UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA IN INFORMATICA

TITOLO TESI

Relatore:
Prof. Anna Maria Zanaboni

Correlatore:
Prof. Dario Malchiodi

Candidato:
Alessia Lombarda

Matr. 908879

ANNO ACCADEMICO 2019/2020

Indice

1 Introduzione

2 Apprendimento supervisionato e induzione di insiemi fuzzy

2.1 Apprendimento supervisionato

L'apprendimento automatico è una branca dell'intelligenza artificiale che studia algoritmi che utilizzano l'esperienza per migliorare la propria performance o per fare predizioni più accurate, dove con esperienza intendiamo l'informazione passata disponibile al sistema che apprende, tipicamente una collezione di dati resi disponibili all'analisi. Questi dati possono essere insiemi di informazioni a cui sono state abbinate manualmente delle etichette oppure dati estratti direttamente dall'ambiente: gli algoritmi avranno l'obiettivo di effettuare predizioni sulla base dei dati appresi. L'apprendimento automatico si applica a svariati campi; alcune delle classi principali di problemi di apprendimento sono:

- **Classificazione:** l'assegnamento di ogni elemento ad una categoria; in questo caso di solito il numero di classi possibili è ridotto
- **Regressione:** la predizione di un valore reale per ogni elemento nel set; in questo caso l'errore associato ad una predizione errata dipende della differenza tra il valore predetto e quello reale
- **Ranking:** l'ordinamento di elementi in base a un criterio definito a priori
- **Clustering:** il partizionamento di elementi in regioni omogenee
- **Riduzione della dimensionalità:** la trasformazione di un set di elementi in una rappresentazione degli stessi a meno dimensioni, preservando alcune proprietà degli oggetti iniziali

L'apprendimento automatico può essere *supervisionato*, *non supervisionato* o *semi-supervisionato*; si tratterà la prima categoria di algoritmi.
L'apprendimento supervisionato è uno dei tipi di apprendimento automatico che definisce gli algoritmi in cui il learner riceve un set di dati con etichette associate (*training set*) per poi eseguire delle predizioni su dati non noti.
L'algoritmo in questione dovrà quindi apprendere una funzione, f, che cercherà di approssimare al meglio con una funzione h, dove sia f che h sono funzioni di un vettore di input $X = (x_1, x_2, ..., x_i, ..., x_n)$ di n componenti. Nel caso di apprendimento supervisionato si conoscono quindi (a volte solo in modo approssimato) i valori di f per gli m elementi del training set, Ξ. Assumiamo che, se possiamo trovare una funzione h che restituisce valori molto vicini a quelli di f per gli elementi di Ξ, allora questa funzione costituisce una buona predizione per f, specialmente se Ξ è grande.

Per quanto riguarda il vettore di input X, questo può essere costituito da valori reali, discreti o categorici, che a loro volta possono essere ordinati (ad

esempio {*small, medium, large*}) o non ordinati; un caso a sè è l'uso di valori booleani, che può essere considerato un caso particolare dell'utilizzo di numeri discreti (1,0) o di variabili categoriche (*True*, *False*).
L'output può essere invece un numero reale, ovvero una stima, oppure un valore categorico, tipico dei classificatori. Nel caso di output booleani, invece, possiamo distinguere *istanze positive*, ovvero quelle con label 1, e *istanze negative*, quelle con label 0; se anche l'input è booleano il classificatore implementa una *funzione booleana*.

Definiamo ora il *bias*, ovvero un insieme di informazioni che devono essere note a priori affinché l'algoritmo approssimi correttamente la funzione f: perché ciò avvenga, bisogna infatti limitare l'insieme di possibili funzioni, tra cui poi l'algoritmo sceglierà la migliore. Se ad esempio si considera un algoritmo che deve approssimare una funzione booleana in n variabili, si potranno avere 2^n possibili input. Si supponga di non avere bias, che sia $\mathcal{H}$ l'insieme delle 2^n

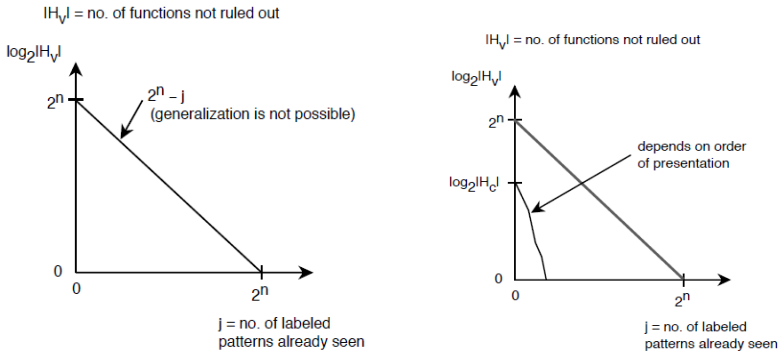

Figura 1: Relazione tra il numero di funzioni possibili e gli elementi del training set già esplorati in assenza e presenza di bias

possibili funzioni booleane e che non si abbiano preferenze tra quelle che danno corrispondenze sui dati del training set. In questo caso, dopo aver considerato un elemento del training set e la sua label si potrà suddividere a metà l'insieme delle funzioni, ovvero distinguere quelle che classificherebbero correttamente quell'elemento dalle restanti. Presentando via via più elementi del training set, ad ogni passo del processo si dimezza il numero di funzioni considerabili come ipotesi di approssimazione per f, come mostrato in Figura 1a. Non è possibile in questo caso generalizzare, perché i pattern già trovati non danno alcun indizio su quelli ancora da scoprire: è solo possibile memorizzare i primi, e ciò causa un apprendimento molto oneroso.
Se invece si limitasse l'insieme $\mathcal{H}$ a un sottoinsieme $\mathcal{H}_c$, dipendentemente dal sottoinsieme e dall'ordine di presentazione dei dati del training set la curva che rappresenta il numero di possibili ipotesi di funzioni apparirebbe come in Fi-

gura 1b: potrebbe infatti accadere che dopo aver visto meno di 2^n campioni si arrivi già a selezionare una funzione h che ben approssimi f. L'introduzione del bias, quindi, ci permette di considerare solo alcune classi di funzioni, rendendo l'apprendimento più efficiente.

L'algoritmo di apprendimento viene quindi addestrato sul *training set* e successivamente valutato su un insieme distinto di dati, il *test set*. Si dice che una funzione *generalizza* se effettua predizioni perlopiù corrette sul test set.
In questo contesto è anche necessario selezionare le features rilevanti da considerare per ogni campione, in quanto features significative possono guidare l'algoritmo di apprendimento correttamente, mentre altre povere di significato possono portare a significativi errori: prima di procedere con l'apprendimento è quindi necessario utilizzare un algoritmo di feature selection.
Altra questione da considerare e gestire è la possibile presenza di rumore nei vettori del training set: possiamo distinguere il *rumore di classe*, che altera il valore della funzione f, e quello di attributo, che altera in modo causale i valori del vettore di input X, rendendo imprecisa la corrispondenza tra i valori di input e quelli della funzione applicata su di essi.

Fondamentale è poi la definizione di un metodo per la valutazione della performance dell'algoritmo: consideriamo a questo scopo l'*accuratezza* e le *funzioni di errore*. L'accuratezza consiste nel conteggiare il numero di predizioni corrette, e dividere questo valore per il la cardinalità del set di dati: si otterrà un valore compreso tra 0 e 1, che sarà maggiore se l'algoritmo predice in modo corretto. Uno stimatore più robusto è invece l'*errore quadratico medio*, che si vuole minimizzare e rappresenta il rapporto tra la varianza entro i gruppi e la numerosità totale:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2$$

dove y_i è la label attesa e $h(x_i)$ quella predetta dall'algoritmo, e la sua radice quadrata, l'RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2}$$

Per migliorare la performance dell'algoritmo ed evitare che questo si specializzi sui dati (*overfitting*), è necessario svolgere una fase di training accurata; a causa della ridotta dimensione del training set disponibile, solitamente si è soliti usare una tecnica nota come *n-fold cross validation*, usata sia per la model selection (la selezione dei parametri liberi dell'algoritmo), sia per la fase di training.

Approfondiamo inizialmente l'uso della n-fold cross validation per la fase di model selection: detto θ il vettore dei parametri liberi dell'algoritmo, per

un fissato valore di θ si partiziona in modo causale un set S di m elementi con label associate in n sottogruppi, o fold. L'i-esimo fold sarà quindi un campione $((x_{i1}, y_{i1}), ..., (x_{im}, y_{im}))$ di cardinalità m_i. Poi, per ogni $i \in [1, n]$ l'algoritmo viene addestrato su un *validation set* costituito da tutti i fold tranne l'i-esimo, per generare un'ipotetica h_i, e la performance di h_i viene testata sull'i-esimo fold. Il valore del parametro θ è scelto sulla base dell'errore medio di h_i, chiamato *cross-validation error*, denotato da $\widehat{R}_{CV}(\theta)$ e definito da

$$\widehat{R}_{CV}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=m_i}^{n} L(h_i(x_{ij}), y_{ij})$$

I fold sono generalmente definiti della stessa taglia, ovvero $m_i = m/n$ per ogni $i \in [1, n]$. La scelta di n è molto importante ai fini dell'utilizzo di questo metodo: con un n grande ogni fold avrà taglia prossima ad m (la linea rossa sulla destra in 3), la cardinalità dell'intero dataset, ma i fold saranno piuttosto simili, e quindi il metodo avrà un bias ridotto ed una grande varianza. Viceversa, valori bassi di n portano a training set differenziati, ma la loro taglia è significativamente più bassa di m (la linea rossa sulla sinistra in 3) e quindi il metodo tenderà ad avere una varianza minore ma un bias maggiore. Solitamente si scelgono come valori per n 5 o 10.

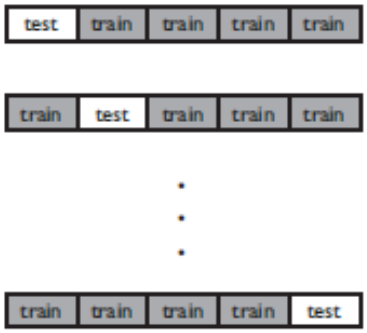

Figura 2: n-fold cross validation

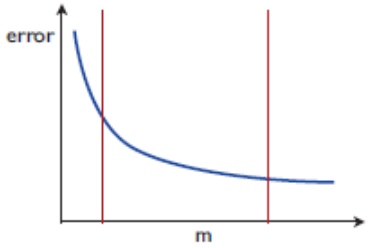

Figura 3: Grafico dell'errore di predizione di un classificatore in funzione della dimensione del training set

Per la model selection, quindi, la cross validation è utilizzata in questo modo: l'intero dataset viene suddiviso inizialmente in training e test set, dove il training set ha taglia m. Questo viene poi utilizzato per computare il cross validation error $\widehat{R}_{CV}(\theta)$ per un certo numero di possibili valori di θ; viene considerato come θ_0 il valore per cui $\widehat{R}_{CV}(\theta)$ è minore, e l'algoritmo viene addestrato con i parametri di θ_0 sull'intero training set di taglia m. La sua performance è poi valutata sul test set. Il metodo si può applicare allo stesso modo per quanto riguarda la fase di training: si otterranno delle valutazioni parziali (accuratezze

o errori), di cui possiamo calcolare la media per avere un'indicazione della bontà dell'apprendimento.

2.2 Insiemi fuzzy e induzione della funzione di membership

La teoria classica degli insiemi, inizialmente formulata da Zadeh [1], è basata sul concetto fondamentale di *insieme* di cui un elemento fa parte oppure no, una distinzione netta che permette di identificare un chiaro confine tra ciò che appartiene e non appartiene all'insieme. Molti problemi reali non possono però essere descritti e gestiti dalla teoria classica degli insiemi, in quanto trattano elementi che hanno appartenenza solo parziale ad un dato insieme. La teoria dei *fuzzy set* (o insiemi sfocati), generalizza ed estende in questo senso la teoria classica degli insiemi.
Nella teoria classica, possiamo definire la funzione caratteristica di un insieme, ovvero la funzione di appartenenza ad un insieme A come

$$X_A(x) = \begin{cases} 1, & if \text{ x} \in \text{A} \\ 0, & if \text{ x} \notin \text{A} \end{cases}$$

Ma se un elemento ha membership soltanto parziale ad un insieme dobbiamo generalizzare questa funzione per determinare il grado di membership dell'elemento stesso: un valore maggiore indicherà un grado di membership più alto all'insieme. Se consideriamo ad esempio l'insieme universo S di tutti gli esseri umani, e S_f come

$$S_f = \{\, s \in S \mid s \text{ is old} \,\},$$

S_f è un *sottoinsieme fuzzy* di S, perché la proprietà "vecchio" non è ben definita e non può essere misurata in modo preciso: dobbiamo quindi stabilire una funzione che assegni "il valore di vecchiaia" di un essere umano, ma non esiste un criterio unico o universale per scegliere questa funzione.
Un sottoinsieme fuzzy è quindi definito da un sottoinsieme e da una funzione di membership μ ad esso associata, che associa ad ogni elemento dell'insieme un valore reale nell'intervallo $[0, 1]$ (si vedano alcuni esempi in Figura 4).

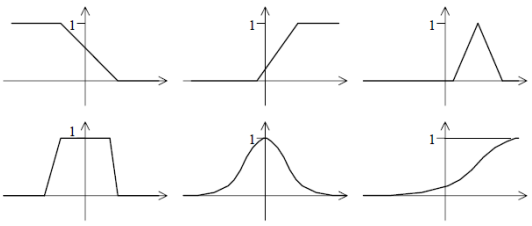

Figura 4: Forme di funzioni di membership comunemente usate

Parallelamente alla definizione di insiemi fuzzy si può effettuare una distinzione tra logica classica e logica fuzzy. La logica classica, infatti, riguarda proposizioni che possono essere vere o false ma che non possono mai assumere contemporaneamente entrambi i valori di verità, che hanno opposto e possono essere combinate tra loro. La logica fuzzy, che pone la base del ragionamento approssimato, si differenzia da quella tradizionale in quanto ammette l'utilizzo termini linguistici imprecisi come

- Predicati fuzzy: vecchio, giovane, alto, veloce
- Quantificatori fuzzy: molto, poco, quasi, di solito
- Valori di verità fuzzy: molto vero, probabilmente falso, sicuramente falso

che permettono di trattare fenomeni della realtà difficilmente descrivibili in modo quantitativo, ma su cui la maggior parte del ragionamento umano si basa. In particolare si dicono *variabili linguistiche* gli attributi dei sistemi fuzzy che assumono *valori linguistici* espressi in linguaggio naturale. Questi valori hanno un significato e non un preciso valore numerico, e partizionano i possibili valori delle variabili linguistiche in modo soggettivo (ovvero solitamente basato sull'intuizione umana). Se ad esempio consideriamo come variabile linguistica la superficie abitabile di un appartamento A, e definiamo i valori linguistici $\{tiny, small, medium, large, huge\}$ e le funzione di membership come in Figura 5:

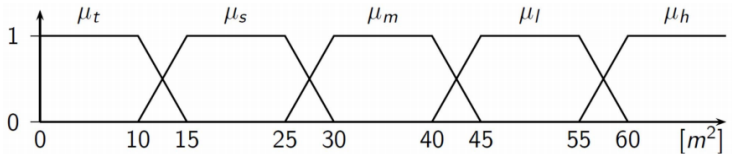

Figura 5: Esempio: funzioni di membership in riferimento alla superficie abitabile di un appartamento

Possiamo notare che ogni $x \in A$ ha $\mu(x) \in [0, 1]$ per ogni valore. Se ad esempio consideriamo $x = 42.5m^2$, avremo che $\mu_t(x) = \mu_s(x)\mu_h(x) = 0$, mentre $\mu_m(x) = \mu_l(x) = 0.5$.

L'applicazione della logica fuzzy a problemi di apprendimento passa attraverso l'utilizzo dei FIS (Fuzzy Inference Systems), che composti da cinque blocchi funzionali:

- Rule Base: contiene le regole if-then fuzzy definite da esperti
- Database: definisce le membership function dei fuzzy set usati nelle regole fuzzy
- Inference Engine: esegue operazioni sulle regole

- Fuzzifier: converte le quantità crisp in quantità fuzzy
- Defuzzifier: converte le quantità fuzzy in quantità crisp

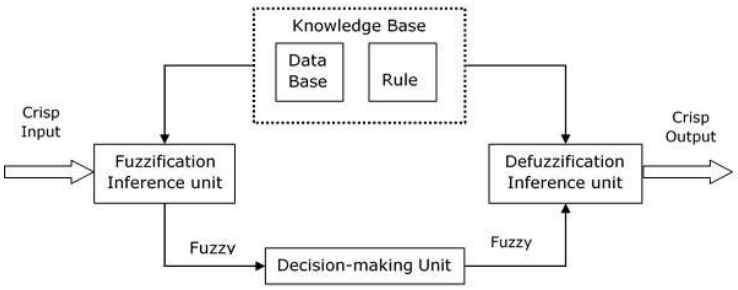

Figura 6: Struttura di un FIS

Il FIS quindi, e in particolare il fuzzifier converte l'input da crisp a fuzzy, aggiornando la Knowledge Base formata da Database e Rule base; l'input viene poi passato all'Inference Engine, che determina il match con le regole if-then (e aggiorna le regole stesse sulla base dell'input), ovvero associa input fuzzy ad output fuzzy, che viene poi ritrasformato in output crisp dal defuzzifier.
I concetti della teoria dei fuzzy set applicati all'apprendimento automatico hanno portato alla nascita del *fuzzy machine learning*, che ha permesso l'estensione di tecniche di apprendimento non fuzzy a corrispondenti tecniche fuzzy: si può parlare di fuzzy clustering, fuzzy support vector machines, fuzzy k-nearest neighbour e così via. Queste tecniche sono utilizzate principalmente in tre tipi di problemi: quelli di classificazione e data analysis, i problemi di decision-making e il ragionamento approssimato; questi problemi mostrano le tre semantiche attribuibili al grado di membership, ovvero la similarità, la preferenza e la possibilità. Per quanto riguarda la prima categoria, che approfondiremo in seguito, il grado di membership di un elemento u $\mu(u)$ indica la prossimità al valore che si ha come prototipo; questa semantica è specialmente utilizzata per problemi di pattern classification, clustering e regressione.

Nel campo dell'apprendimento automatico, quindi, le tecniche di induzione di funzioni di appartenenza producono modelli che trasformano variabili di input in gradi di appartenenza ad un insieme fuzzy. Si possono distinguere a questo proposito tecniche induttive o deduttive: l'uso di metodi deduttivi implica la generazione di funzioni di appartenenza da parte di esperti, ovvero sulla percezione umana. Le tecniche induttive, invece, si basano principalmente sui dati. Si approfondiranno in seguito queste ultime, più affidabili in quanto slegate dalle diverse interpretazioni che esperti possono dare agli stessi attributi o variabili.

Riferimenti bibliografici

[1] L. Zadeh. Fuzzy sets. *Information and Control*.