0.977 ± 0.069
	R	1.000 ± 0.000	1.000 ± 0.000	0.785 ± 0.296
	F_1	1.000 ± 0.000	1.000 ± 0.000	0.778 ± 0.302
KRKZeroOne	P	0.995 ± 0.002	0.999 ± 0.002	0.943 ± 0.092
	R	0.995 ± 0.003	0.999 ± 0.002	0.148 ± 0.243
	F_1	0.995 ± 0.003	0.999 ± 0.002	0.113 ± 0.187

Tabella 4.2: Risultati (medie ± deviazioni standard): precisione (P), richiamo (R), e F_1-measure (F_1)

standard.

Considerando la metrica F_1, questa ci mostra che il modello MBM-EM risulta avere risultati migliori rispetto ad MBM. Il modello esteso (MBM-EM) risulta essere migliore su quasi tutte le ontologie usate per l'esperimento, il cui grado di qualità dipende sostanzialmente dai dati mancanti stessi, in particolare dalla quantità e dal pattern di distribuzione delle feature dei singoli individui. Si ricorda che la stima dei parametri nel modello MBM avviene prendendo in considerazione solo i dati *etichettati* a differenza dell'idea proposta che si pone l'obiettivo di considerare il dataset completo, stimando l'etichetta dei dati incerti. Da questo, però, si evince che in alcuni casi il modello sia esposto ad *outlier* che spesso risultano essere proprio i dati la cui appartenenza alla classe target non è nota,

influendo negativamente durante fase di predizione.

Nel caso di FINANCIAL i risultati risultano essere molto simili tra i modelli, notando solo un piccolo incremento in termini di recall per la versione che implementa EM. Si è notata, in questa ontologia, una bassa quantità di dati incerti (non *etichettati*) per ogni problema affrontato e questo porta i due modelli a funzionare in maniera simile (discorso analogo per LUBM). D'altro canto l'impiego di EM per il calcolo dei valori incerti per le singole feature da sicuramente il miglior contributo in questo caso, completando meglio il dataset e dando modo di sfruttare le informazioni dei dati che nel modello basilare di MBM non hanno influenza.

Discorso differente per quanto riguarda NTNAMES e KRKZEROONE che presentano una mole di dati incerti ed incompleti maggiore rispetto le altre due ontologie, come si può anche notare in Figura 4.5. Queste due ontologie si prestano meglio all'esperimento proposto, dando una visione più completa di come un approccio generativo nei riguardi di dati incerti possa dare risultati migliori rispetto ad un approccio che non prevede una loro gestione.

Le basse prestazioni della regressione logistica sono dovute al fatto che sono state adottate penalizzazioni per produrre modelli più semplici, infatti, questo sottolinea i difetti derivanti dalla bassa interpretabilità della regressione logistica rispetto i primi due più interpretabili. Il modello in questione, comunque, potrebbe mantenere buone prestazioni rispetto ad una adeguata regolarizzazione.

Anche in termini di deviazione standard notiamo come i modelli MBM e MBM-EM risultino essere più *stabili* rispetto la regressione logistica. Da questo possiamo dedurre come i primi due modelli siano meno soggetti all'impatto negativo dei valori anomali, come per esempio la presenza di *outlier*.

Un Esempio di Regola Estratta

In Figura 4.7 viene mostrato un esempio di definizione delle regole per un

IF $C(\mathtt{x})$ $(\pi = 0.82)$ THEN

AND Account $(p = 0.99)$

AND Some_hasOwner_range $(p = 0.96)$

AND Some_hasPermanentOrder_range $(p = 0.89)$

AND Some_hasStatementIssuanceFrequency_range $(p = 0.99)$

...

Figura 4.7: Un esempio di regola probabilistica estratta dall'ontologia Financial

problema target affrontato per l'ontologia Financial, inserendo i nomi di alcune feature estratte tra quelle generate ed omettendo, con lo scopo di rendere più leggibile e semplice l'esempio stesso, le definizioni estratte con probabilità minore di 0.89. Nell'esempio specifico il problema affrontato C è equivalente a ad un problema di classificazione `[From25To35 | Some_hasPermanentOrder_range]` i cui risultati riportano 158 individui positivi, 773 negativi e 69 non specificati. Notiamo quanto più interpretabile risulti la fase di definizione delle regole di classificazione riducendo l'effetto *black-box* dei modelli meno interpretabili.

Capitolo 5

Conclusioni e Sviluppi Futuri

In questo lavoro sono stati presentati metodi che hanno lo scopo di generare classificatori probabilistici come modelli che si basano su funzioni di variabili discrete. Queste possono essere trasformate in regole probabilistiche come già visto, incrementando il livello di interpretabilità del modello stesso rispetto ad una *baseline* meno interpretabile. Dai risultati in Tabella 4.2 si evince come un approccio generativo rispetto alla gestione di dati incerti possa portare a diversi miglioramenti. In particolar modo, l'uso di EM come metodo di imputazione al fine di stimare l'appartenenza alla classe per ogni individuo privo di *etichetta* (nel training set) aiuta a rendere lo stesso più completo, oltre che aumentarne la dimensione e quindi l'utilità in certi casi. Consideriamo EM adattabile anche a casi più specifici: come abbiamo visto è possibile implementarlo anche per stimare valori di feature incerte. Anche in questo contesto si nota come i dati che in precedenza non erano influenti in quanto incerti (quindi non presi in considerazione) tendono a completare ulteriormente il dataset.

EM è un algoritmo che può portare benefici in termini di qualità dei risultati, portando avanti anche i suoi pregi legati alla sua facilità di implementazione e alla sua interpretabilità. Ciononostante soffre di alcuni difetti, come il suo convergere agli ottimi locali.

Si potrebbero, quindi, implementare logiche atte al miglioramento dell'algoritmo stesso, come per esempio attraverso l'implementazione di una logica basata sul *random restart* per portare la convergenza da un ottimo locale ad un ottimo globale.

Concludendo, un'ulteriore possibile estensione riguardo l'interpretabilità può essere l'incorporamento di regole *esistenti* nei modelli probabilistici descritti, come in [THA97] per i modelli gaussiani.

Bibliografia

[AH09] Grigoris Antoniou e Frank van Harmelen. «Web Ontology Language: OWL». In: *Handbook on Ontologies*. A cura di Steffen Staab e Rudi Studer. Springer, 2009, pp. 91–110.

[AM19] Colin Aitken e Dimitris Mavridis. «Reasoning under uncertainty». In: *Evid. Based. Ment. Health* 22.1 (2019), pp. 44–48.

[AS09] Rajendra Akerkar e Priti Sajja. *Knowledge-based systems*. Jones & Bartlett Publishers, 2009.

[Ayi+19] Olawale F. Ayilara et al. «Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry». In: *Health Qual. Life Outcomes* 17.1 (2019), p. 106.

[Baa+07] Franz Baader et al., cur. *The Description Logic Handbook*. 2ª ed. Cambridge University Press, 2007.

[BCV14] Yoshua Bengio, Aaron Courville e Pascal Vincent. *Representation Learning: A Review and New Perspectives*. 2014. arXiv: 1206.5538 [cs.LG].

[Ber09] Michael K. Bergman. *The Open World Assumption: Elephant in the Room*. AI3 Adaptive Information blog. Dic. 2009. URL: https://www.mkbergman.com/852/the-open-world-assumption-elephant-in-the-room/.

[BHS08] Franz Baader, Ian Horrocks e Ulrike Sattler. «Description Logics». In: *Foundations of Artificial Intelligence* 3 (2008), pp. 135–179.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[BL04] Ronald J. Brachman e Hector J. Levesque. *Knowledge representation and reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2004.

[Bod08] Todd E. Bodner. «What improves with increased missing data imputations?» In: *Struct. Equ. Modeling* 15.4 (2008), pp. 651–675.

[BS84] Bruce G. Buchanan e Edward H. Shortliffe. *Rule-based expert systems*. Addison Wesley Longman Publishing, 1984.

[Cha14] Munesh Trivedi Chandra. *A classical approach to artificial intelligence*. Khanna Publishing House, 2014.

[dFE08] Claudia d'Amato, Nicola Fanizzi e Floriana Esposito. «Query answering and ontology population: An inductive approach». In: *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Proceedings 5*. Springer. 2008, pp. 288–302.

[DLR77] Arthur P. Dempster, Nan M. Laird e Donald B. Rubin. «Maximum Likelihood from Incomplete Data via the EM Algorithm». In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38.

[DP97] Pedro M. Domingos e Michael J. Pazzani. «On the Optimality of the Simple Bayesian Classifier under Zero-One Loss». In: *Machine Learning* 29 (1997), pp. 103–130.

[DPJ17] Mauro Dragoni, Maria Poveda-Villalon e Ernesto Jimenez-Ruiz, cur. *OWL: Experiences and directions - reasoner evaluation*. LNCS. Springer, 2017.

[Fd22] Nicola Fanizzi e Claudia d'Amato. «Towards Interpretable Probabilistic Classification Models for Knowledge Graphs». In: *Proceedings of IEEE-SITIS 2022*. IEEE, 2022, pp. 25–31. DOI: 10.1109/SITIS57111.2022.00013. URL: https://ieeexplore.ieee.org/document/10090018.

[GJ93] Zoubin Ghahramani e Michael Jordan. «Supervised learning from incomplete data via an EM approach». In: *Advances in Neural Information Processing Systems*. A cura di J. Cowan, G. Tesauro e J. Alspector. Vol. 6. Morgan-Kaufmann, 1993.

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[HB11] Tom Heath e Christian Bizer. «Linked data: Evolving the web into a global data space». In: *Synth. Lect. Semant. Web Theory Technol.* 1.1 (2011).

[Hec22] David Heckerman. *A Tutorial on Learning With Bayesian Networks*. 2022. arXiv: 2002.00269 [cs.LG].

[Hog+21] Aidan Hogan et al. «Knowledge Graphs». In: *ACM Computing Surveys* 54.4 (2021), pp. 1–37. DOI: 10.1145/3447772. URL: https://doi.org/10.1145%2F3447772.

[How07] David C. Howell. «The Treatment of Missing Data». In: *The SAGE handbook of social science methodology*. A cura di William Outhwaite e Stephen Turner. SAGE Publications, 2007.

[JJ93] Michael I. Jordan e Robert A. Jacobs. «Hierarchical mixtures of experts and the EM algorithm». In: *Neural computation* 2 (1993), pp. 1339–1344.

[Lam17] Jean-Baptiste Lamy. «Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for

biomedical ontologies». In: *Artificial Intelligence in Medicine* 80 (2017), pp. 11–28.

[Lit19] Roderick J.A. Little. *Statistical analysis with missing data*. 3ª ed. Wiley Series in Probability and Statistics. John Wiley & Sons, 2019.

[LR02] Roderick J.A. Little e Donald B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, Inc., 2002.

[McC59] John McCarthy. «Programs with Common Sense». In: *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*. Her Majesty's Stationary Office, 1959, pp. 75–91.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Mur22] Kevin P. Murphy. *Probabilistic Machine Learning*. MIT Press, 2022.

[Mye00] William R. Myers. «Handling missing data in clinical trials: An overview». In: *Drug Inf. J.* 34.2 (2000), pp. 525–533.

[Obi07] Marek Obitko. «Translations between Ontologies in Multi-Agent Systems». Dissertation Thesis. Czech Technical University in Prague Faculty of Electrical Engineering Department of Cybernetics, 2007.

[PM17] David A. Poole e Alan K. Mackworth. «Retrospect and Prospect». In: *Artificial Intelligence*. 2ª ed. Cambridge University Press, 2017. Cap. 16.

[Ret+12] Achim Rettinger et al. «Mining the Semantic Web: Statistical learning for next generation knowledge bases». In: *Data Mining and Knowledge Discovery* 24 (2012), pp. 613–662.

[RN99] Stuart Russell e Peter Norvig. *Artificial intelligence*. 2ª ed. Prentice Hall series in artificial intelligence. Pearson, 1999.

[Rub76] Donald B. Rubin. «Inference and Missing Data». In: *Biometrika* 63.3 (1976).

[SB75] Edward H. Shortliffe e Bruce G. Buchanan. «A model of inexact reasoning in medicine». In: *Math. Biosci.* 23.3-4 (1975), pp. 351–379.

[Sch15] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». In: *Neural Networks* 61 (2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: https://doi.org/10.1016%2Fj.neunet.2014.09.003.

[SCL22] Yuxiang Sun, Seok-Ju Chun e Yongju Lee. «Learned semantic index structure using knowledge graph embedding and density-based spatial clustering techniques». In: *Appl. Sci.* 12.13 (2022). DOI: 10.3390/app12136713.

[Sow00] John F. Sowa. *Knowledge representation: Logical, philosophical, and computational foundations*. Course Technology Ptr, 2000.

[Sow83] John F. Sowa. *Conceptual structures*. The Systems programming series. Longman Higher Education, 1983.

[Sow92] John F. Sowa. «Semantic networks». In: *Encyclopedia of artificial intelligence* 2 (1992), pp. 1493–1511.

[THA97] Volker Tresp, Jürgen Hollatz e Subutai Ahmad. «Representing probabilistic rules with networks of gaussian basis functions». In: *Machine Learning* 27 (1997), pp. 173–200.

[TL91] Steven G. Tuthill e Susan T. Levy. *Knowledge-based Systems: A Manager's Perspective*. TAB Professional e Reference Books, 1991.

[Tur90] Efraim Turban. «Expert systems-based robot technology». In: *Expert Systems* 7.2 (1990), pp. 102–110.

[Wan+17] Quan Wang et al. «Knowledge graph embedding: A survey of approaches and applications». In: *IEEE Trans. Knowl. Data Eng.* 29.12 (2017), pp. 2724–2743.

[Zha+18] Chenwei Zhang et al. «On the generative discovery of structured medical knowledge». In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. A cura di Yike Guo e Faisal Farooq. ACM, 2018, pp. 2720–2728.