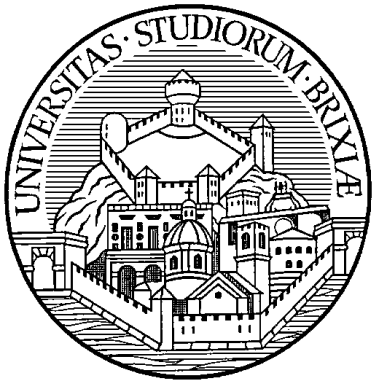
Università degli Studi di Brescia - Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica per l'Automazione



UN NUOVO ALGORITMO PER LA COSTRUZIONE DI CLASSIFICATORI DA DATI SPERIMENTALI CON ERRORE DI GENERALIZZAZIONE GARANTITO

Relatore
PROF. MARCO C. CAMPI

Tesi di laurea specialistica
di
ALGO CARÈ
matr. 60772

Anno Accademico 2007-2008

Indice

Introduzione

In questo lavoro si presenta RealGPE, un nuovo algoritmo per la generazione, a partire da un insieme di dati, di classificatori binari con opzione di rigetto. L'algoritmo ha delle proprietà teoriche che consentono di avere garanzie stringenti, in probabilità, sull'errore di generalizzazione dei classificatori costruiti, indipendentemente dalla realtà che genera i dati.

Il lavoro è diviso in due parti. La **prima parte** si apre con il capitolo 1 che fornisce un'introduzione ai concetti chiave dell'apprendimento automatico e della classificazione, come il concetto di *errore di generalizzazione*. Nel capitolo 2 è descritto il funzionamento di RealGPE e sono evidenziate le sue importanti proprietà teoriche. Nel capitolo 3 viene presentato IdealGPE, un algoritmo ideale, ma simulabile, per il quale la distribuzione dell'errore di generalizzazione è esattamente nota a priori: esso permette di meglio comprendere il funzionamento di RealGPE e di dimostrarne le proprietà (nei capitoli 6 e 7 della seconda parte). Il capitolo 4 conclude l'esposizione dei risultati attraverso verifiche sperimentali sia su semplici dati artificiali, per fornire una prima conferma circa la correttezza della teoria sviluppata e le relazioni tra IdealGPE e RealGPE, sia su dati tratti da casi reali, nei quali RealGPE si dimostra competitivo, a confronto con altri algoritmi, e anche robusto quando le ipotesi teoriche non sono del tutto soddisfatte. La **seconda parte** è di approfondimento teorico ai risultati esposti nella prima. Il capitolo 5 giustifica la scelta di classificare con opzione di rigetto, alla luce di un risultato fortemente negativo circa la possibilità di garantire a priori l'errore di generalizzazione per classificatori binari. Il capitolo 6 presenta in astratto la base matematica necessaria per garantire le proprietà di RealGPE, mentre il capitolo 7 conduce realmente alla dimostrazione delle proprietà di IdealGPE e quindi di RealGPE. Il capitolo 8 racchiude alcune note circa la letteratura disponibile sulla problematica della garanzia dell'errore e sull'impiego dell'opzione di rigetto. Il capitolo 9 chiude il lavoro, presentando alcuni cenni a possibili ricerche future. Nell'appendice sono riportate alcune dimostrazioni omesse, per facilità di lettura, nel corso della trattazione.

Parte I

Algoritmi e proprietà

Capitolo 1

Che cos'è la classificazione

In questo capitolo si definisce l'ambito entro cui si colloca questa tesi, introducendo in maniera prima discorsiva e poi più formale i concetti chiave relativi all'apprendimento induttivo automatico, tra cui quello di *classificazione*. Nella sezione 1.4 si definisce l'importante concetto di *errore di generalizzazione*. Conclude il capitolo un catalogo con la menzione di alcune tra le più popolari tecniche per la classificazione.

1.1 Il problema dell'apprendimento

Con il termine "apprendimento" si fa riferimento ad una classe di concetti assai vasta ed inquadrabile secondo una tale varietà di prospettive che sarebbe assai pretenzioso il tentativo di comprenderle tutte in una breve introduzione ad un lavoro che, d'altra parte, si colloca in un contesto ben preciso. In una prospettiva ingegneristica, le questioni legate all'apprendimento riguardano la possibilità che *le macchine possano apprendere*: è detto *machine learning* (tradotto normalmente con *apprendimento automatico*) il ramo dell'intelligenza artificiale che si occupa di ideare algoritmi che consentano ad agenti non umani di manipolare dati in modo da ricavare *modelli* della realtà, al fine di rilevare e sfruttare regolarità per effettuare studi, previsioni o intraprendere decisioni. Prima di proseguire nel circoscrivere l'ambito in cui si collocheranno i risultati qui esposti, si vuole ricordare il monito dell'eclettico studioso Gregory Bateson che, da par suo, ammoniva:

> Non si deve chiedere: "Le macchine possono apprendere?" ma piuttosto: "Quale livello o ordine di apprendimento può raggiungere una data macchina?"[1]

[1] si veda [4].

Notava infatti come anche dispositivi meccanici semplicissimi siano in grado di presentare un livello base di apprendimento, fondato su relazioni causali tra stimolo e risposta, che definiva "apprendimento zero". Questo tipo di apprendimento è quello a cui si fa peraltro riferimento nel linguaggio comune, quando ad esempio si dice:

> dalla sirena della fabbrica "apprendo" che è mezzogiorno.[2]

Ma bando alle pretese onnicomprensive, in una materia tanto vasta e perfino terminologicamente intricata.

1.2 Apprendimento induttivo

1.2.1 Concetti di base

Questo lavoro si colloca nel campo dell'*apprendimento induttivo automatico*, il cui fine è lo studio di algoritmi per l'individuazione di *ipotesi* attraverso le quali si vuole approssimare la *realtà* sulla base della conoscenza di alcuni *esempi*. Questa definizione è molto generica e si adatta bene a situazioni disparate, alcune delle quali esulano dalla presente trattazione: occorre perciò delimitare ulteriormente il campo di interesse.

Supponiamo di volerci avventurare in un bosco alla ricerca di funghi per sperimentare alcune ricette. Com'è noto, la cosa migliore è portare al nostro fianco un esperto, del quale seguire le mosse. Sul nostro cammino incontreremo diversi funghi, nessuno dei quali identico ad un altro[3]. Dall'esperto potremo però imparare, col tempo, a distinguere quelle caratteristiche che ci permetteranno di riconoscere i funghi eduli da quelli da evitare. Questo significa che, da un numero limitato di *esempi*, saremo in grado di costruire un modello mentale, cioè un'*ipotesi* sulla base della quale decideremo come comportarci di fronte ad un qualsiasi fungo di quel bosco.

Come già chiarito, non siamo certo intenzionati a studiare l'apprendimento umano, tuttavia, l'esempio dei funghi fornisce un'idea di che cosa si intenda per *apprendimento induttivo supervisionato*: la realtà propone ad un agente intenzionato ad apprendere un insieme di *oggetti* (i funghi), ciascuno dei quali ha una certa *natura* (di essere commestibile o non esserlo); un supervisore (l'esperto di funghi) conosce la natura degli oggetti e assegna loro

[2] ibid.

[3] Assumendo quanto scriveva G.W.Leibniz ne "La Monadologia", e cioè che non ci siano mai in natura due esseri perfettamente identici.

un'*etichetta* ("commestibile o non commestibile"), cioè fornisce all'agente degli *esempi* di corretta etichettatura. Al termine dell'*addestramento* (una serie di passeggiate nel bosco durante le quali si prende visione di un numero sufficiente di funghi ai quali il supervisore associa la relativa etichetta), l'agente sarà in grado di esprimere giudizi sulla base di un modello che chiamiamo *ipotesi*. Un'ipotesi definisce quindi una relazione tra oggetti o, meglio, alcune qualità degli oggetti (dimensioni, colore, caratteristiche dell'anello, delle lamelle,...) ed etichette, che, nelle intenzioni, dovrebbe rispecchiare il più possibile la relazione esistente nella realtà.

Decisioni basate su ipotesi tratte dall'acquisizione di un numero finito di esempi si presentano in continuazione e in campi molto diversi, in molti dei quali i calcolatori possono rivestire un ruolo fondamentale di ausilio per la loro capacità di trattare una grande quantità di dati in tempi rapidi. Una banca potrebbe voler prevedere, sulla base delle caratteristiche di un cliente, se concedergli o meno un credito, basandosi su dati di esperienze passate, di cui sono noti gli esiti; i medici, e i pazienti, sono interessati al riconoscimento di malattie o alla loro evoluzione, e via esemplificando. Un *software* di apprendimento automatico può essere inoltre parte di sistemi automatici più ampi, collegato a strumenti per il riconoscimento di caratteri scritti, per l'individuazione di particolari segnali vocali, di pacchetti in transito nelle reti di comunicazione, di *e-mail* pubblicitarie indesiderate, etc.

In tutti i casi, è necessario formalizzare il problema al fine di renderlo trattabile da un calcolatore.

1.2.2 Formalizzazione del problema

Gli oggetti e la loro natura Lo spazio degli oggetti viene modellizzato attraverso un insieme X.

L'associazione ad un oggetto della relativa natura è matematicamente descritta dalla funzione

$$y(\cdot) : X \longrightarrow Y,$$

dove Y è l'insieme delle nature, o etichette[4].

L'insieme di addestramento Un *esempio* è una coppia oggetto-natura: $(x_i, y(x_i))$. Un *insieme di addestramento* è costituito da più esempi e forma-

[4]Matematicamente, l'insieme delle nature e quello delle etichette coincidono, nonostante che il concetto di *natura* sembri più adatto a riferirsi ad una proprietà effettiva dell'oggetto, mentre *etichetta* presuppone un intervento esterno di "etichettatura". La formulazione matematica adottata giustifica normalmente un uso interscambiabile.

lizzato attraverso una N-upla $((x_1, y_1), ..., (x_N, y_N))$ dove si è indicato con y_i il valore $y(x_i)$.

L'algoritmo di apprendimento Vorremmo, a partire dall'insieme di addestramento, ricostruire la funzione y. Se y assume valori nel continuo, quello proposto è un problema di *regressione*. Se la cardinalità di Y è invece finita, si parla di *classificazione*. Questa tesi si concentra sulla *classificazione binaria*, binaria in quanto $|Y| = 2$. Quando non altrimenti specificato, si assumerà allora che le nature possano essere rappresentate dai valori simbolici 0 oppure 1. Risolvere il problema significa trovare un **algoritmo di classificazione**, cioè una procedura che consente, a partire dall'insieme di addestramento, di ottenere una *funzione di decisione*

$$\widehat{y}_N(\cdot) : X \longrightarrow Y.$$

La funzione $\widehat{y}_N$ assume il ruolo di *ipotesi* circa la realtà e permette di etichettare, cioè *classificare*, qualsiasi elemento di X con etichette 0 o 1: è perciò detta *classificatore binario*. Vorremmo, naturalmente, che tale classificatore fosse in grado di riprodurre al meglio y.

Classificazione binaria con opzione di rigetto Poniamoci nei panni dell'appena addestrato ricercatore di funghi, che si aggira nel bosco finalmente da solo. Supponiamo che egli s'imbatta in un fungo sconosciuto e non riesca a dedurne la natura. Evidentemente, in tale circostanza sarebbe prudente che egli ammettesse la propria mancanza di informazioni a riguardo e sospendesse il giudizio in attesa di approfondimenti. Qualunque responso definitivo sarebbe infatti potenzialmente dannoso: per ovvie ragioni di salute in un caso, per ragioni culinarie nell'altro. Può quindi essere opportuno, anche in applicazioni reali, per esempio quelle mediche, riservarsi la possibilità di *non classificare*. Un classificatore che può non dare alcuna risposta circa l'etichetta di un oggetto che gli si chiede di classificare è detto *classificatore con opzione di rigetto* o, semplicemente, *con rigetto* e gli oggetti che si rifiuta di classificare si dicono *rigettati*. Per esprimere questa possibilità, si aggiunge all'insieme delle etichette assegnabili da y_N il *valore speciale* R. I campioni x tali che $\widehat{y}_N(x) = \mathsf{R}$ sono i campioni rigettati. Avremo allora

$$\widehat{y}_N(\cdot) : X \longrightarrow Y \cup \{\mathsf{R}\}.$$

Nella seconda parte, al capitolo 5, sarà esposto un risultato teorico che giustifica, non solo alla luce del buon senso, l'interesse verso i classificatori con rigetto.

L'osservazione di campioni Occorre a questo punto cercare di modellizzare un aspetto fondamentale: come può essere costruito un insieme di apprendimento? Come si perviene a conoscere un certo esempio? Se il maestro e l'allievo, attraversando il bosco, si ripromettessero di seguire un percorso molto regolare e di esaminare uno a uno, ordinatamente, qualsiasi fungo trovato sul proprio cammino, verosimilmente si imbatterebbero spesso in successioni di funghi tra loro molto simili e quindi della stessa natura: questo perché alcuni funghi crescono a gruppi. Normalmente, però, della realtà da cui vengono tratti gli esempi si sa pochissimo (anche, talvolta, per via di una sua intrinseca, eccessiva complessità) e l'assunzione comunemente adottata è che l'osservazione di un oggetto avvenga del tutto indipendentemente dalla natura di altri oggetti visti. Ciò rende bene conto del fatto che, per esempio, le caratteristiche di un paziente che entra in un dato momento in uno studio medico, generalmente, non hanno relazione con quelle del paziente appena uscito. Nel caso dei funghi, l'indipendenza sarebbe invece plausibile se si passeggiasse casualmente per il bosco, aprendo gli occhi solo di tanto in tanto, per riprendere la passeggiata casuale non appena esaminato un fungo.

Con terminologia statistica, l'atto di osservare un certo oggetto (l'imbattersi in un fungo, l'ingresso in studio di un paziente,...) può essere visto come un *campionamento* che ha per esito l'estrazione di un elemento di X (che assume pertanto il ruolo di *spazio campionario*). Assumiamo allora che la probabilità di imbattersi in un certo oggetto in X sia descritta attraverso una misura di probabilità μ. Così, un oggetto osservato sarà considerato come un *campione estratto* da X, in accordo con la probabilità μ. Al concetto di campionamento viene ricondotta sia l'osservazione di un oggetto in sé, sia l'osservazione di un oggetto accompagnato dalla sua natura, così come avviene in fase di addestramento: un insieme di addestramento di N esempi verrà allora a costituirsi in seguito a N estrazioni indipendenti e pertanto verrà talvolta chiamato *multi-estrazione*. La probabilità di ricavare dalla realtà un certo insieme di addestramento di lunghezza N sarà, quindi, descritta dalla misura prodotto μ^N su X^N. Questa, di avere *campioni indipendenti ed identicamente distribuiti*, è l'unica ipotesi circa la realtà che imporremo: ciò significa escludere ogni conoscenza, anche circa μ, della quale si assume soltanto l'esistenza. Più avanti, affinché per l'algoritmo che sarà proposto siano dimostrabili alcune importanti proprietà, si richiederà anche che μ su X ammetta una densità, ma niente di più. Ogni altra conoscenza a priori del mondo è esclusa dal presente contesto di *induzione pura*.

1.3 Etichette non deterministiche

> Eadem sunt, quorum unum potest substitui alteri salva veritate
>
> G.W.Leibniz

Esaminiamo criticamente l'assunzione fondamentale, circa l'esistenza di una funzione che, ad un certo oggetto (in X), associa un'etichetta (in Y)

$$y : X \longrightarrow Y$$

Nella pratica, a ciascun oggetto *reale* viene associato un oggetto matematico in X, in base a certe caratteristiche opportunamente codificate: X sarà un insieme di vettori numerici che codificano gli *attributi* degli oggetti reali che sono considerati rilevanti per il fine che ci si prefigge. Quando, per esempio, vogliamo decidere se una persona è uomo o donna, ci basiamo su dati fisici, quali le armoniche della voce e la sporgenza del pomo di Adamo: a poco serve conoscere se è nata in un giorno feriale o festivo.

Pur ammettendo che ogni oggetto reale abbia una propria natura ben definita associabile agli elementi di Y, può accadere che la funzione y definita su X non sia in grado di rendere conto di tale relazione. Infatti, nulla garantisce a priori che oggetti che noi identifichiamo per la loro comunanza di caratteristiche rilevanti siano *nella realtà* di natura diversa. Tornando all'esempio, per discriminare tra uomini e donne è possibile tentare di basarsi solo sul timbro della voce e procedere nel seguente modo: date delle persone, si campiona la loro voce e si estraggono dalle sequenze campionate alcune caratteristiche. Può darsi, però, che nel mondo esistano un uomo e una donna per cui tali caratteristiche coincidono. Ciò significa ammettere che y possa non essere una funzione. Quando ciò capita, una soluzione è descrivere il problema in termini probabilistici, assumendo che una certa estrazione renda nota una coppia appartenente allo spazio $X \times Y$, senza introdurre dipendenze funzionali tra X e Y. Oltre a μ, misura di probabilità su X, si può allora introdurre una funzione η, definita per ogni x, come la probabilità che l'etichetta che accompagna x valga 1 (per tutto il corso della trattazione, con $\mathbb{P}$ intenderemo in generale un valore dipendente da una misura di probabilità definita opportunamente a seconda dell'argomento):

$$\eta(x) = \mathbb{P}(Y = 1|X = x) = \mathbf{E}\left[Y|X = x\right]$$

Note μ ed η, ci si può chiedere quale sia il classificatore *binario* migliore, che chiamiamo $v(\cdot)$ e che dunque si vorrebbe ricostruire. Un buon classificatore minimizzerà la probabilità che ad un certo campione sia assegnata

un'etichetta diversa dalla sua vera natura e cioè deve valere $\mathbb{P}(\upsilon(X) \neq Y) \leq \mathbb{P}(h(X) \neq Y)$ per qualsiasi classificatore binario h.

υ è detta *funzione di decisione bayesiana* ed è così definita

$$\upsilon(x) = \begin{cases} 1 & \text{se } \eta(x) > 0.5 \\ 0 & \text{se altrimenti} \end{cases}$$

Per approfondimenti in questa direzione, si veda per es. [36].

Secondo un approccio diverso da quello probabilistico, in questa sede si assumerà implicitamente che gli oggetti reali, ciascuno con la propria determinata natura, possano, almeno idealmente, essere biunivocamente associati agli oggetti matematici X. A questo punto il problema diventa semplicemente che, definito X come l'insieme degli oggetti e assumendo un oggetto scomponibile come

$$x = (x_{\text{osservate}}, x_{\text{nascoste}})$$

sicché

$$y(x) = f(x_{\text{osservate}}, x_{\text{nascoste}}),$$

l'osservatore (l'algoritmo) possa o voglia prendere in considerazione solo le $x_{\text{osservate}}$. In tali circostanze, è di nuovo necessario ammettere un'indeterminazione nella corrispondenza tra oggetti ed etichette, ma questa volta *solo dal punto di vista dell'algoritmo* e non della funzione y. Si supporrà quindi che la natura per ogni oggetto $(x_{\text{osservate}}, x_{\text{nascoste}})$ sia data deterministicamente ma che *i valori di* $\widehat{y}_N$, funzione matematicamente definita sullo spazio $X = \{(x_{\text{osservate}}, x_{\text{nascoste}})\}$, *dipendano solo dagli attributi osservati*.

1.4 Valutare la classificazione

Data una multi-estrazione $\boldsymbol{d} = ((x_1, y_1)), ...(x_N, y_N))$, un algoritmo di classificazione ricaverà la funzione di decisione $\widehat{y}_N$. È interessante conoscere qual è la probabilità che venga estratto un campione $x \in X$ che sia misclassificato da $\widehat{y}_N$, cioè tale che $\widehat{y}_N(x) \neq \mathsf{R}$ e $y(x) \neq \widehat{y}_N(x)$,

Per una certa $\boldsymbol{d}$ e la relativa $\widehat{y}_N$ si dà allora la seguente

Definizione 1. *PE*$(\widehat{y}_N) = \mu(\{x \in X \text{ t.c. } \widehat{y}_N(x) \neq \mathsf{R} \text{ e } y(x) \neq \widehat{y}_N(x)\})$

PE$(\widehat{y}_N)$ è detto *errore di generalizzazione* del classificatore $\widehat{y}_N$. Più avanti, per brevità di scrittura, PE$(\widehat{y}_N)$ potrà essere scritto anche come V_N. L'insieme $\{x \in X \text{ t.c. } \widehat{y}_N(x) \neq \mathsf{R} \text{ e } y(x) \neq \widehat{y}_N(x)\}$ sarà spesso chiamato *insieme*

(o *regione*) *di misclassificazione*. Si noti che $\mathrm{PE}(\widehat{y}_N)$ è una funzione di $\boldsymbol{d}$, in quanto lo è $\widehat{y}_N$, che potrebbe essere scritta, per mettere meglio in evidenza questo fatto, come $\widehat{y}_{\boldsymbol{d}}$ o, se utile, $\widehat{y}_{N(x_1,...,x_N)}$[5]. È dunque anch'essa una variabile casuale di cui si può calcolare il valore atteso:

$$\mathbf{E}\left[\mathrm{PE}(\widehat{y}_N)\right] = \int_{X^N} \mathrm{PE}(\widehat{y}_N)_{(x_1,...,x_N)} \mu^N(\mathrm{d}x_1, ..., \mathrm{d}x_N) \tag{1.1}$$

Il valore atteso di $\mathrm{PE}(\widehat{y}_N)$ risponde alla seguente domanda:

> *Qual è la probabilità di osservare N esempi, produrre mediante l'algoritmo il relativo classificatore $\widehat{y}_N$ e, infine, osservare un nuovo campione (che non sia rigettato e) tale che $y_N(x) \neq y(x)$?*

In assenza di rigetto, il valore atteso dell'errore di generalizzazione è un modo sintetico per fornire informazioni sulla bontà di un algoritmo di classificazione, che, in termini di tassi di errore attesi, può dirsi completamente caratterizzato dalla distribuzione di $\mathrm{PE}(\widehat{y}_N)$. Si badi però che, generalmente, per conoscere esattamente queste grandezze, occorre conoscere μ e y.

In presenza di rigetto, tali grandezze offrono ancora un'importante informazione circa l'accuratezza dell'algoritmo, sebbene diventi centrale, per valutare l'efficacia dell'algoritmo, anche il tasso di rigetto: si mostrerà in questa tesi come sia possibile concepire algoritmi per i quali si possono avere garanzie a priori, cioè indipendenti da μ ed y, sulla distribuzione dell'errore di generalizzazione e che si dimostrino al contempo efficaci in problemi pratici.

1.5 Cenni ad alcune tecniche note per la classificazione

Prima di presentare il nuovo algoritmo vero oggeto della presente trattazione, si vogliono brevemente introdurre, senza nessuna pretesa di completezza, riferimenti ad alcune delle tecniche più note per la classificazione binaria.

1.5.1 Alberi di decisione

Un albero di decisione è costituito da tre tipi di nodi:

- il nodo radice, che corrisponde ad un attributo

[5]Ricordiamo che stiamo assumendo che, per ogni $x \in X$, resti determinata univocamente $y(x)$, questa è la ragione per cui si può non esplicitare la dipendenza dalle etichette dei punti considerati.

- nodi interni, corrispondenti ad attributi, con un ramo entrante e dei rami uscenti corrispondenti a possibili valori dell'attributo
- nodi foglie, corrispondenti alle possibili etichette di un campione

Per classificare un certo campione, l'albero viene percorso dall'alto in basso, seguendo il ramo che soddisfa la condizione sul valore dell'attributo associato al nodo di provenienza. Un albero può rappresentare qualsiasi funzione booleana: lo spazio delle ipotesi è quindi molto vasto. Per indurre un albero a partire dall'osservazione di alcuni dati è, allora, generalmente necessario appoggiarsi su considerazioni relative all'informazione portata dai diversi attributi; ad esempio, è evidente che l'attributo che consente una maggiore discriminazione dovrebbe trovarsi al nodo radice. D'altro canto, occorrerebbe evitare che l'albero formalizzi indebitamente delle regolarità soltanto apparenti (problema del sovradattamento, o *overfitting*). Alcuni risultati di interesse circa le euristiche per l'apprendimento di alberi di decisione sono reperibili in [52, 6].

1.5.2 Classificatori a regole

Un classificatore di questo tipo applica un insieme di regole $R = \{r_1, r_2, ..., r_m\}$, ciascuna delle quali è rappresentabile attraverso un *antecedente*, che esprime una condizione, e un conseguente che esprime un'etichetta da assegnare al campione che verifica la condizione:

$$r : (Condizione) \rightarrow y$$

Se un campione x verifica la condizione per una certa regola, si dice che tale regola *copre* x. Se più regole coprono uno stesso campione possono verificarsi conflitti, la cui risoluzione può essere demandata, per esempio, a politiche di voto o risolta mediante l'imposizione di priorità. D'altro canto, se un campione non è coperto da alcuna regola, gli si assegna generalmente un'etichetta predefinita. I metodi per indurre regole possono essere sia *ad hoc*, sia indiretti, ovvero basati sulla "traduzione" di alberi di decisione (per esempio, si veda il capitolo 5 di [53]) .

Quando le regole sono valutate in rigida sequenza, si ha una **lista di decisione** (abbreviata in DL; k-DL se ogni condizione è una congiunzione con al più k letterali).

1.5.3 Reti neurali

Da quando Warren McCulloch e Walter Pitts in [45] introdussero il primo modello logico-matematico di neurone, molto tempo è passato e la letteratu-

ra sulle reti neurali è diventata di dimensioni difficilmente gestibili, così come innumerevoli sono stati gli ambiti di applicazione. Un *rete neurale* è costituita da uno o più strati di neuroni artificiali connessi. Un neurone calcola la somma pesata dei propri ingressi e produce un'uscita in base all'applicazione a tale valore di una funzione non lineare a soglia dura, così da ammettere soltanto uscite binarie, o a soglia morbida, così da consentire uscite capaci di variare con continuità in un intervallo. L'uscita potrà terminare in un altro neurone per causarne, in concorso con altri, l'attivazione. Oltre ad uno strato di ingresso e ad uno di uscita, vi possono essere diversi strati *nascosti* di neuroni. A differenza delle reti neurali *ricorrenti*, le quali possono esibire un comportamento dinamico, le reti neurali alimentate in avanti (*feedforward*) non ammettono cicli tra le connessioni: matematicamente, rappresentano funzioni (generalmente) non lineari degli input, caratterizzate da molti parametri che vanno tarati, in fase di addestramento, minimizzando una funzione di errore. Pesi, connessioni e numero di strati influenzano l'espressività della funzione computata. Le difficoltà del metodo riguardano principalmente le decisioni architetturali e, contestualmente, il rischio di sovradattamento.

1.5.4 Reti bayesiane

Le *reti bayesiane* sono uno strumento che consente di definire in modo solitamente più compatto una funzione di densità congiunta di variabili casuali, sfruttando reciproche indipendenze e dipendenze, rappresentate per mezzo di un grafo diretto aciclico. L'approccio è divenuto celebre da quando Judea Pearl ideò metodi efficaci per effettuare inferenze in maniera efficiente basandosi su reti bayesiane generiche (ad es. si veda [49]). Tali strutture trovano impiego anche nella classificazione: se si interpreta ogni campione come un vettore di valori assunti da variabili casuali, dato un insieme di addestramento, si cerca la rete bayesiana che modella meglio le dipendenze condizionali delle variabili. Nella delicata fase di costruzione della rete intervengono normalmente ipotesi semplificative o conoscenze di dominio: uno degli approcci più seguiti, il cosiddetto *modello bayesiano ingenuo*, consiste addirittura nel supporre che gli attributi siano condizionalmente indipendenti l'uno dall'altro data la classe di appartenenza.

Una volta indotto il modello, alla presenza di un nuovo campione da classificare, si calcola la probabilità che la sua etichetta (corrispondente ad una variabile casuale nella rete) valga 1 o 0. La seconda fase riguarda l'effettiva *decisione* di assegnare un campione ad una certa classe. Il problema della decisione in un contesto bayesiano sarà ripreso all'inizio della sezione 8.2.

L'utilizzo di reti bayesiani è generalmente apprezzato per la sua robustezza a fronte di dati rumorosi (il cui peso viene attenuato per via del contesto

probabilistico) e per la capacità di resistere all'influenza di eventuali attributi irrilevanti.

1.5.5 Nearest Neighbour Classifier (NNC)

Un *classificatore ai vicini più prossimi* si basa sull'idea che le proprietà di un certo campione siano ragionevolmente le stesse di quelle di campioni *simili* ad esso. Matematicamente, due campioni "si somigliano più di altri" quando la loro distanza, calcolata secondo una certa metrica, è più piccola rispetto alle altre misurate. Definita opportunamente tale distanza (ad esempio come la distanza euclidea), si può pensare di memorizzare ciascun campione dell'insieme di addestramento con la propria etichetta. Quando si rende necessario classificare un nuovo campione, si può confrontarlo con i campioni memorizzati e decidere la sua natura sulla base della natura di quelli che più gli somigliano. Normalmente, per valutare la natura di un nuovo x, si trovano i k campioni che, secondo la metrica definita, gli sono *più prossimi*. k può essere scelto per tentativi, valutandone la bontà su insiemi di validazione. Il classificatore che si ottiene fissato un certo k è chiamato k-NNC.

La decisione potrà avvenire semplicemente in favore dell'etichetta posseduta dalla maggioranza dei k campioni, oppure in base a criteri leggermente più sofisticati che tengano conto anche dell'effettiva distanza, in modo che i più prossimi, tra i k che hanno voce in capitolo, contino più di quelli periferici.

1.5.6 Support Vector Machine (SVM)

L'idea di base per le SVM è quella di costruire, nello spazio dei campioni X (qui assumiamo con etichette in $Y = \{+1, -1\}$), un *iperpiano separatore ottimo* (OSH, o Optimal Separating Hyperplane) che divida i campioni di etichette diverse e che soddisfi la proprietà per cui $l_{+1} + l_{-1}$ è massima, dove con l_{+1} (l_{-1}) intendiamo la più breve distanza dal piano del più vicino campione etichettato con $+1$ (-1). Com'è facile immaginare pensando anche solo a casi in cui il campionamento effettuato sia rumoroso, non sempre ciò è possibile. È tuttavia possibile ammettere che alcuni campioni possano trovarsi dalla "parte sbagliata", introducendo però nella formalizzazione una penalità C (parametro di *soft margin*), in modo che l'eventualità, se possibile, sia scongiurata. Limitandosi a questa formulazione si otterrebbe un classificatore generalmente davvero poco espressivo. L'intuizione chiave è che, mappando attraverso una funzione Φ i campioni in uno spazio di dimensione "grande", questi possono, nel nuovo spazio, essere davvero linearmente separati da un piano. La forza di quest'intuizione sta nel fatto che la formulazione del problema di ottimizzazione finalizzato a ricavare l'OSH può essere riscritto in

modo da ottenere una soluzione nello spazio $\Phi(X)$ senza calcolare esplicitamente le trasformate attraverso Φ dei campioni $\boldsymbol{x}$ in gioco[6], ma limitandosi a sostituire i prodotti scalari tra campioni $\boldsymbol{x_i} \cdot \boldsymbol{x_j}$ con il valore $K(\boldsymbol{x_i} \cdot \boldsymbol{x_j})$, dove K è una funzione nucleo (*kernel*), tale che $K(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) = \Phi(\boldsymbol{x_i}) \cdot \Phi(\boldsymbol{x_j})$. Il fatto che trovare funzioni K con queste proprietà sia relativamente facile, e che esse per giunta siano normalmente calcolabili in modo efficiente, fa sì che ci si possa disinteressare completamente delle caratteristiche delle Φ corrispondenti e si possa operare agevolmente perfino in spazi a infinite dimensioni semplicemente calcolando alcuni valori di K. Ad esempio, lo spazio in cui "vive" il prodotto scalare corrispondente al valore della funzione nucleo gaussiana

$$K(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) = e^{-\gamma||\boldsymbol{x_i} - \boldsymbol{x_j}||^2} \tag{1.2}$$

è infinito-dimensionale.

Una funzione di decisione prodotta attraverso il ragionamento sopra esposto può pure essere calcolata senza utilizzare Φ e ha, per giunta, l'interessantissima proprietà di dipendere da un sottoinsieme dei campioni nell'insieme di addestramento, i cui elementi sono detti *vettori di supporto*: rimuovere dall'insieme di addestramento un campione che non è un vettore di supporto non modifica la soluzione. Alcuni riferimenti: [15, 7, 16].

1.5.7 Set Covering Machine (SCM)

Una Set Covering Machine è un classificatore introdotto in [43] (si veda anche [42], [35] e [34] per una formulazione basata su programmazione lineare) che classifica i campioni sulla base di una regola costruita come disgiunzione o congiunzione di *caratteristiche* binarie, che possono assumere in corrispondenza di un campione un valore positivo o negativo. Campioni nell'insieme di addestramento possono essere misclassificati nella convinzione che questo possa portare ad una maggiore generalizzazione: un certo controllo sulla possibilità di misclassificare nell'insieme di addestramento è fornito da due parametri: uno continuo, p, che rappresenta una penalità il cui incremento porta a ridurre il numero di errori su esempi positivi[7] dell'insieme di addestramento, ed s, che determina il numero massimo di caratteristiche utilizzate dalla funzione di decisione prodotta dall'algoritmo. Se s non è infinito, possono essere commessi errori sui campioni negativi. Le *caratteristiche* su cui si

[6] Usiamo il grassetto per indicare che si tratta generalmente di vettori in uno spazio potenza di $\mathbb{R}$.

[7] A rigore, l'aggettivo "positivi" è corretto se la funzione di decisione prodotta dall'algoritmo è una congiunzione, ma i termini "positivi" e "negativi" vanno invertiti, qui e nel resto della discussione, se invece la funzione prodotta è una disgiunzione. Per i dettagli, si rimanda comunque agli articoli citati.

basa la regola di decisione sono dipendenti dai dati: la prima formulazione, in [43], suggerisce palle centrate attorno ai campioni (solo negativi o anche positivi), definite sicché il valore al loro esterno sia la negazione di quello assunto all'interno.

Capitolo 2

RealGPE

Si presenta ora RealGPE, un algoritmo per la costruzione di classificatori binari con opzione di rigetto. La sua caratteristica principale è che fornisce garanzie a priori sull'errore di generalizzazione dei classificatori prodotti, indipendentemente dalla distribuzione dei campioni e dalla loro natura. Durante la costruzione del classificatore, l'algoritmo cerca, passo dopo passo, di coprire sempre di più lo spazio dei campioni, fino al verificarsi di una condizione legata all'errore di generalizzazione atteso dall'operatore.

Dapprima si riporta la descrizione di RealGPE, in una forma piuttosto discorsiva che permette una discreta libertà di commento, ma sufficientemente rigorosa da consentire l'agevole traduzione in un qualsiasi altro codice. Segue una breve analisi dell'algoritmo e delle sue proprietà, corredata da esempi grafici.

2.1 Definizione di RealGPE

RealGPE è un algoritmo che per ogni ingresso costituito da

- N esempi di addestramento[1]
- un parametro d, $1 \leq d < N$

produce un classificatore binario con rigetto, cioè una funzione di decisione definita su $\mathbb{R}^n$, a tre valori (0,1 e quello speciale R), che chiamiamo $\widehat{y}_N$. L'algoritmo è il seguente.

[1]Si assumerà sempre di non avere campioni uguali etichettati in modo contradditorio.

☞Si assume di avere in ingresso gli N esempi di addestramento estratti casualmente $((x_0, y_0), (x_1, y_1), ..., (x_{N-1}, y_{N-1}))$, dove $x_j \in \mathbb{R}^n$ e $y_j \in \{0, 1\}$ $(j = 0, ..., N-1)$, e il numero d

1. Si sceglie x_0 come primo punto di base ($x_b := x_0$). Si inizializza l'insieme $S := \emptyset$. L'insieme L contiene inizialmente tutti i punti nell'insieme di addestramento. L'algoritmo, al termine, avrà costruito una successione di m regioni, rappresentate da coppie (ξ_i, κ_i), $i = 1, ..., m$, ciascuna delle quali formata dal sottoinsieme $\xi_i \in \mathbb{R}^n$ e da un marcatore κ_i che può assumere i valori 1 o 0. m è il valore di una variabile che corrisponde ad un numero naturale ed è inizializzata a 1.

2. - SE $|S| + \frac{n(n+1)}{2} + n \leq d$, ALLORA si trova la soluzione A^*, B^* del problema convesso $P_{\frac{n(n+1)}{2}+n}(x_b, y(x_b))$, definito come segue:

   $P_{\frac{n(n+1)}{2}+n}(x_b, y(x_b))$:

   $$\begin{array}{ll} \min_{A=\in\mathbb{R}^{n\times n}, B\in\mathbb{R}^{1\times n}} & \text{Traccia}(A) \\ \text{con vincoli:} & (x_j - x_b)^T A(x_j - x_b) + B(x_j - x_b) \geq 1, \\ & \qquad \forall x_j \in \text{L}, y(x_j) \neq y(x_b) \\ & A = A^T \\ & A \succeq 0 \end{array}$$

   Nel caso di soluzioni multiple, si sceglie quella che minimizza la norma L2 di B ($||B|| = \sqrt{\sum_{i=1}^{n} b_i^2}$). Se ancora ci sono più soluzioni, allora si sceglie quella per cui l'elemento $a_{1,1}$ di A è minore; se ciò non basta ad individuarne soltanto una, si sceglie tra le candidate quella che minimizza $a_{1,2}$, e così via con le altre componenti della matrice, finché non si individua una sola soluzione.

   - SE $|S|+n+1 \leq d < |S|+\frac{n(n+1)}{2}+n$, ALLORA si trova la soluzione k^*, B^* del problema lineare $P_{n+1}(x_b, y(x_b))$, definito come segue:

     P_{n+1} $(x_b, y(x_b))$:

     $$\begin{array}{ll} \min_{k\geq 0, B\in\mathbb{R}^{1\times n}} & k \\ \text{con vincoli:} & k||x_j - x_b||^2 + B(x_j - x_b) \geq 1, \\ & \qquad \forall x_j \in \text{L}, y(x_j) \neq y(x_b) \end{array}$$

     e si impone $A^* = Ik^*$, $A \in \mathbb{R}^{n\times n}$ (I matrice identità). Nel caso di soluzioni multiple si sceglie quella che minimizza $||B||$.

   - SE $d < |S|+n+1$, ALLORA si trova la soluzione k^* del problema lineare

     $P_{\text{CMax}}(x_b, y(x_b))$, definito come segue:

$$
\begin{array}{l}
P_{\mathrm{CMax}}(x_b, y(x_b)): \\
\quad \begin{array}{ll}
\min_{k\geq 0} & k \\
\text{con vincoli:} & k||x_j - x_b||^2 \geq 1, \\
 & \qquad \forall x_j \in \mathrm{L}, y(x_j) \neq y(x_b)
\end{array}
\end{array}
$$

e si impongono $A^* = Ik^*$, $A \in \mathbb{R}^{n\times n}$ (I matrice identità) e $B^* = 0$

3. Si definisce la regione m-esima (ξ_m, κ_m),

   $\xi_m := \{x \in X t.c. (x - x_b)^T A^* (x - x_b) + B^*(x - x_b) < 1\}$
   $\kappa_m := y(x_b)$

4. Si tolgono da L i punti che appartengono a ξ_m.

5. Chiamiamo S_c l'insieme dei punti $x \in L$, con $y(x) \neq y(x_b)$, e che soddisfano con l'eguaglianza il vincolo

   $$(x - x_b)^T A(x - x_b) + B(x - x_b) \geq 1$$

   L'insieme S viene ridefinito come $S := S \cup S_c$ (cioè, l'insieme S dovrà da qui in poi contenere i punti in S_c).

6. SE $S_c = \emptyset$ o $|S| \geq d$ ALLORA **SI TERMINA** (CON AVVERTIMENTO se $|S| \neq d$)

7. Il punto di S_c a maggior distanza (in norma) da x_b viene scelto come nuovo punto di base e ribattezzato x_b. In caso tale punto non fosse unico, si prende quello la cui prima componente è minore; a parità di prima componente si guarda la seconda, etc.

8. $m := m + 1$ (la variabile m viene incrementata di un'unità) e si torna al punto 2.

☜

L'algoritmo descritto produce una successione di m regioni (ξ_i, κ_i), $i = 1, ..., m$. Ad esse si associa un classificatore, $\widehat{y}_N$ così definito:

$$
\widehat{y}_N(x) = \begin{cases}
\mathsf{R} \text{ (rigettato)} & \text{se } \nexists\, j,\ 1 \leq j \leq m, \text{ t.c. } x \in \xi_j \\
\kappa_i & \text{altrimenti, con} \\
 & \quad i = \min\{j,\ 1 \leq j \leq m,\ x \in \xi_j\}
\end{cases}
$$

2.2 Discussione

2.2.1 Logica di funzionamento

Il classificatore $\widehat{y}_N$ è definito basandosi sulle regioni costruite dall'algoritmo attorno a ciascun punto di base (i punti di base sono quelli che, di volta in volta, assumono il ruolo di x_b). Il primo punto di base per il quale si costruisce una regione è x_0. Costruita la regione, i punti al suo interno sono rimossi da L, che inizialmente contiene l'intero insieme di addestramento, e non rientreranno più in gioco[2], mentre i punti che stanno ai bordi della stessa vengono inseriti nell'insieme S e tra essi viene scelto il punto attorno al quale costruire una nuova regione. Le regioni attorno ad ogni x_b sono costruite in modo da escludere dal loro interno tutti i punti rimanenti in L che non siano della stessa natura di x_b. Esse saranno classificate, per come è costruita la funzione di decisione, in base alla natura di x_b. Nelle intersezioni tra regioni costruite su punti di base di nature opposte le prime generate hanno la priorità. Le regioni possono essere di diverso tipo. $P_{\frac{n(n+1)}{2}+n}$ dà luogo a regioni convesse che sono interne a sezioni di paraboloidi e relative degenerazioni cilindriche: il problema tende alla minimizzazione della traccia di A, col significato intuitivo di *allargare* quanto più possibile le regioni che si vengono a creare; quando la traccia di A si annulla, l'intera matrice A diventa zero (perché dev'essere semidefinita positiva) e quindi si viene a creare un iperpiano che divide lo spazio dei campioni in due parti di cui una classificata secondo la natura di x_b. P_{n+1} produce regioni ipersferiche di raggio massimo che racchiudono x_b, che può essere decentrato, oppure, quando A si annulla, di nuovo degenerano in una separazione dello spazio. P_{CMax} porta ancora alla costruzione di ipersfere, delle quali si massimizza il raggio, ma che devono avere come centro x_b. Tutti e tre i problemi, se risolti quando i campioni rimanenti sono tutti della stessa natura del punto di base, hanno per soluzione una regione che copre l'intero spazio dei campioni. L'algoritmo termina quando il numero dei punti in S raggiunge d oppure quando una regione si estende all'intero spazio dei campioni.

[2]Il lettore che ha familiarità con le tipiche tecniche di classificazione, troverà analogie tra quanto si è fin qui detto ed i classificatori basati su regole (per alcuni cenni, si veda la sezione 1.5.2). La tecnica di costruire regole rimuovendo via via dall'insieme di addestramento i punti coperti ad ogni passo è nota come *separate-and-conquer* (termine ispirato al celebre motto latino *divide et impera*) o *covering strategy* (strategia di copertura) ed è ricorrente in letteratura: [29] è uno studio esplorativo su alcuni algoritmi che ne fanno uso (nell'ambito del *concept learning*) e che vengono ricondotti ad un *framework* comune.

2.2.2 Proprietà teoriche

La caratteristica teorica principale di RealGPE è che, per un certo d, se gli N campioni con cui l'algoritmo viene addestrato sono estratti in maniera indipendente, secondo una probabilità (ignota) μ definita su $X = \mathbb{R}^n$ con una densità, allora vale la seguente disequazione:

$$\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon) \leq \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1} \tag{2.1}$$

Ciò significa che si hanno garanzie sulla probabilità dell'errore di generalizzazione $\mathrm{PE}(\widehat{y}_N)$ (la cui definizione si è data alla sezione 1.4) e giustifica la seconda parte del nome dell'algoritmo, "GPE", acronimo di "Guaranteed Probability of Error" (Probabilità di Errore Garantita). Si vuole evidenziare che nel termine a destra non appaiono termini dipendenti da μ o da y e che pertanto il risultato esposto vale indipendentemente dalle caratteristiche della realtà che genera gli esempi.

Nota la sola dimensione dell'insieme di addestramento, un operatore può allora stabilire (con una certa probabilità) il massimo errore di generalizzazione che è disposto a tollerare, fissando d e contando sul fatto che l'algoritmo cercherà di costruire un classificatore mediando tra la necessità di mantenere sotto controllo l'errore e la propensione ad azzerare il tasso di rigetto. Più in dettaglio, RealGPE si spingerà, incrementalmente, a coprire lo spazio dei campioni secondo un criterio euristico che massimizza l'ampiezza delle regioni via via create, finché non sarà soddisfatta una condizione, legata all'errore di generalizzazione attraverso il parametro d, che gli imporrà di fermarsi. Come si mostrerà anche sperimentalmente nel capitolo 4, la (2.1) è praticamente un'uguaglianza eccetto che per situazioni nelle quali la realtà da apprendere è tale da consentire con probabilità non trascurabile che il classificatore costruito dall'algoritmo riesca a coprire l'intero spazio dei campioni prima che la condizione che coinvolge d si verifichi.

Nel capitolo 5 si dimostra che la (2.1) non potrebbe valere se i classificatori generati non ammettessero il rigetto. Il lettore interessato alla dimostrazione della (5) è invitato alla lettura dei capitoli 3, 6 e 7. Nel capitolo 3 si presenta un algoritmo molto simile ad RealGPE, IdealGPE, per il quale la relazione (2.1) vale sempre all'uguaglianza e che consente di fare maggior luce sulle proprietà di RealGPE.

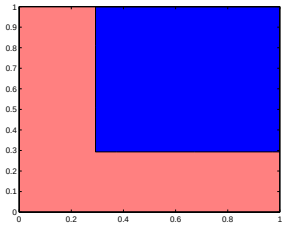
Il funzionamento dell'algoritmo è certo meglio comprensibile attraverso qualche esempio, che sarà dato di seguito.

2.2.3 Esempi grafici

Si mostrerà l'algoritmo all'opera con campioni prelevati da $X = [0,1]^2$, distribuiti in modo artificiale con lo scopo di verificarne in casi elementari e facilmente visualizzabili il comportamento. Per la verifica delle garanzie teoriche e prove su dati reali si rimanda al capitolo 4. Si utilizzerà una distribuzione uniforme su X, generata mediante la funzione `rand()` di MATLAB® [44]. L'apprendimento riguarderà due tipi di concetti [3] descritti dalle seguenti funzioni:

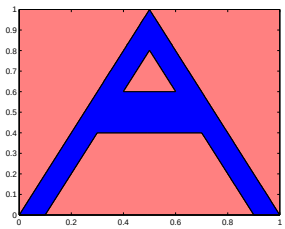
Quadrato in un angolo

$$y(\mathbf{x}) = \begin{cases} 1 & \text{se } 1 - \frac{\sqrt{2}}{2} \leq x_2 \leq 1 \text{ e } 1 - \frac{\sqrt{2}}{2} \leq x_1 \leq 1 \\ 0 & \text{altrimenti} \end{cases} \tag{2.2}$$



Lettera A

$$y(\mathbf{x}) = \begin{cases} 1 & \text{se } 0.4 \leq x_2 \leq 0.6 \text{ e } |x_1 - \frac{1}{2}| \leq \frac{1}{2} - \frac{x_2}{2} \\ & \text{oppure se } |x_1 - \frac{1}{2}| \leq \frac{1}{2} - \frac{x_2}{2} \\ 0 & \text{altrimenti} \end{cases} \tag{2.3}$$



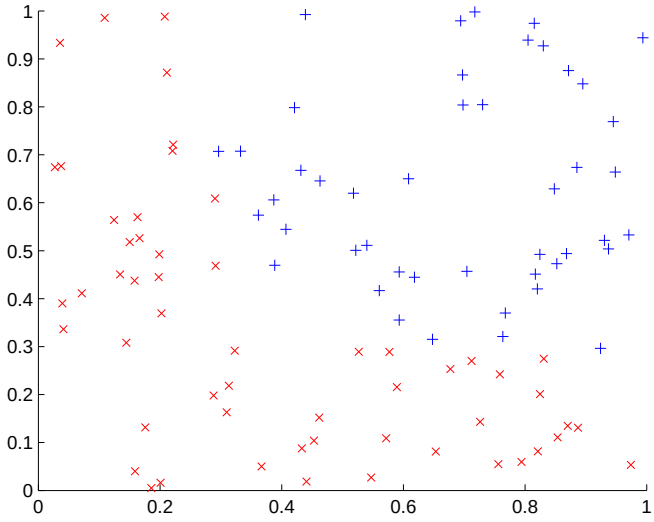
[3] L'uso del termine *concetto*, per indicare una funzione che assegna 1 o 0 a ciascun elemento di X, trova giustificazione in una vasta letteratura, specialmente laddove ci si pone come obbiettivo (anche se non è il nostro caso) di ricavare algoritmi o proprietà relativi all'apprendimento di circoscritte *classi di concetti*.

Nei grafici, il blu e il rosso rappresentano rispettivamente le due nature secondo cui sono etichettati i campioni.

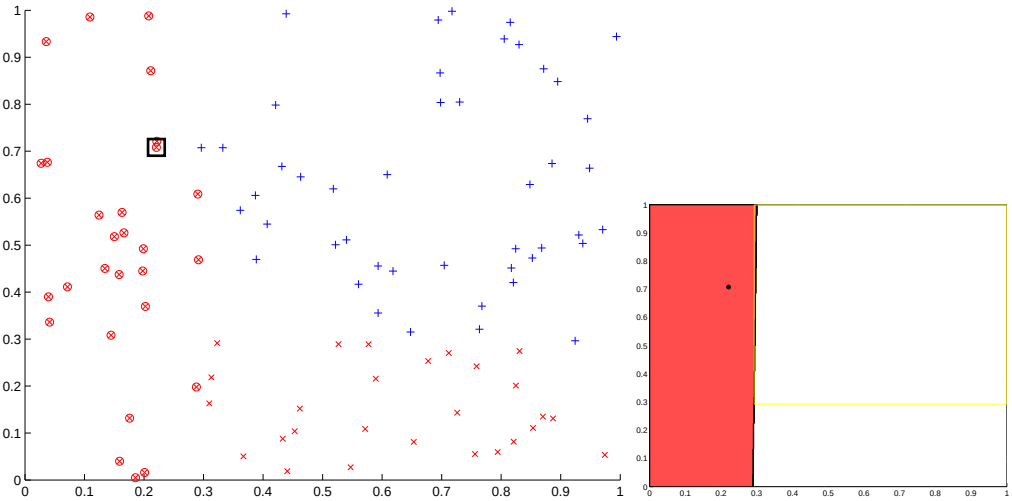
Per ognuno dei due concetti, vengono estratti 100 campioni. Il parametro d è posto in entrambi i casi a 5 (ciò significa che ci attendiamo che l'algoritmo, in media, generi dei classificatori che commettono un errore di generalizzazione sotto il 5%).

Quadrato Il primo caso è abbastanza semplice. Eseguendo l'algoritmo con $d = 5$ si scopre che il classificatore generato è in grado di coprire l'intero spazio dei campioni.

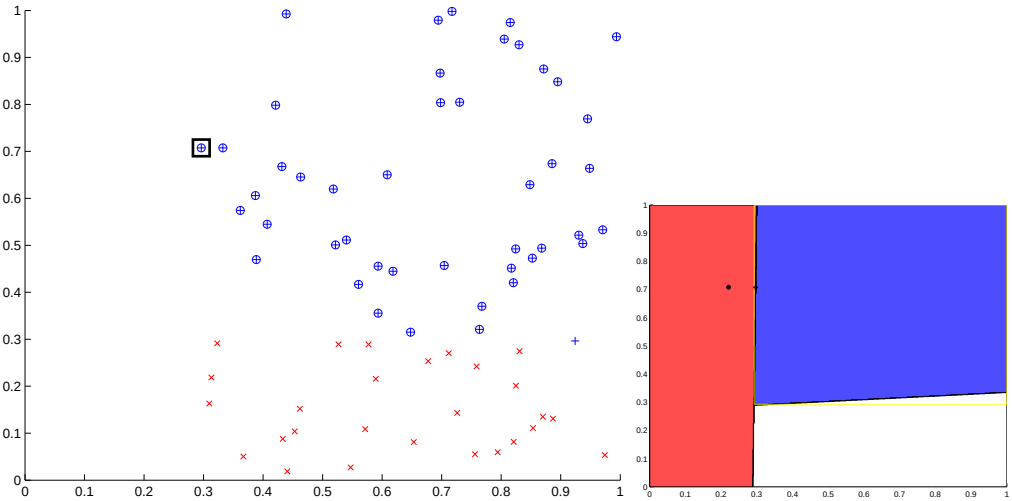
- *Campioni estratti*
  Sono mostrati i campioni estratti in $[0, 1]^2$ rappresentati diversamente a seconda della loro natura: con crocette greche ("+") quelli etichettati con 1 (blu), con crocette di Sant'Andrea ("x") quelli che valgono 0 (rosso). Uno di essi (il primo estratto) sarà utilizzato come primo punto di base.



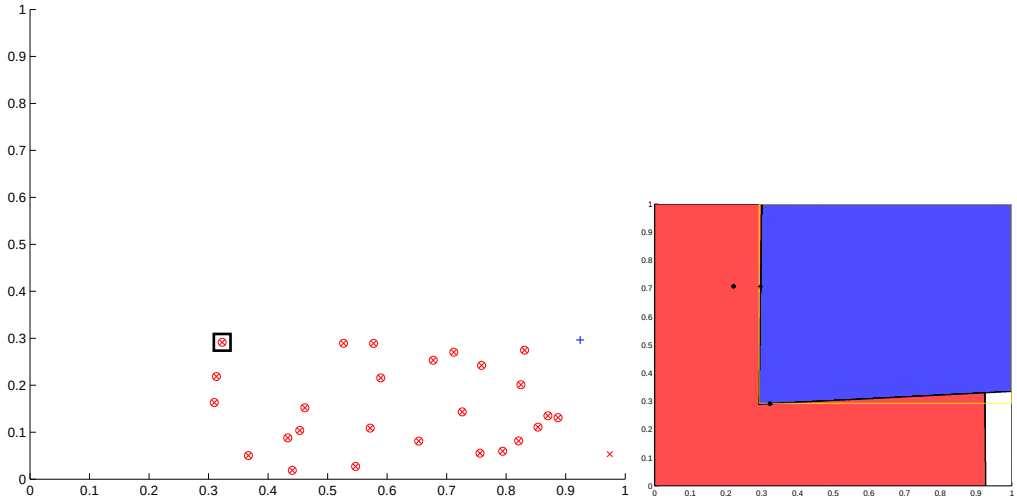
- *Azione della prima regione costruita* La prima regione è costruita risolvendo $P_{\frac{n(n+1)}{2}+n}$, cioè P_5 e tocca un solo punto (cioè ammette un vincolo attivo) che viene aggiunto ad S. Sotto a destra, si vede che la funzione di decisione inizia a classificare entro la regione generata.



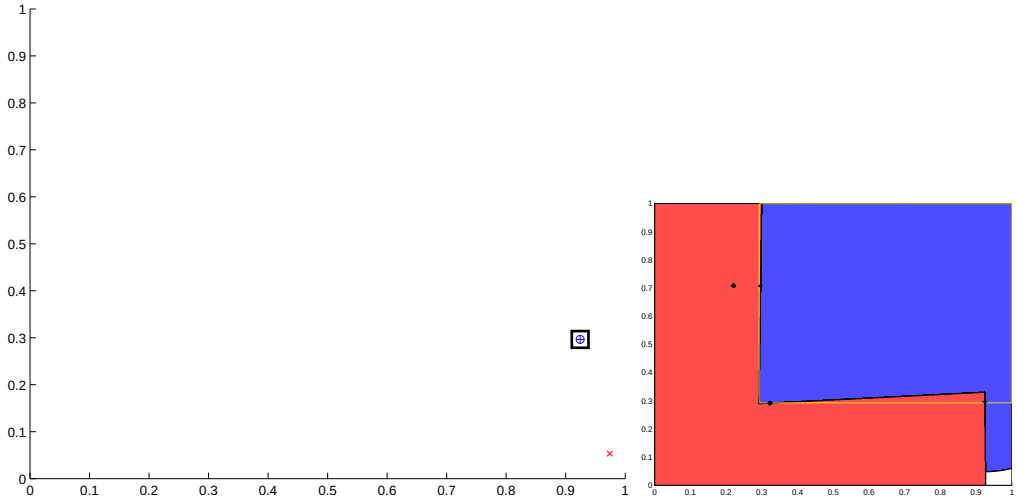
- *Azione della seconda regione sui campioni non ancora coperti* Poiché $|S| + \frac{n(n+1)}{2} + n = 1 + 5 > 5$, ma $|S| + n + 1 = 1 + 3 = 4 < 5$, la nuova regione è costruita risolvendo P_{n+1}. Essa conduce all'aggiunta di un altro punto ad S.



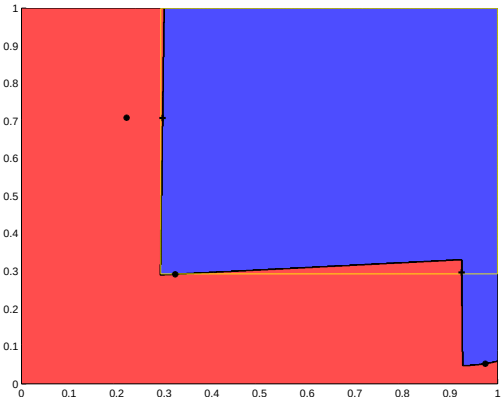
- *Azione della terza regione sui campioni non ancora coperti* Poiché $|S| + n + 1 = 2 + 3 = 5$, la nuova regione è costruita risolvendo ancora P_{n+1}. Essa conduce ancora all'aggiunta di un punto ad S.



- *Azione della quarta regione sui campioni non ancora coperti* Poiché $|S| + n + 1 = 3 + 3 = 6 > 5$, la nuova regione è costruita risolvendo P_{CMax}, costruendo cioè la circonferenza di raggio massimo centrata sul campione evidenziato. Un nuovo punto (l'unico rosso rimasto, in basso a destra) entra in S.

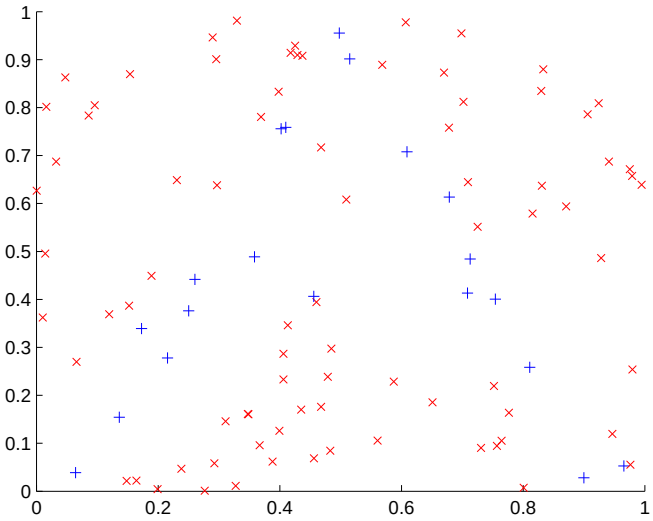


- *Azione dell'ultima regione sui campioni non ancora coperti* La nuova regione è costruita risolvendo ancora P_{CMax}. Mancando però a questo punto campioni blu, la regione copre l'intero spazio e ad S_c viene assegnato l'insieme vuoto. La condizione $S_c = \emptyset$ fa sì che l'algoritmo termini.

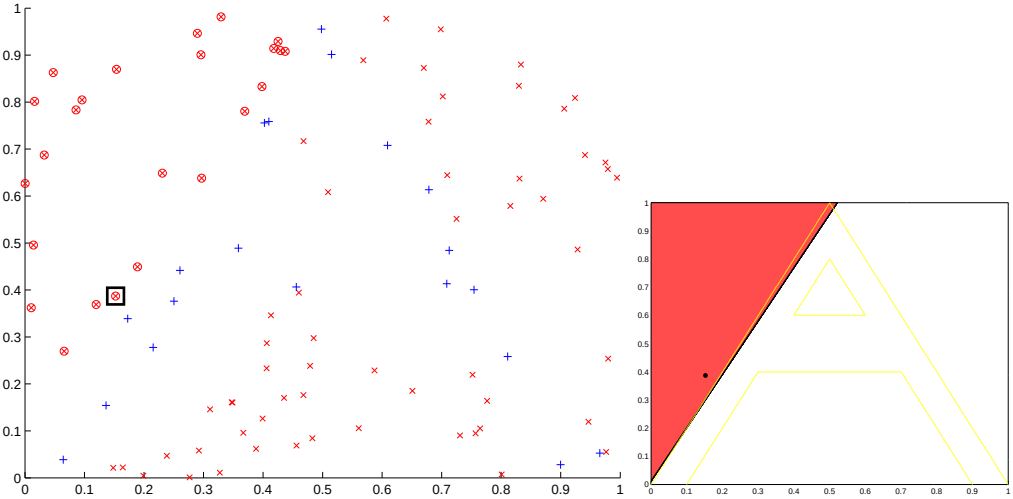


Lettera A Sempre con lo stesso parametro $d = 5$ si mostra l'algoritmo in azione quando la realtà da apprendere è più complicata. L'algoritmo non classificherà tutto lo spazio, l'errore è però sotto controllo.

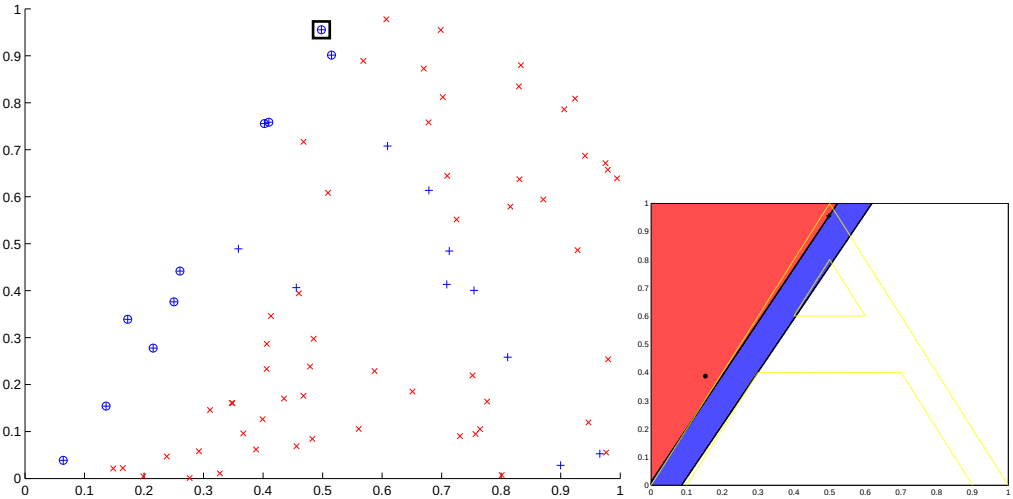
- *Campioni estratti*



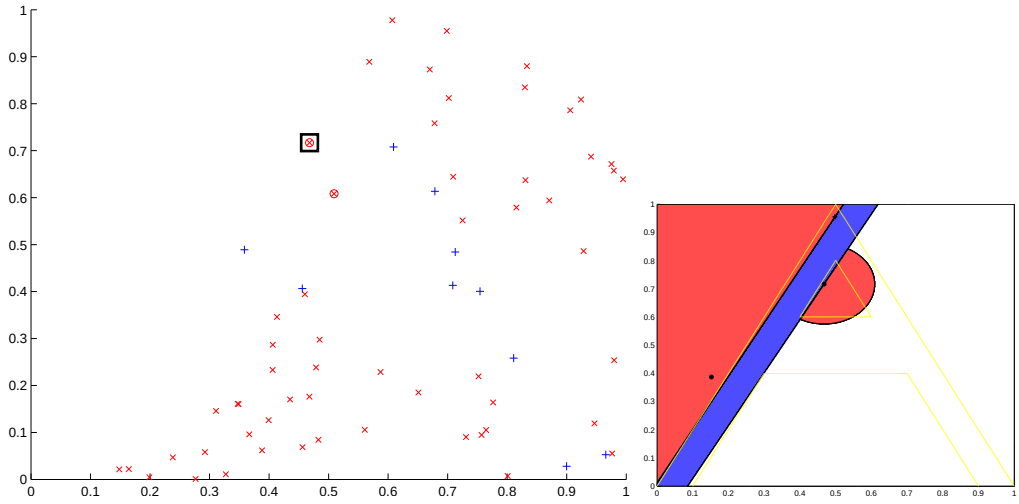
- *Azione della prima regione costruita*
  La prima regione è ottenuta ancora ottimizzando P_5, e tocca due punti.



- *Azione della seconda regione sui campioni non ancora coperti*
  Si risolve P_3 e la regione generata tocca altri due punti.



- *Azione della terza regione sui campioni non ancora coperti*
  La terza regione è ottenuta da P_{CMax}, in quanto, fin qui $|S| = 4$. Risolvendo P_{CMax} si ottiene esattamente un punto sul bordo della circonferenza costruita e si raggiunge la condizione $|S| = 5 = d$. L'algoritmo, pertanto, termina, lasciando una buona parte di spazio non classificata.



Capitolo 3

IdealGPE

In questo capitolo si propone un algoritmo per la classificazione binaria con rigetto, IdealGPE, per il quale è ***esattamente* nota a priori** (indipendente dalla densità di probabilità definita su X e da y) la distribuzione dell'errore di classificazione. IdealGPE ha una caratteristica che, a rigore, lo rende "ideale", cioè non realmente implementabile. Tuttavia, si mostrerà che il suo funzionamento è simulabile. Il ruolo rivestito da IdealGPE all'interno del presente lavoro è essenzialmente quello di strumento per dimostrare le proprietà di RealGPE, realmente implementabile e già descritto nel capitolo 2, la cui distribuzione dell'errore è sempre o uguale o migliore di quella di IdealGPE.

3.1 Caratteristiche principali

L'algoritmo, come RealGPE, riceve in ingresso un parametro d. Sotto le stesse ipotesi, la probabilità di errore dell'algoritmo addestrato su N campioni è caratterizzata dalla seguente:

$$\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon) = \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1} \tag{3.1}$$

In termini di distribuzione dell'errore di generalizzazione (F_{V_N}) si ha allora la 3.2

$$\begin{aligned} F_{V_N}(\epsilon) &= 1 - \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1} \\ &= \binom{N-1}{d} d \int_0^\epsilon (1-\alpha)^{N-d-1} \alpha^{d-1} \mathrm{d}\alpha \end{aligned} \tag{3.2}$$

Si noti che la distribuzione, scritta nella formula integrale secondo l'ultima eguaglianza (3.2), coincide con la ben nota espressione della funzione Beta

di Eulero[1] incompleta regolarizzata, e si può dunque scrivere

$$F_{V_N}(\epsilon) = \frac{B(\epsilon; d, N-d)}{B(d, N-d)}.$$

Naturalmente, se tale distribuzione non offrisse la speranza di buoni risultati, non si giustificherebbe l'interesse per un algoritmo che da essa è caratterizzato. Fissato $N = 500$ e tracciando il grafico di $1 - F_{V_N} = \mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon)$ per alcuni valori di d (si veda la figura 3.1) si può osservare come possano essere desiderabili algoritmi in grado di produrre simili distribuzioni dell'errore. Per esempio, per $d = 2$, la probabilità che un classificatore costruito a partire da 500 campioni commetta un errore di generalizzazione maggiore del 3% non supera l'ordine di uno su un milione. Inoltre, dalla distribuzione dell'errore di generalizzazione si può ricavare il valore atteso:

$$\mathbf{E}\left[V_N\right] = \frac{d}{N} \tag{3.3}$$

Intuitivamente, più si vuole ridurre il tasso di rigetto, cioè lo spazio non classificato, più si corre il rischio di commettere errori, specialmente laddove le nature siano distribuite in modo complesso nello spazio dei campioni. Si può pensare ad IdealGPE come ad un algoritmo che cerca di spingersi a coprire lo spazio dei campioni fintantoché gli è possibile garantire, allo stesso tempo, l'errore. Dalla definizione dell'algoritmo, nella sezione 3.3, si vedrà che, fino ad un certo punto della sua esecuzione, IdealGPE si comporta esattamente come RealGPE. Questa fase può essere definita "costruttiva" in quanto ha per fine la copertura progressiva dello spazio dei campioni. A differenza di RealGPE, però, IdealGPE deve garantire *sempre* che la distribuzione dell'errore di generalizzazione sia rispettata, deve gestire perciò anche il caso in cui la realtà si riveli particolarmente *semplice* e si pervenga ad una precoce copertura completa dello spazio dei campioni. L'esecuzione di IdealGPE, in questi

[1] La funzione Beta è stata studiata da Eulero nel XVIII secolo, ma la sua "popolarità" non è certo diminuita, specialmente da quando, alla fine degli anni '60, Gabriele Veneziano e Mahiko Suzuki la riscoprirono nell'ambito della fisica delle interazioni forti, aprendo il campo alla teoria delle stringhe. La funzione Beta è definita come

$$B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1}\mathrm{d}t.$$

La funzione Beta incompleta, al numeratore nel corpo testo, è

$$B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1}\mathrm{d}t.$$

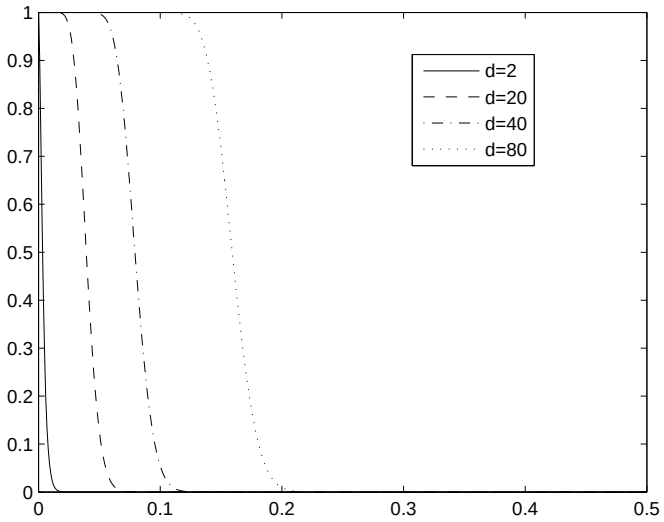


Figura 3.1: Con $N = 500$, grafici di $\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon)$ (ordinata), al variare di ϵ (ascissa), per diversi valori di d

casi, non termina, ma entra in una fase "distruttiva", al fine di aumentare deliberatamente l'errore di generalizzazione. Si prenda un esempio estremo, per comprendere che cosa ciò significhi in pratica: supponiamo di avere una distribuzione di campioni per cui si estrarrà un campione di natura 1 con probabilità solo di 10^{-10}; con $d = 80$ l'algoritmo dovrà comunque garantire una distribuzione dell'errore come quella in figura 3.1 e con $d = 499$ dovrà garantire un valore atteso per l'errore di generalizzazione del 99.8%! Nella sezione successiva si spiega qual è l'accorgimento adottato da IdealGPE per assicurare simili risultati.

3.2 Un algoritmo ideale

A rigore, IdealGPE non è realmente implementabile in quanto i classificatori da esso prodotti possono essere caratterizzati da una funzione $\widehat{y}_N$ che, a fronte di un campione in ingresso x, può comportarsi in uno dei seguenti modi

1. classificare il campione: $\widehat{y}_N(x)$ assume il valore 0 o 1
2. rigettare il campione: $\widehat{y}_N(x) = \mathsf{R}$
3. **classificare in modo sicuramente errato il campione**:

$$\widehat{y}_N(x) = \begin{cases} 1 & \text{se } y(x) = 0 \\ 0 & \text{se } y(x) = 1 \end{cases}$$

Ciò che può capitare è che la funzione di classificazione prodotta sia in grado di misclassificare sicuramente qualsiasi campione per campioni appartenenti ad un *determinanto* sottoinsieme di X. Normalmente, la regione di misclassificazione associata ad una certa $\widehat{y}_N$, per essere determinata, richiede la conoscenza e di $\widehat{y}_N$ e di y; in tal caso, invece, alcune parti della regione di misclassificazione, che chiameremo **aree di misclassificazione estesa**, sono determinabili a partire unicamente da $\widehat{y}_N$! Ovviamente, un classificatore in grado di misclassificare con sicurezza campioni non ancora osservati senza una conoscenza surrettizia della funzione y è irrealizzabile. Senza contare che, se si sapesse di sbagliare sicuramente, allora si potrebbe classificare correttamente con la stessa certezza! Tuttavia, classificatori di questo tipo sono simulabili banalmente, per esempio in un test atto a verificare le proprietà dell'algoritmo[2]: in corrispondenza dei campioni nell'area di misclassificazione estesa, $\widehat{y}_N$ può assumere un valore speciale; tale valore segnala a chi opera con l'algoritmo che il campione in esame va *conteggiato come se fosse stato misclassificato*. Evidentemente, è già chiaro a questo punto che, se anziché conteggiare alla cieca la previsione come errata, ci si sforzasse almeno di azzardarla, si potrebbe solo migliorare la prestazione dell'algoritmo, poiché, per esempio, basando l'esito della classificazione sul lancio casuale di una monetina si sbaglierà il 50% delle volte anziché il 100%. Si conclude attraverso questo semplice ragionamento che, se per IdealGPE la distribuzione dell'errore è nota ed è in accordo con la 3.1, per un algoritmo che produce classificatori uguali in tutto a quelli prodotti da IdealGPE, fuorché nella presenza di una zona di misclassificazione estesa, l'errore sarà necessariamente minore o uguale. Come sarò presto chiaro, quello delineato è in effetti il rapporto esistente tra IdealGPE e RealGPE.

Passiamo ora alla descrizione dell'algoritmo IdealGPE, per la gran parte identica a quella di RealGPE, con la differenza che, al verificarsi della condizione per cui l'intero spazio dei campioni è interamente coperto e il numero di punti raccolti nell'insieme S è ancora minore di d, viene costruita opportunamente una regione ipersferica di *misclassificazione estesa* centrata attorno al primo punto dell'insieme di addestramento.

3.3 Definizione di IdealGPE

☞

[2] È ciò che si è fatto nella sezione 4.1.

Si assume di estrarre *casualmente* gli N esempi di addestramento $((x_0, y_0), (x_1, y_1), ..., (x_{N-1}, y_{N-1}))$, posti in ingresso all'algoritmo insieme al numero d, $1 \leq d < N$

1. Si sceglie x_0 come primo punto di base ($x_b := x_0$). Si inizializza l'insieme $S := \emptyset$. L'insieme L contiene inizialmente tutti i punti nell'insieme di addestramento. L'algoritmo, al termine, avrà costruito una successione di m regioni, rappresentate da coppie (ξ_i, κ_i), $i = 1, ..., m$, ciascuna delle quali formata dal sottoinsieme $\xi_i \in \mathbb{R}^n$ e da un marcatore κ_i che può assumere i valori 1 o 0. L'insieme $\mathcal{M} \in R^d$ descriverà l'area di misclassificazione estesa. m è una variabile che corrisponde ad un numero naturale ed è inizializzata a 1

2. - SE $|S| + \frac{n(n+1)}{2} + n \leq d$, ALLORA si trova la soluzione A^*, B^* del problema convesso $P_{\frac{n(n+1)}{2}+n}(x_b, y(x_b))$, definito come segue:

$$
\begin{aligned}
&P_{\frac{n(n+1)}{2}+n}(x_b, y(x_b)) : \\
&\qquad \min_{A=\in\mathbb{R}^{n\times n}, B\in\mathbb{R}^{1\times n}} \ \text{Traccia}(A) \\
&\qquad \text{con vincoli:} \quad (x_j - x_b)^T A(x_j - x_b) + B(x_j - x_b) \geq 1, \\
&\qquad\qquad\qquad\qquad \forall x_j \in \mathrm{L}, y(x_j) \neq y(x_b) \\
&\qquad\qquad\qquad\qquad A = A^T \\
&\qquad\qquad\qquad\qquad A \succeq 0
\end{aligned}
$$

   Nel caso di soluzioni multiple, si sceglie quella che minimizza la norma L2 di B ($||B|| = \sqrt{\sum_{i=1}^{n} b_i^2}$). Se ancora ci sono più soluzioni, allora si sceglie quella per cui l'elemento $a_{1,1}$ di A è minore; se ciò non basta ad individuarne soltanto una, si sceglie tra le candidate quella che minimizza $a_{1,2}$, e così via con le altre componenti della matrice, finché non si individua una sola soluzione.

   - SE $|S|+n+1 \leq d < |S|+\frac{n(n+1)}{2}+n$, ALLORA si trova la soluzione k^*, B^* del problema lineare $P_{n+1}(x_b, y(x_b))$, definito come segue:

$$
\begin{aligned}
&P_{n+1}\ (x_b, y(x_b)): \\
&\qquad \min_{k\geq 0, B\in\mathbb{R}^{1\times n}} \ k \\
&\qquad \text{con vincoli:} \quad k||x_j - x_b||^2 + B(x_j - x_b) \geq 1, \\
&\qquad\qquad\qquad\qquad \forall x_j \in \mathrm{L}, y(x_j) \neq y(x_b)
\end{aligned}
$$

     e si impone $A^* = Ik^*$, $A \in \mathbb{R}^{n\times n}$ (I matrice identità) Nel caso di soluzioni multiple si sceglie quella che minimizza $||B||$.

   - SE $d < |S|+n+1$, ALLORA si trova la soluzione k^*, del problema lineare $P_{\mathrm{CMax}}(x_b, y(x_b))$, definito come segue:

$$
\begin{array}{lll}
P_{\mathrm{CMax}}(x_b, y(x_b)): & & \\
\quad \min_{k\geq 0} & k & \\
\quad \text{con vincoli:} & k||x_j - x_b||^2 \geq 1, & \\
 & & \forall x_j \in \mathrm{L}, y(x_j) \neq y(x_b)
\end{array}
$$

e si impongono $A^* = Ik^*$, $A \in \mathbb{R}^{n\times n}$ (I matrice identità), e $B^* = 0$

3. Si definisce la regione m-esima (ξ_m, κ_m),

   $\xi_m := \{x \in X t.c. (x - x_b)^T A^* (x - x_b) + B^*(x - x_b) < 1\}$
   $\kappa_m := y(x_b)$

4. Si tolgono da L i punti che appartengono a ξ_m.

5. Chiamiamo S_c l'insieme dei punti $x \in L$, con $y(x) \neq y(x_b)$, e che soddisfano con l'eguaglianza il vincolo

   $$(x - x_b)^T A(x - x_b) + B(x - x_b) \geq 1$$

   L'insieme S viene ridefinito come $S := S \cup S_c$.

6. SE $S_c = \emptyset$ ALLORA

   - SE $|S| < d$, $\mathcal{M} :=$ EXTMISC$(x_0, x_1, ..., x_{N-1}, S, n)$ e **TERMINA**
   - ALTRIMENTI **TERMINA**

7. Il punto di S_c a maggior distanza (in norma) da x_b viene scelto come nuovo punto di base e ribattezzato x_b. In caso tale punto non fosse unico, si prende quello la cui prima componente è minore; a parità di prima componente si guarda la seconda, etc.

8. $m := m + 1$ (la variabile m viene incrementata di un'unità) e si torna al punto 2.

☜

Occorre definire la procedura EXTMISC, che ha come input i punti dell'insieme di addestramento, l'insieme dei punti corrispondenti a vincoli attivi finora individuati S e il parametro n.

EXTMISC ☞
Si inizializzano: $\mathcal{M} := \emptyset$ e $S_{\text{esteso}} := S$.

1. Si trova la soluzione k^* che risolve il problema $P_{\text{CMax}'}(x_0)$ (si noti che è diverso da P_{CMax} a due argomenti sopra definito in quanto qua non conta la natura del punto di base, né i vincoli sono selezionati in base alla loro natura).

   $P_{\text{CMax}'}(x_0)$:
   $\min_{k\geq 0}$ k
   con vincoli: $k||x_j - x_0||^2 \geq 1$,
   $j = 1, ..., N-1$ t.c. $x_j \notin S_{\text{esteso}} \cup \mathcal{M}$

2. Chiamiamo S_c l'insieme dei punti $x_j \notin \mathrm{S} \cup \mathcal{M}$ $(j = 1, ..., N-1)$ che soddisfano con l'eguaglianza il vincolo

   $$k^*||x_j - x_0||^2 \geq 1$$

3. Si ridefiniscono

   $S_{\text{esteso}} := S_{\text{esteso}} \cup S_c$
   $\mathcal{M} := \{x \in X \text{ t.c. } k^*||x - x_0||^2 \leq 1\} \setminus \{x_0, x_1, ..., x_{N-1}\}$

4. SE $|S_{\text{esteso}}| < d$, si torna al punto 1; ALTRIMENTI **TERMINA restituendo** $\mathcal{M}$ (CON AVVERTIMENTO SE $|S_{\text{esteso}}| > d$)

☜
Così, l'algoritmo descritto produce una successione finita di m regioni, rappresentate da coppie (ξ_i, κ_i) con $i = 1, ..., m$, ed un insieme $\mathcal{M}$ di misclassificazione estesa. A tali strutture si associa una funzione di decisione (il classificatore), $\widehat{y}_N$, così definita:

$$\widehat{y}_N(x) = \begin{cases} \textbf{misclassificato} & \text{se } x \in \mathcal{M} \\ \mathsf{R} & \text{se } x \notin \mathcal{M} \text{ e } \nexists\, j,\ 1 \leq j \leq m, \text{ t.c. } x \in \xi_j \\ \kappa_i & \text{altrimenti, con} \\ & i = \min\{j,\ 1 \leq j \leq m,\ x \in \xi_j\} \end{cases}$$

Capitolo 4

Verifiche sperimentali

Il presente capitolo conclude l'esposizione delle caratteristiche di RealGPE dal punto di vista del suo impiego pratico. Nella sezione 4.1 si mostrano conferme sperimentali delle proprietà di RealGPE e di IdealGPE, che peraltro hanno il pregio di evidenziare le ragioni della possibile laschezza del maggiorante (in probabilità) sull'errore di generalizzazione per RealGPE: all'aumentare di d o, d'altro lato, al semplificarsi della realtà da apprendere, IdealGPE è portato ad introdurre regioni di misclassificazione estesa, le quali rendono conto della laschezza. Nella sezione 4.2, RealGPE è messo alla prova su dati reali e le sue prestazioni si rivelano competitive.

4.1 Verifica sperimentale delle proprietà su dati artificiali

In questa sezione si fornisce un confronto tra i risultati teorici attesi e quelli empirici[1]. Sì è fatto riferimento fin dalla sezione 1.4 alla probabilità dell'insieme di misclassificazione, cioè all'errore di generalizzazione, di un certo classificatore $\widehat{y}$, per il calcolo esatto del quale occorre conoscere la distribuzione μ e la funzione y. La bontà di un algoritmo di classificazione dipende da come si distribuisce l'errore di generalizzazione, che è una variabile casuale, funzione delle N-uple di addestramento.

[1] RealGPE e IdealGPE sono stati implementati in MATLAB® v.7.4 (si veda [44]). Per i problemi di ottimizzazione si è sfruttato il risolutore CVX [46]. Si esprime inoltre gratitudine all'Ing. Fabrizio Guerrini per aver messo a disposizione del codice (relativo ad un diverso algoritmo di classificazione e alle relative procedure di test) che è stato prezioso oggetto di studio e riuso.

Per esempi sintetici come quelli presentati nel capitolo 2 si conosce tutto ciò che serve per valutare, dato un certo classificatore, il suo errore di generalizzazione. Estraendo, per esempio, un numero sufficiente di N-uple è allora possibile farsi un'idea piuttosto precisa di come si distribuiscono gli errori di generalizzazione e, quindi, ottenere una stima di $\mu^N(\mathrm{PE}(\widehat{y}_N) > \varepsilon)$ al variare di ε, costruendo il relativo istogramma.

Per ciascun istogramma presente in questa sezione, l'algoritmo è stato eseguito 1000 volte su multi-estrazioni di 100 ($N = 100$) campioni ciascuna, con etichette in accordo con la funzione (2.3) (*Lettera A*) e una distribuzione uniforme dei campioni in $[0, 1]^2$. Insieme agli istogrammi è tracciata con linea continua la funzione che determina la distribuzione di probabilità (teorica) dell'errore di generalizzazione:

$$\mu^N(\mathrm{PE}(\widehat{y}_N) > \varepsilon) = \sum_{i=0}^{d-1} \binom{N-1}{i} \varepsilon^i (1-\varepsilon)^{N-i-1}$$

Gli istogrammi sono stati ricavati suddividendo con passo 0.001 l'asse $[0, 1]$, sul quale varia ε, e contando per ogni valore discreto così ottenuto il numero di classificatori per i quali l'errore di generalizzazione (stimato) supera tale valore e normalizzando infine il risultato mediante la divisione per 1000.

Il calcolo dell'errore di generalizzazione per ciascuno dei 1000 classificatori generati è stato *stimato* col metodo Monte Carlo: il test è avvenuto calcolando il tasso di errore su 1000 campioni generati attraverso la funzione rand() ed etichettati secondo la funzione *Lettera A*.

L'istogramma in figura 4.1(a) rende conto dell'esecuzione dell'algoritmo per $d = 5$, valore per il quale il numero di campioni non si è mai esaurito prima che il computo dei punti in S raggiungesse d.

L'istogramma in figura 4.2(a) è stato ottenuto con $d = 25$. Con un parametro d così alto, capita che l'algoritmo termini prima che $|S|$ raggiunga d.

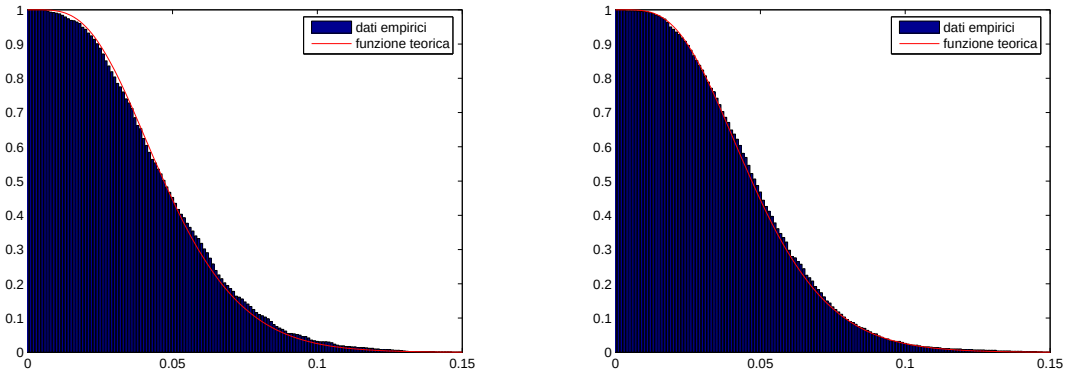
Gli istogrammi 4.1(b) e 4.2(b) derivano dall'esecuzione di IdealGPE. In questo caso, quando il classificatore riduce a zero la probabilità di rigetto ma $|S|$ è ancora minore di d, vengono introdotte le regioni di misclassificazione estesa. I campioni generati col metodo Monte Carlo che cadevano entro tali regioni sono stati conteggiati come misclassificati. Com'era prevedibile, l'istogramma per $d = 5$ è molto simile[2] a quello in figura 4.2(a). Per $d = 25$ la differenza si fa invece più marcata. Questo è dovuto al fatto che IdealGPE

[2]I due istogrammi non sono identici: il motivo è che le 1000 multi-estrazioni utilizzate in questo caso sono diverse da quelle precedenti e il metodo Monte Carlo utilizza ogni volta campioni diversi.

utilizza più spesso regioni di misclassificazione estesa. Il fenomeno si fa più netto al crescere di d o quando la realtà si fa più semplice. Per esempio, poniamo che l'apprendimento riguardi la banale funzione (4.1) definita su $[0,1]^2$ (probabilità uniforme)

$$y(\mathbf{x}) = \begin{cases} 1 & \text{se } x_1 < 0.5 \\ 0 & \text{altrimenti} \end{cases} \tag{4.1}$$

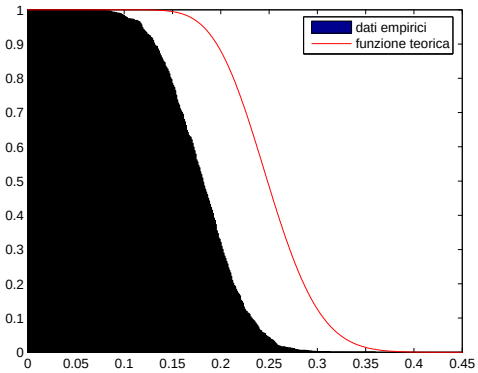
in cui la natura dei campioni dipende solo dalla loro appartenenza al semipiano $x_1 < 0.5$. Se poniamo $d = 15$, affinché siano realmente individuati 15 punti di supporto, l'algoritmo deve introdurre grandi regioni di misclassificazione estesa nei classificatori generati. I risultati sono presentati negli istogrammi 4.3(a) e 4.3(b).



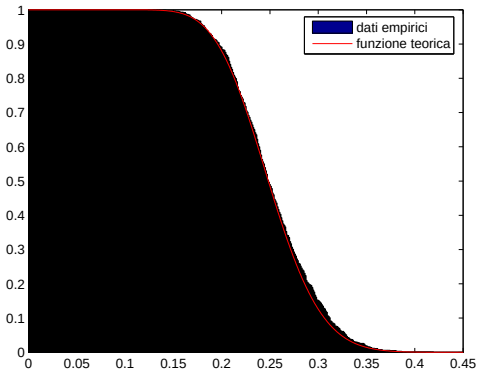
(a) RealGPE

(b) IdealGPE

Figura 4.1: $d = 5$, Lettera A

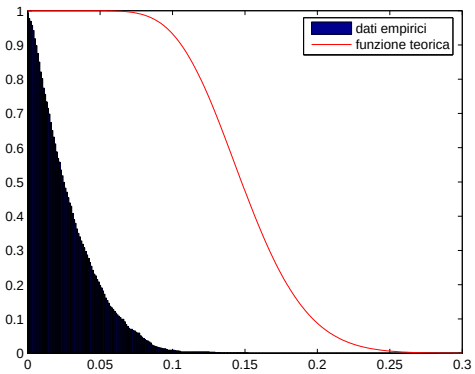


(a) RealGPE

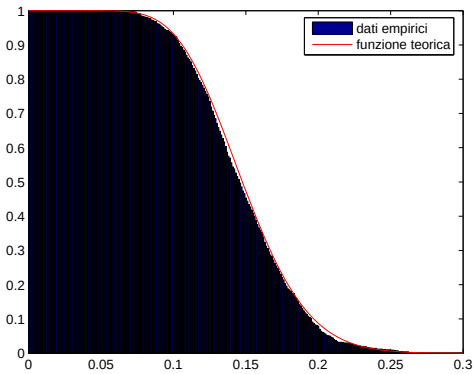


(b) IdealGPE

Figura 4.2: $d = 25$, Lettera A



(a) RealGPE



(b) IdealGPE

Figura 4.3: $d = 15$, Campioni linearmente separabili

4.2 Risultati su dati reali

La parte restante del capitolo è articolata nel seguente modo. Nella prima sezione si opera un confronto tra l'algoritmo proposto e alcuni dati riportati nell'articolo di presentazione delle Set Covering Machine (SCM) in [43], relativi alle SCM e a loro concorrenti. Nella sezione successiva si utilizzano dati tratti dal celebre insieme di dati (*dataset*) Iris e si effettua un breve confronto con i risultati ottenuti addestrando delle SVM (senza rigetto). Nella terza sezione ci si sofferma su due dataset nel dominio del riconoscimento di caratteri, tracciando il grafico dei risultati ottenuti dall'algoritmo secondo le cosiddette *curve A-R* (accuratezza, rigetto), delle quali si mostrano due esempi tratti da [27] relativi alla valutazione di SVM con rigetto applicate allo stesso dominio.

Infine, si riportano in forma tabellare i risultati di svariate esecuzioni dell'algoritmo per alcuni valori di d su tutti i dataset.

I dataset utilizzati

Sezione 4.2.1

1. Breast: 683 campioni; 10 attributi (interi)
2. Haberman: 294 campioni; 3 attributi (interi)
3. Pima: 768 campioni; 8 attributi (interi e reali)
4. Bupa: 345 campioni; 6 attributi (interi e reali)
5. Credit: 653 campioni; 15 attributi (categorici, reali, interi)
6. Glass: 163 campioni; 9 attributi (**reali**)

Sezione 4.2.2

1. Iris Setosa VS Versicolor e Virginica: 149 campioni; 4 attributi (**reali**)
2. Iris Versicolor VS Virginica: 99 campioni; 4 attributi (**reali**)

Sezione 4.2.3

1. Lettere AH: 1523 campioni; 16 attributi (interi)
2. Lettere FT: 1571 campioni; 16 attributi (interi)

Questi dati sono tratti dalla raccolta dell'*UCI* (si veda [2]), con l'eccezione di Glass, che invece è stato reperito dal sito di Tom Bylander [8] e semplificato: quello nel sito è più complesso ed è stato ridotto differenziando solo vetri di finestra prodotti con il sistema a galleggiamento (float) da quelli prodotti con altre tecniche (non-float). Breast e Pima sono stati utilizzati nella versione scalata tra $[-1, 1]$ disponibile al sito [11]. Per gli altri si è proceduto ad una normalizzazione che imponesse a ciascun attributo media nulla e covarianza unitaria. In ogni caso, sono stati rimossi campioni etichettati in maniera contraddittoria.

Per gli esempi che hanno solo attributi in spazi finiti o discreti, le ipotesi dell'algoritmo non sono soddisfatte a rigore e il risultato teorico non è quindi applicabile. Anche nel caso di attributi misti, non essendo definibile una densità, non si ha la garanzia a priori che non si possano verificare casi degeneri. In entrambe le situazioni, si può comunque eseguire l'algoritmo e valutarne il funzionamento, per verificare che si comporti in maniera competitiva o almeno confrontabile con algoritmi che, pure, nella sostanza non forniscono garanzie circa l'errore.

4.2.1 Confronto con SCM, SVM, Nearest Neighbour Classifier

Con SCM, facciamo riferimento al classificatore detto *Set Covering Machine* (SCM) introdotto da M.Marchand e J.Shawe-Taylor in [43] e brevemente descritto nella sezione 1.5.7 a pagina 17. Ricordiamo che i parametri a disposizione dell'utente, per influenzare la possiblità di misclassificazioni nell'insieme di addestramento, sono: p (continuo) ed s (intero).

Per un tentativo di confronto "alla pari" con le SCM, che non ammettono l'opzione di rigetto, si riportano nella colonna centrale della tabella 4.1 i risultati ottenuti da RealGPE quando l'intero spazio dei campioni risulta classificato[3].

Il **metodo di test** impiegato per trovare i dati nella tabella 4.1 è la validazione incrociata 10-fold. Con questo metodo, un decimo dei dati si usa

[3]Si ponga attenzione al fatto che utilizzare l'algoritmo per d crescenti fino al raggiungimento della completa copertura dello spazio dei campioni e, a quel punto, fermarlo e tenere come funzione di decisione il classificatore così prodotto, significa mettere in campo una procedura che è essa stessa un algoritmo di classificazione, ed è un algoritmo *diverso* da quello per il quale i risultati sono garantiti, il quale presuppone che il parametro d sia scelto a priori e non stocasticamente. L'impostazione legittima da dare ai commenti circa i risultati dovrebbe rifarsi al seguente modello: "imponendo $d = 50$ ed eseguendo l'algoritmo RealGPE sui campioni di Glass, la media dell'errore di generalizzazione dei classificatori prodotti sarà al più del 34%. Si osserva che, eseguito l'algoritmo, nel nostro caso lo spazio dei campioni risulta totalmente coperto dal classificatore risultante".

dapprima come test per il classificatore ottenuto utilizzando come insieme di addestramento i nove decimi rimanenti; in seguito si ripete l'operazione utilizzando come insieme di test un altro decimo disgiunto dal precedente e così via per dieci volte. Per garantire la medesima dimensione dell'insieme di addestramento per ognuno dei dieci passi (e quindi calcolare agevolmente a priori la media dell'errore di generalizzazione avendo sempre lo stesso numero al denominatore), si è impiegato come insieme di test ad ogni passo un insieme di $\lfloor \frac{N}{10} \rfloor$ elementi [4], mentre i restanti costituivano l'insieme di apprendimento.

Si osservi che i dati per il confronto, copiati da [43] nelle tabelle 4.3, 4.4 e 4.2, sono pure i risultati di una validazione incrociata 10-fold, ma è molto improbabile che la suddivisione in 10 parti sia avvenuta allo stesso modo di quella che ha portato ai risultati della 4.1, così come potrebbero esserci differenze nella rappresentazione dei dati, per esempio per via di differenti normalizzazioni. Nell'ipotesi che il totale dei campioni nei 10 insiemi di test utilizzati in [43] desse davvero il totale dei campioni del dataset, i risultati della nostra validazione incrociata (che, invece, come si è spiegato, nel complesso utilizza per il test un numero di campioni leggermente inferiore al totale) sono stati incrementati moltiplicandoli per $\frac{N}{\lfloor N/10 \rfloor}$ ed è per questo che non sono interi (tabella 4.1 colonna a destra).

Dataset	Errori	Per il confronto
Breast	28	28.12
Haberman	84	84.14
Pima	241	243.54
Bupa	129	130.9
Credit	151	151.7
Glass	40	40.75

Tabella 4.1: Risultati di RealGPE in assenza di rigetto (copertura completa) in termini di numero di errori

Nel confrontare la tabella 4.1 con le 4.2 e 4.3, è bene tenere in considerazione il fatto che laddove vi era libertà nella scelta dei parametri, i risultati scritti per le SCM sono quelli per i quali era minimo l'errore di validazione incrociata ottenuto. Per esempio, nella tabella 4.3 le prime due colonne riportano per ciascuna riga *il* risultato ottenuto costruendo una SCM con parametri s e p posti ad infinito, mentre nelle ultime due è riportato il risultato per il valore di s, un intero, per il quale l'errore di crossvalidazione era minimo. Nella tabella 4.3 invece anche p può variare, per giunta con

[4] Si farà uso del simbolo $\lfloor \cdot \rfloor$ per indicare la funzione parte intera inferiore; $\lceil \cdot \rceil$ la parte intera superiore.

Dataset	Simple	Simple disg	Complex	Complex disg
Breast	18	16	15	16
Haberman	71	93	71	71
Pima	189	206	189	206
Bupa	109	106	121	107
Credit	198	195	198	194
Glass	35	36	33	36

Tabella 4.2: Errori di classificazione per SCM con centri sempre negativi (Simple) o anche positivi (Complex). Con congiunzioni e disgiunzioni (disg.) da [43]

Dataset	**RealGPE**	$s = \infty$	$s = \infty$ disg	$s < \infty$	$s < \infty$ disg
Breast	28.12	26	34	23	34
Haberman	84.14	104	128	76	127
Pima	243.54	262	230	262	208
Bupa	130.9	144	131	142	118
Credit	151.7	255	225	253	217
Glass	40.75	45	36	40	36

Tabella 4.3: Confronto tra RealGPE e SCM con $p = \infty$, congiunzioni e disgiunzioni (disg) [43]

continuità: per ogni riga, è riportato il valore migliore ottenuto dopo una scansione esaustiva di molti valori per i parametri p ed s.

Questo sicuramente può spiegare, almeno in parte, il fatto che dal confronto tra 4.1 e la 4.2 le SCM si rivelino più accurate. Vi è comunque l'eccezione di Credit, sul quale RealGPE ottiene un ottimo risultato. Quando il parametro p è invece infinito il confronto con RealGPE si fa agguerrito. RealGPE ottiene poi prestazioni confrontabili anche a quelle dei concorrenti delle SCM pure pubblicate in [43] e riportate in tabella 4.4. In questa tabella non si è riportato il dettaglio dei parametri utilizzati, che comunque sono ancora quelli che minimizzano l'errore di validazione incrociata, trovati a seguito di un grande numero di test (esaustivi). NNC indica un altro classificatore usato per il confronto basato sul metodo del "vicino più prossimo" (Nearest Neighbour Classifier). Brevi note su tecniche quali SVM ed NNC sono riportate nella sezione 1.5.

Pare di dover concludere che RealGPE dal confronto si rivela, nel complesso, un algoritmo competitivo.

Dataset	**RealGPE**	NNC	SVM $C = \infty$	SVM $C < \infty$
Breast	28.12	29	27	19
Haberman	84.14	107	111	71
Pima	243.54	247	243	203
Bupa	130.9	124	121	107
Credit	151.7	214	205	190
Glass	40.75	36	42	34

Tabella 4.4: Numero di errori di classificazione - confronto con altri concorrenti, da [43]

4.2.2 Iris

Il dataset Iris, costituito da dati raccolti dall'eminente botanico Edgar Anderson, è uno dei più citati nella letteratura, da quando Ronald Aymer Fisher lo utilizzò nel 1936 in un articolo [24] nel quale si proponeva un metodo per classificare un individuo come appartenente ad un certo gruppo sfruttando un *discriminante lineare*, una funzione lineare negli attributi misurati. L'idea era di ottenere i coefficienti che costituiscono l'opportuna combinazione lineare in modo da massimizzare la "separazione" tra gruppi, opportunamente definita. Il dataset Iris dell'UCI, che contiene due campioni modificati rispetto a quello originario di Fisher (qui corretti in base alle indicazioni fornite sempre nel sito dell'UCI) è formato da tre classi: Iris Setosa (50 campioni), Iris Versicolor (50 campioni) e Iris Virginica (49 campioni; uno era ripetuto ed è stato eliminato). Negli esempi riportati si è proceduto a discriminare Iris Setosa dalle altre due e poi Iris Versicolor da Iris Virginica. Il primo caso è molto facile: si può verificare che le due classi sono linearmente separabili. Il secondo si mostra leggermente più ostico, essendo le due classi in parte "sovrapposte".

Per valutare le prestazioni dell'algoritmo e avere termini di paragone, si è dapprima diviso in due parti il dataset, l'una è stata utilizzata per l'addestramento e l'altra per il test. Si sono poi eseguiti alcuni test col programma winSVM [56], molto essenziale, che si appoggia su *mySVM* [55] e prevede un ausilio per la selezione dei parametri (esecuzione con diversi nuclei e parametri di un numero considerevole di addestramenti, con valutazione del relativo scarto quadratico medio tra uscita della SVM e valore dei campioni).

Com'è prevedibile, con Iris Setosa si può ottenere facilmente un tasso di misclassificazione pari a zero, senza nessun bisogno di introdurre nuclei complicati. Anzi, con un nucleo "generico" come quello gaussiano (con $\gamma = 5$

e $C = 1$, configurazione che porta ad errore nullo sull'insieme di addestramento[5]) si ottiene un'accuratezza del $94,7\%$ sull'insieme di test (4 campioni non sono riconosciuti come appartenenti alla specie Setosa). Invertendo i ruoli dell'insieme di test e di addestramento e riaddestrando il classificatore, si ottiene invece ancora un errore nullo. Per quanto concerne Iris Versicolor VS Virginica, con un nucleo lineare (e $C = 1$, che conduce ad una misclassificazione in fase di addestramento) si ottiene un'accuratezza del $91,8\%$ (4 campioni misclassificati). Utilizzando un nucleo gaussiano (con $\gamma = 5$ e $C = 100$) si ottiene un'accuratezza solo del $79,6\%$ (10 campioni misclassificati). Anche qui, scambiando i ruoli dei due insiemi e riaddestrando si ottiene ancora un'accuratezza del $91,8\%$ (4 campioni misclassificati) nel caso con nucleo lineare e del $93,8\%$ (solo 3 campioni misclassificati) nel caso con nucleo gaussiano. Di seguito si riportano i risultati ottenuti da RealGPE, con l'insieme diviso in due parti a ruoli alternati (si riportano le somme dei campioni misclassificati, rigettati e classificati correttamente per i due test). Non essendo i dataset divisibili per 2 la colonna "V.A." contiene un'approssimazione dei valori calcolati applicando la (3.3) con, al denominatore, $\lfloor N/2 \rfloor$ e $\lceil N/2 \rceil$, dove N è la numerosità del dataset. Si ricordi comunque che l'utilizzo della (3.3), qui, non è rigorosamente giustificato alla luce della teoria, essendo violate le ipotesi per la sua validità. Dall'esame dei risultati ottenuti da RealGPE, mostrati di seguito, si può concludere che l'algoritmo si comporta in maniera confrontabile col risultato ottenuto dalla SVM in entrambi i casi.

Iris Setosa

d	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	∼2.7%	5	0 (0.00%)	144
3	∼4%	3	1 (0.67%)	145
4	∼5.4%	0	1 (0.67%)	148

Iris Virginica VS Versicolor

d	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	∼4%	70	3 (3.06%)	26
3	∼6%	66	3 (3.00%)	30
4	∼8%	66	3 (3.00%)	30
5	∼10%	0	4 (4.00%)	95

[5]Si veda la sezione 1.5.6 per un cenno sul significato dei parametri.

4.2.3 Riconoscimento di caratteri alfabetici

In [27] sono pubblicati grafici che confrontano, basandosi sul dataset Letter dell'UCI, la tecnica di SVM con rigetto proposta dagli autori (SVM-reject) con quella più ingenua tradizionalmente impiegata in letteratura (SVM-light). Le visualizzazioni ivi proposte rappresentano le cosiddette *curve A-R* per la valutazione della relazione tra accuratezza e rigetto di un classificatore. Ispirandoci dunque a [27], poniamo il tasso di rigetto in ascissa e l'accuratezza[6] in ordinata, sebbene si possa osservare che, nel contesto teorico a cui si presta l'algoritmo RealGPE, verrebbe più naturale considerare l'accuratezza come variabile indipendente, in quanto probabilisticamente sotto controllo. Per quanto nei due grafici riportati l'accuratezza di SVM-reject e SVM-light cresca più rapidamente al variare del tasso di rigetto, RealGPE non sfigura e anzi è migliore per tassi di rigetto molto bassi. I risultati proposti di seguito sono frutto di una validazione incrociata 2-fold: $\lfloor N/2 \rfloor$ campioni sono stati utilizzati come test per l'algoritmo addestrato sui campioni rimanenti e così per altri $\lfloor N/2 \rfloor$, diversi dai precedenti. I risultati dei due test, sommati, sono nelle tabelle 4.5 e 4.6, mentre nelle figure 4.5 e 4.7 sono tracciate le curve A-R. In ogni tabella, al variare di d, il punto iniziale e i due sottoinsiemi di addestramento/test sono gli stessi. I dati ripresi da [27] (a cui si rimanda per i dettagli ed eventuali, più approfonditi, spunti per confronti) erano stati ottenuti dividendo il dataset in un insieme per l'addestramento e uno per il test, senza inversione di ruoli.

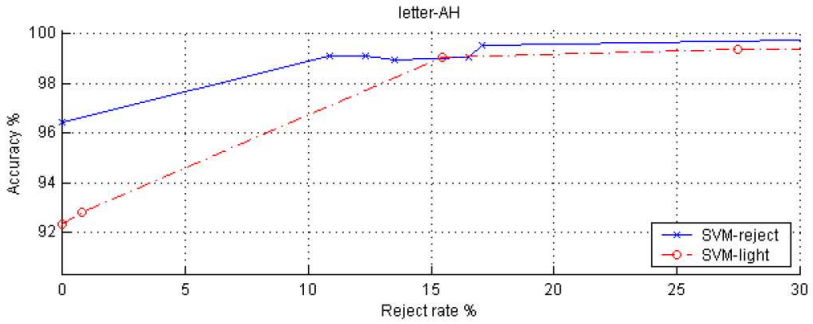


Figura 4.4: Lettere AH, grafico tratto da [27]

[6]Per *accuratezza* si intende:

$$1 - \frac{\text{Numero di errori}}{\text{Totale campioni}}$$

(normalmente espressa in percentuale). Per altri cenni si veda il paragrafo *Come valutare un classificatore*, nella sezione 8.2.

Tabella 4.5: Lettere AH, con validazione incrociata 2-fold

d	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	0.26%	1233	1 (0.07%)	288
3	0.39%	1223	1 (0.07%)	298
5	0.66%	1065	9 (0.59%)	448
10	1.31%	917	18 (1.18%)	587
15	1.97%	816	31 (2.04%)	675
16	2.10%	792	31 (2.04%)	699
17	2.23%	448	28 (1.84%)	1046
18	2.36%	431	28 (1.84%)	1063
20	2.62%	388	33 (2.17%)	1101
25	3.28%	97	50 (3.29%)	1375
30	3.94%	80	70 (4.60%)	1372
35	4.59%	0	55 (3.61%)	1467

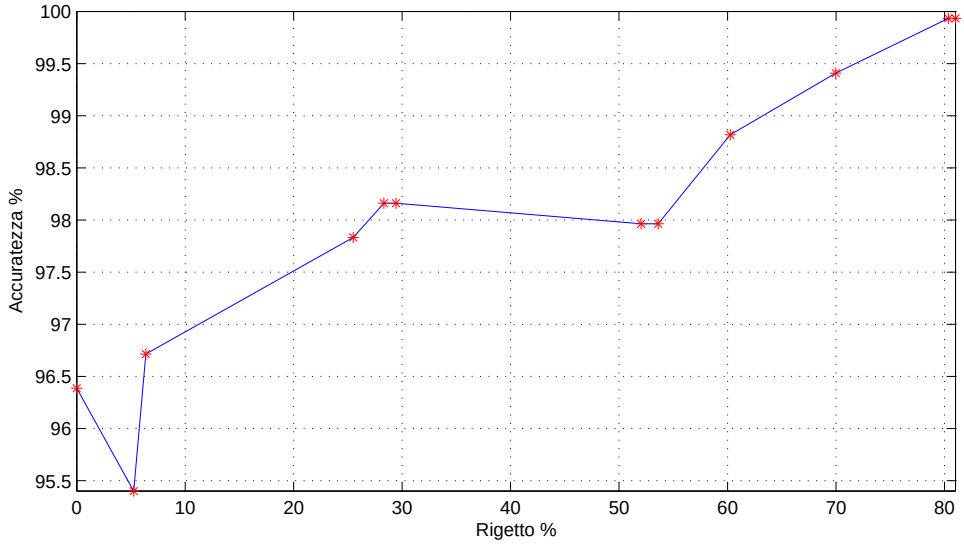


Figura 4.5: Grafico rigetto-accuratezza per Lettere AH, RealGPE

Tabella 4.6: Lettere FT, con validazione incrociata 2-fold

d	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	0.25%	1386	2 (0.13%)	182
3	0.38%	1383	2 (0.13%)	185
5	0.64%	1309	3 (0.19%)	258
10	1.27%	1171	10 (0.64%)	389
15	1.91%	1024	25 (1.59%)	521
16	2.04%	1009	28 (1.78%)	533
17	2.16%	1071	34 (2.17%)	465
18	2.29%	1060	35 (2.23%)	475
20	2.54%	1025	45 (2.87%)	500
25	3.18%	735	43 (2.74%)	792
30	3.82%	408	62 (3.95%)	1100
35	4.45%	251	63 (4.01%)	1256
40	5.09%	145	80 (5.10%)	1345
45	5.73%	92	95 (6.05%)	1383
50	6.36%	37	86 (5.48%)	1447
55	7.00%	0	93 (5.92%)	1477

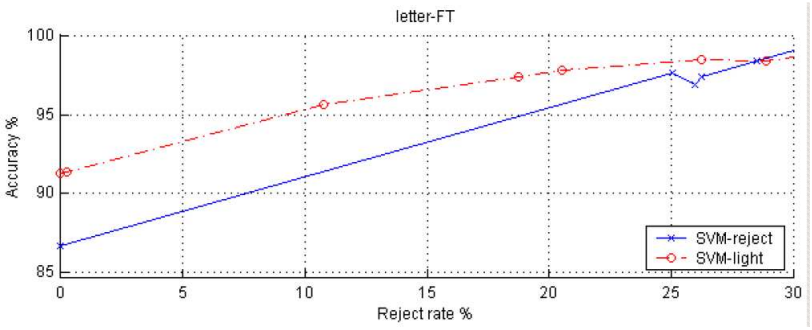


Figura 4.6: Lettere FT, grafico tratto da [27]

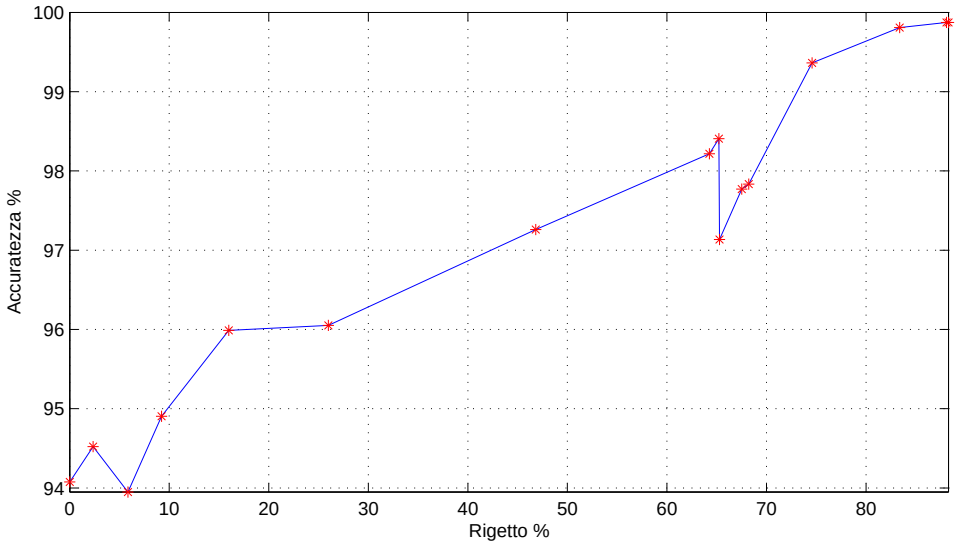


Figura 4.7: Grafico rigetto-accuratezza per Lettere FT, RealGPE

4.2.4 Raccolta di risultati empirici

I risultati seguenti sono tutti esiti di test effettuati col metodo delle validazioni incrociate 10-fold. Per ogni esecuzione con un certo dataset esiste una tabella che porta il nome corrispondente. Il parametro d è un input dell'algoritmo e rappresenta il numero (massimo) di punti ai bordi delle regioni costruite. Il punto di inizializzazione dell'algoritmo viene estratto a caso dall'insieme di addestramento. Il punto di base iniziale e la suddivisione del dataset è la stessa per ogni tabella. *M-min* e *M-max* rispettivamente indicano il minimo e massimo numero di regioni costruite dall'algoritmo nell'insieme delle 10 procedure di test effettuate. *V.A.* indica il bound del valore atteso del tasso di misclassificazioni, calcolato a priori in accordo con la formula (3.3):

$$\frac{d}{N - \lfloor \frac{N}{10} \rfloor}$$

dove N è l'insieme dei campioni nel dataset e k è il numero di fold (qui $k = 10$). Il campo "V.A.", pur se presente in tutte le tabelle, non ha validità teorica laddove i campioni non siano distribuiti secondo una densità ed è stato incluso ovunque solo per mostrare che l'approccio teorico gode in pratica di una certa robustezza a fronte di ipotesi non soddisfatte. È poi presente il conteggio complessivo dei campioni rigettati, misclassificati e classificati correttamente nell'insieme dei 10 test.

Tabella 4.7: **Breast**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
3	3	3	0.49%	267	5 (0.74%)	408
5	5	5	0.81%	265	6 (0.88%)	409
10	10	10	1.63%	252	8 (1.18%)	420
11	8	11	1.79%	200	11 (1.62%)	469
12	9	12	1.95%	198	12 (1.76%)	470
15	10	14	2.44%	128	19 (2.79%)	533
20	8	15	3.25%	76	23 (3.38%)	581
25	11	17	4.07%	29	23 (3.38%)	628
30	15	20	4.88%	8	31 (4.56%)	641
35	11	24	5.69%	2	27 (3.97%)	651
40	10	20	6.50%	0	28 (4.12%)	652

Tabella 4.8: **Haberman**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	0.75%	283	2 (0.69%)	5
3	3	3	1.13%	278	3 (1.03%)	9
4	1	2	1.51%	273	4 (1.38%)	13
5	2	4	1.89%	262	6 (2.07%)	22
8	3	5	3.02%	260	9 (3.10%)	21
9	5	6	3.40%	257	10 (3.45%)	23
10	4	6	3.77%	235	11 (3.79%)	44
15	5	8	5.66%	225	15 (5.17%)	50
20	9	11	7.55%	219	19 (6.55%)	52
30	13	15	11.32%	200	28 (9.66%)	62
40	17	22	15.09%	165	35 (12.07%)	90
50	20	26	18.87%	136	47 (16.21%)	107
60	25	31	22.64%	110	55 (18.97%)	125
70	30	37	26.42%	91	61 (21.03%)	138
80	34	42	30.19%	52	66 (22.76%)	172
90	38	50	33.96%	27	75 (25.86%)	188
100	44	54	37.74%	4	82 (28.28%)	204
106	44	55	40.00%	0	83 (28.62%)	207

Tabella 4.9: **Pima**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	0.29%	756	2 (0.26%)	2
3	3	3	0.43%	754	4 (0.53%)	2
5	5	5	0.72%	752	6 (0.79%)	2
8	8	8	1.16%	739	11 (1.45%)	10
9	3	5	1.30%	729	7 (0.92%)	24
10	4	6	1.45%	724	9 (1.18%)	27
15	6	9	2.17%	719	13 (1.71%)	28
20	8	11	2.89%	704	16 (2.11%)	40
30	9	14	4.34%	657	28 (3.68%)	75
43	12	16	6.21%	628	41 (5.39%)	91
44	12	17	6.36%	626	43 (5.66%)	91
45	13	18	6.50%	611	44 (5.79%)	105
50	13	18	7.23%	601	49 (6.45%)	110
70	20	25	10.12%	508	77 (10.13%)	175
100	28	34	14.45%	402	114 (15.00%)	244
125	33	42	18.06%	328	144 (18.95%)	288
150	41	51	21.68%	244	158 (20.79%)	358
170	50	58	24.57%	201	177 (23.29%)	382
200	53	71	28.90%	92	209 (27.50%)	459
230	72	88	33.24%	22	237 (31.18%)	501
250	73	92	36.13%	0	241 (31.71%)	519

Tabella 4.10: **Bupa**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
3	3	3	0.96%	328	3 (0.88%)	9
6	6	6	1.93%	318	7 (2.06%)	15
7	1	2	2.25%	314	9 (2.65%)	17
8	2	3	2.57%	313	10 (2.94%)	17
10	4	5	3.22%	308	11 (3.24%)	21
15	4	5	4.82%	303	16 (4.71%)	21
26	8	12	8.36%	275	23 (6.76%)	42
27	7	10	8.68%	262	29 (8.53%)	49
28	8	11	9.00%	258	32 (9.41%)	50
29	9	12	9.32%	257	31 (9.12%)	52
30	10	13	9.65%	255	31 (9.12%)	54
50	14	18	16.08%	203	50 (14.71%)	87
70	22	27	22.51%	148	79 (23.24%)	113
90	30	34	28.94%	104	89 (26.18%)	147
100	33	40	32.15%	85	98 (28.82%)	157
120	41	50	38.59%	15	122 (35.88%)	203
130	41	59	41.80%	6	122 (35.88%)	212
135	41	60	43.41%	0	129 (37.94%)	211

Tabella 4.11: **Credit**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	0.34%	647	2 (0.31%)	1
5	5	5	0.85%	617	5 (0.77%)	28
10	10	10	1.70%	568	12 (1.85%)	70
15	15	15	2.55%	551	18 (2.77%)	81
16	7	10	2.72%	569	15 (2.31%)	66
17	8	11	2.89%	568	16 (2.46%)	66
30	15	18	5.10%	482	28 (4.31%)	140
50	18	23	8.50%	371	49 (7.54%)	230
80	27	35	13.61%	228	91 (14.00%)	331
100	32	43	17.01%	147	111 (17.08%)	392
120	33	50	20.41%	64	136 (20.92%)	450
135	33	60	22.96%	15	145 (22.31%)	490
136	33	64	23.13%	12	146 (22.46%)	492
137	33	68	23.30%	9	146 (22.46%)	495
150	37	57	25.51%	2	143 (22.00%)	505
155	37	63	26.36%	0	151 (23.23%)	499

Tabella 4.12: **Glass**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	1.36%	156	2 (1.25%)	2
3	3	3	2.04%	140	3 (1.88%)	17
9	9	9	6.12%	111	12 (7.50%)	37
10	1	5	6.80%	122	14 (8.75%)	24
11	2	6	7.48%	109	15 (9.38%)	36
20	9	12	13.61%	83	21 (13.12%)	56
30	12	19	20.41%	41	28 (17.50%)	91
35	14	20	23.81%	26	33 (20.62%)	101
40	15	26	27.21%	18	33 (20.62%)	109
42	16	23	28.57%	11	33 (20.62%)	116
44	15	25	29.93%	4	36 (22.50%)	120
45	14	25	30.61%	1	41 (25.62%)	118
50	14	27	34.01%	0	40 (25.00%)	120

Tabella 4.13: **Iris Setosa**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
1	1	1	0.74%	47	1 (0.71%)	92
2	2	2	1.48%	0	2 (1.43%)	138

Tabella 4.14: **Iris Versicolor VS Virginica**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	2.22%	58	2 (2.22%)	30
3	3	3	3.33%	55	5 (5.56%)	30
4	4	4	4.44%	48	6 (6.67%)	36
5	3	4	5.56%	38	5 (5.56%)	47
6	4	5	6.67%	32	6 (6.67%)	52
10	4	8	11.11%	0	8 (8.89%)	82

Tabella 4.15: **Lettere AH**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	0.15%	1494	2 (0.13%)	24
3	3	3	0.22%	1469	4 (0.26%)	47
5	5	5	0.36%	1417	6 (0.39%)	97
10	10	10	0.73%	1284	12 (0.79%)	224
15	15	15	1.09%	1189	16 (1.05%)	315
16	16	16	1.17%	1143	16 (1.05%)	361
17	9	12	1.24%	630	19 (1.25%)	871
18	10	13	1.31%	614	19 (1.25%)	887
20	12	15	1.46%	568	23 (1.51%)	929
25	16	17	1.82%	370	24 (1.58%)	1126
30	15	20	2.19%	297	29 (1.91%)	1194
35	16	23	2.55%	185	41 (2.70%)	1294
40	14	21	2.92%	94	43 (2.83%)	1383
45	10	21	3.28%	0	44 (2.89%)	1476

Tabella 4.16: **Lettere FT**

d	M-min	M-max	V.A.	Tot. Rigetti	Tot. Errori	Tot. Corretti
2	2	2	0.14%	1548	2 (0.13%)	20
3	3	3	0.21%	1538	2 (0.13%)	30
5	5	5	0.35%	1527	4 (0.25%)	39
10	10	10	0.71%	1487	10 (0.64%)	73
15	15	15	1.06%	1418	19 (1.21%)	133
16	16	16	1.13%	1403	20 (1.27%)	147
17	13	15	1.20%	1244	17 (1.08%)	309
18	14	16	1.27%	1220	18 (1.15%)	332
20	16	18	1.41%	1197	18 (1.15%)	355
25	14	18	1.77%	1093	37 (2.36%)	440
30	13	19	2.12%	826	33 (2.10%)	711
35	17	21	2.48%	804	39 (2.48%)	727
40	18	21	2.83%	713	34 (2.17%)	823
45	16	23	3.18%	595	40 (2.55%)	935
50	21	25	3.54%	502	43 (2.74%)	1025
55	22	26	3.89%	336	57 (3.63%)	1177
60	20	29	4.24%	150	58 (3.69%)	1362
65	23	29	4.60%	114	66 (4.20%)	1390
70	21	30	4.95%	35	63 (4.01%)	1472
75	17	31	5.30%	10	67 (4.27%)	1493
80	17	31	5.66%	0	65 (4.14%)	1505

Parte II

Derivazioni teoriche

Capitolo 5

Un risultato negativo

> Insegna alla tua lingua a dire
> "non lo so" affinché tu non
> venga sorpreso in falso
>
> Talmud - Derech Eretz Zuta, III

Prima di procedere con la spiegazione dei fondamenti teorici che permettono di fornire garanzie sull'errore di un classificatore binario con opzione di rigetto, è opportuno discutere un risultato (reperibile, con la dimostrazione, in [36]) che sta alla base del presente lavoro, in quanto *limita in maniera fortemente negativa la capacità di prevedere a priori l'errore che sarà commesso da un qualsiasi algoritmo di classificazione binario* e fornisce pertanto una motivazione teorica forte per la costruzione di classificatori con rigetto.

5.1 Il risultato

Dal teorema 7.1 in [36] è immediato arrivare al seguente:

Teorema 1. *Sia $\rho > 0$ un numero piccolo a piacere. Per ogni N e per qualsiasi algoritmo di classificazione binaria senza opzione di rigetto, esiste una misura di probabilità μ e una $y(\cdot)$ tali che*

$$\mu^N(PE(\widehat{y}_N) > \epsilon) \geq 0.5 - \epsilon - \rho \tag{5.1}$$

Per completezza, la dimostrazione è presente nell'appendice A.4.

5.2 Discussione del risultato negativo

Il termine a sinistra della disequazione (5.1) è una descrizione in probabilità dell'errore di generalizzazione $\mathrm{PE}(\widehat{y}_N)$ (si veda la sezione 1.4), che resta determinato una volta deciso un algoritmo di classificazione: $1-\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon)$ è evidentemente la distribuzione di probabilità della variabile casuale $\mathrm{PE}(\widehat{y}_N)$. Se, per un algoritmo, la distribuzione della relative probabilità di errore fosse nota (o minorata da una distribuzione nota) indipendentemente dalla realtà da cui sono estratti i campioni, si potrebbero effetture delle previsioni *a priori* circa le prestazioni dei classificatori prodotti, cioè prima di aver preso visione dell'insieme di addestramento, o dei risultati di computazioni svolte su di esso. Nota la distribuzione dell'errore, si può garantire per esempio che sia minore di un certo $\delta(\varepsilon) > 0$ la probabilità dell'evento che ad un classificatore, risultante dall'addestramento su N campioni estratti secondo μ, sia associato un errore di generalizzazione maggiore di una certa soglia ϵ, cioè

$$\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon) < \delta$$

Perciò, nota la distribuzione, sarebbe possibile caratterizzare completamente (seppure in probabilità) il comportamento dell'algoritmo in termini di prestazioni attese dai classificatori prodotti. Sarebbe pure possibile fare considerazioni coinvolgenti anche un solo livello di probabilità, calcolando il valore atteso di $\mathrm{PE}(\widehat{y}_N)$ sullo spazio dei campioni. Purtroppo, il risultato del teorema 1 annienta la speranza di caratterizzare in termini di distribuzione dell'errore indipendente dalla distribuzione dei campioni un algoritmo o, meglio, di caratterizzarlo in maniera utile. Infatti, il teorema garantisce proprio che, per qualsiasi algoritmo e numero di campioni, è possibile immaginare una distribuzione di campioni e relativa natura in modo che la probabilità di estrarre N campioni che conducano ad un classificatore che misclassifichi con probabilità alta sia sempre considerevole. Fissando un ρ molto piccolo e prossimo a zero, il teorema garantisce infatti che i possibili grafici della probabilità $\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon)$ stiano tutti sopra la zona in grigio presente in figura 5.1. Il risultato è fortemente negativo: non ci si lasci ingannare dal fatto che per $\epsilon > 0.5$ il minorante non vincola la probabilità, poiché l'ultimo valore significativo per ϵ è appunto 0.5, corrispondendo un tale tasso di errore ad un classificatore che decide casualmente quale etichetta assegnare ad un certo campione.

5.2.1 Necessità del rigetto

Il risultato presentato del teorema 1 è fortemente dissuasivo: evidentemente, algoritmi in grado di generare classificatori binari che ammettano, indipen-

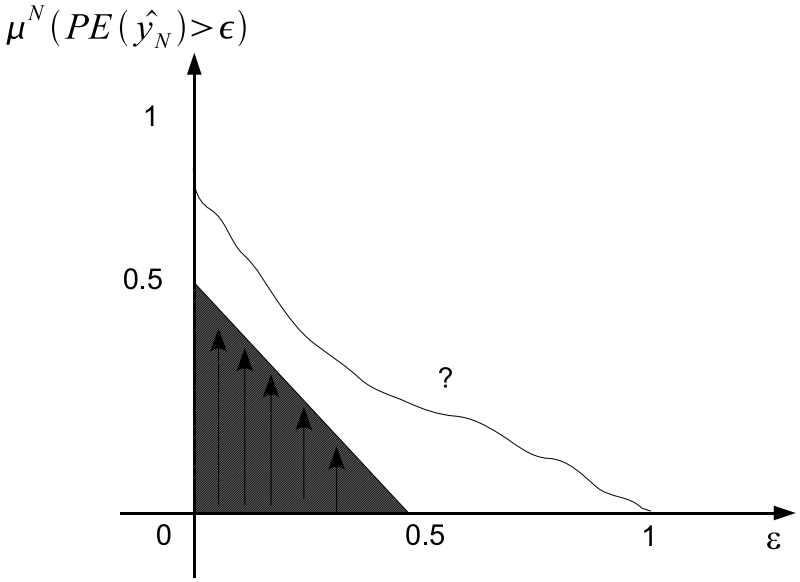


Figura 5.1: Il minorante della disequazione (5.1), per $\rho \sim 0$. Fissato $\rho \sim 0$, per qualsiasi N e qualsiasi algoritmo, esiste una misura di probabilità μ secondo la quale si distribuiscono i campioni tale che il reale valore di $\mu^N(\mathrm{PE}(\widehat{y}_N) > \epsilon)$ stia al di sopra della parte scura.

dentemente da μ e da y, distribuzioni dell'errore "buone", come abbiamo ritenuto essere quelle in figura 3.1 non possono esistere! Se si vuole un classificatore binario e si vuole che sia in grado di garantire buone prestazioni a priori, si vuole troppo. Questo limite ha una portata fortemente intuitiva e non fa altro che riaffermare quanto la tradizione filosofica e il buon senso vanno da sempre suggerendo: l'"esperto" che pretende di sapere tutto e non dice mai "non lo so" finirà con alta probabilità in errore laddove sia chiamato ad esprimere giudizi anche in situazioni complicate, che esulano dal suo campo o che richiederebbero ulteriore indagine[1]. Nella teoria dell'apprendimento,

[1]Per la tradizione filosofica occidentale, il paradigma di tale consapevolezza si trova verosimilmente nell'"Apologia", la celebre opera di Platone, laddove Socrate, in cerca di uomini più sapienti di lui, inizia col raffrontarsi ad un uomo considerato sapiente per concludere: "costui credeva sapere e non sapeva, io invece, come non sapevo, neanche credevo sapere; e mi parve insomma che almeno per una piccola cosa io fossi piú sapiente di lui, per questa che io, quel che non so, neanche credo saperlo". E, ancora, cercando di saggiare la sapienza di alcuni artisti arriva a dire: "per ciò solo che sapevano esercitar bene la loro arte, ognuno di essi presumeva di essere sapientissimo anche in altre cose assai più importanti e difficili; e questo difetto di misura oscurava la loro stessa sapienza"[51].

e della classificazione in particolare, si è già da tempo arrivati a introdurre l'opportunità che una funzione di decisione possa essere parziale, ovvero possa sospendere il giudizio circa l'appartenenza di un nuovo campione ad una certa classe, con l'intento di migliorare l'accuratezza delle previsioni. Si rimanda alla 8.2 per alcuni riferimenti bibliografici.

Concludendo, il fatto di mirare ad un algoritmo che costruisca classificatori con opzione di rigetto non è un capriccio, né una facile scorciatoia, ma deriva dalla *necessità teorica* di porsi al di fuori delle ipotesi del teorema 1; la banale considerazione che un classificatore, se non risponde mai, non sbaglia mai basta a convincere che l'ipotesi dell'assenza di sospensione di giudizio è centrale per la validità del risultato negativo. Naturalmente, questo apre le porte ad alcuni ragionamenti. Ovviamente, non pretendiamo che l'algoritmo garantisca di non sbagliare mai. Abbiamo anzi chiaro l'obbiettivo cui tendere, nella figura (3.1): ciò che si vuole è una garanzia sulla probabilità di errore. Si ammette cioè di poter sbagliare, eppure, garantito in probabilità l'errore, si vorrebbe massimizzare la probabilità dell'insieme di campioni per i quali ciascun classificatore produce una risposta. Nell'ideazione di RealGPE, seppure su un piano euristico, sono state determinanti considerazioni circa questo *requisito di massima copertura*, con l'intento di limitare il ricorso al rigetto al minimo indispensabile.

Capitolo 6

Fondamenti matematici

In questo capitolo si espone il risultato che fornisce la base matematica per le garanzie teoriche offerte da RealGPE e da IdealGPE. Quanto qui presentato, concettualmente, non aggiunge nulla di nuovo ad un risultato che si trova in [10] nell'ambito della programmazione convessa basata su scenario; la formulazione, pur seguendo fedelmente [10], sarà svolta in modo più generale, così da rendere il risultato applicabile al problema della classificazione. L'effettiva applicazione del risultato fondamentale qui delineato agli algoritmi proposti sarà oggetto del capitolo seguente.

6.1 Definizioni e proprietà preliminari

Sia Δ l'insieme dei campioni che, dotato di una σ-algebra, costituisce uno spazio misurabile su cui è definita una misura di probabilità μ.

Volendo poter interpretare una M-upla come un'estrazione indipendente di campioni in Δ, definiamo una multi-estrazione di M punti un qualsiasi elemento dell'insieme Δ^M=$\{(\delta_1, ...\delta_M)$ t.c $\delta_i \in \Delta\}$, sul quale è definita la probabilità prodotto μ^M.

Ciò consente di interpretare una M-upla come estrazione da Δ di campioni tra loro indipendenti. Sia d un numero naturale positivo.

Definizione 2. *Definiamo una funzione C come una funzione che associa ad una qualsiasi M-upla, una d-upla, con $M \geq d$ e alcune proprietà*

a) $\forall\ \boldsymbol{\delta}_M = (\delta_1, ..., \delta_M)$, $M \geq d$, vale $C(\pi(\boldsymbol{\delta}_M)) = C(\boldsymbol{\delta}_M)$, dove π applica una qualsiasi permutazione alla M-upla $\boldsymbol{\delta}_M$. Per comodità, data questa proprietà, le eguaglianze che coinvolgono la funzione C saranno sempre intese a meno di permutazioni.

b) La d-upla $\boldsymbol{\delta}_d$ che C associa ad una certa M-upla $\boldsymbol{\delta}_M$ è sempre formata da una scelta di d componenti di $\boldsymbol{\delta}_M$, eventualmente permutate.

Definizione 3. *Ora, definiamo la funzione E che ad una qualsiasi multi-estrazione associa un sottoinsieme misurabile di Δ:*

$$E : \bigcup_{M \geq d} \Delta^M \rightarrow \mathcal{P}(\Delta)$$

Useremo la notazione $E_{(\delta_1,...,\delta_M)}$ per indicare $E((\delta_1, ..., \delta_M))$. Nel corso della trattazione, con E ci si riferirà tanto alla funzione quanto al generico elemento (sottoinsieme di Δ) ad essa associato. Per qualsiasi M-upla $(\delta_1, ..., \delta_M)$ deve valere la seguente proprietà

c) $\delta_i \notin E_{(\delta_1,...,\delta_M)}, i = 1, ..., M$.

Per qualsiasi coppia di multi-estrazioni $\boldsymbol{\delta'}_{M'}$ e $\boldsymbol{\delta''}_{M''}$, con $M' \geq d$ e $M'' \geq d$, gli insiemi E associati alle multi-estrazioni devono godere della seguente proprietà

d) Se $C(\boldsymbol{\delta'}_{M'}) = C(\boldsymbol{\delta''}_{M''})$, allora $E_{\boldsymbol{\delta'}_{M'}} = E_{\boldsymbol{\delta''}_{M''}}$

In altri termini, si può definire su $\bigcup_{M \geq d} \Delta^M$ la relazione $R(a, b)$:="a e b sono multi-estrazioni a cui C associa la medesima d-upla". Tale relazione è di equivalenza e pertanto induce una partizione su $\bigcup_{M \geq d} \Delta^M$. Tutti gli elementi di $\bigcup_{M \geq d} \Delta^M$ che appartengono alla medesima classe di equivalenza sono associati ad un unico insieme E.

Definizione 4. *Definiamo pertanto la funzione V_d*

$$V_d : (\delta_1, ..., \delta_d) \in \Delta^d \rightarrow \mu(E_{(\delta_1,...\delta_d)})$$

V_d è una variabile casuale, essendo su Δ^d definito uno spazio di probabilità.

In quanto tale, ha associata una distribuzione di probabilità

$$F_{V_d}(\alpha) = \mu^d(\{(\delta_1, ...\delta_d) \in \Delta^d \text{ t.c. } V_d(\delta_1, ..., \delta_d) \leq \alpha\}). \tag{6.1}$$

Per la proprietà "**a**", la funzione C può essere vista come una funzione che *seleziona* alcuni punti di una M-upla di lunghezza maggiore o uguale a d. I punti selezionati da C hanno un ruolo speciale nel contesto che stiamo descrivendo, in quanto da essi è possibile determinare l'insieme E associato all'intera M-upla. Tuttavia, per quanto detto finora, E potrebbe essere anche una funzione che assegna a qualsiasi M-upla, per esempio, l'insieme vuoto: in tal caso, i d punti di una M-upla individuati da C non sarebbero così rilevanti circa la conoscenza degli insiemi E associati alle M-uple. Definiamo ora alcuni punti di ciascuna M-upla che portano davvero informazioni su E, in quanto la loro presenza all'interno della M-upla *fa la differenza*.

Definizione 5. *Un punto δ_j $(1 \leq j \leq M)$ è detto* ***punto di supporto*** *per una certa M-upla $(\delta_1, ..., \delta_M)$ se $\delta_j \in E_{(\delta_1,...,\delta_{j-1},\delta_{j+1},...,\delta_M)}$.*

Presto si mostrerà che se C selezionasse tutti e soli i punti di supporto, allora avrebbe un ruolo fondamentale circa la caratterizzazione di E rispetto ad una multi-estrazione di qualsiasi dimensione. I risultati che seguono si basano proprio sull'interpretazione di C come di una funzione che seleziona tutti e soli i punti di supporto di una certa M-upla. In caso di elementi ripetuti all'interno di una M-upla, però, sussiste una difficoltà, in quanto uno stesso punto potrebbe essere di supporto o non esserlo. Sia, per esempio, $\boldsymbol{\delta}_{d+1} = (\delta_1, ..., \delta_d, \delta_{d+1})$ composta di elementi tutti uguali a $\widehat{\delta}$. Eliminando un elemento dalla $\boldsymbol{\delta}_{d+1}$ si ottiene $\boldsymbol{\delta}_d$. $\widehat{\delta}$ sarà quindi necessariamente selezionato da C. Se fosse di supporto, si avrebbe $\widehat{\delta} \in E_{\boldsymbol{\delta}_d}$ ma, per la proprietà "**c**", anche $\widehat{\delta} \notin E_{\boldsymbol{\delta}_d}$. La presenza di elementi ripetuti in una M-upla contrasta quindi l'interpretazione di C come di *selettore dei punti di supporto* e occorre assumere che C selezioni tutti e soli i punti di supporto di una certa M-upla, *a patto che tale M-upla non contenga punti ripetuti*[1]. Escludendo allora M-uple con ripetizioni, vale la seguente importante proprietà (ciò è dimostrato nell'appendice A.3):

e) Se, per una data M-upla $\boldsymbol{\delta}_M = (\delta_1, ..., \delta_M)$, un qualsiasi $\widehat{\delta} \neq \delta_j$ (con $j = 1, ..., M$) non appartiene a $E_{\boldsymbol{\delta}_M}$, allora $C(\delta_1, ..., \delta_M) = C(\delta_1, ..., \delta_M, \widehat{\delta})$.

Dunque, un generico elemento non appartenente ad $E_{\boldsymbol{\delta}_M}$, se aggiunto alla M-upla, darebbe luogo ad una $M+1$-upla avente ancora associato lo stesso insieme E che è associato alla M-upla.

Per maggiore generalità, assumiamo che C possa fallire nell'individuare tutti e soli i punti di supporto di una M-upla anche in altri casi oltre a quelli finora esclusi, purché l'insieme di tali casi sia di probabilità nulla. Così, d'ora in poi considereremo C *una funzione che seleziona* quasi certamente[2] *tutti e soli i punti di supporto di una M-upla estratta da Δ^M.*

Gli elementi introdotti finora sono sufficienti per caratterizzare completamente in probabilità la variabile casuale V_M, come mostra il seguente risultato fondamentale.

[1] In alternativa a questo distinguo, C potrebbe essere trattata più semplicemente come una funzione che opera su insiemi di M $(M \geq d)$ elementi, anziché su M-uple. Si continua a trattarla come una funzione che opera su ennuple per rendere più agevole la dimostrazione del risultato principale.

[2] Si farà uso della locuzione "quasi certamente" in senso matematico: una proprietà vale *quasi certamente* se *vale con probabilità 1.*

6.2 Il teorema fondamentale

Teorema 2. *Sia M un numero fissato e $M \geq d$. Sia $Z \subseteq \Delta^M$, $\mu(Z) = 0$, l'insieme delle M-uple per le quali C non seleziona tutti e soli i punti di supporto. La distribuzione della variabile casuale*

$$V_M : (\delta_1, ..., \delta_M) \rightarrow \mu(E_{(\delta_1,...,\delta_M)})$$

resta, allora, esattamente determinata ed è:

$$F_{V_M}(\epsilon) = 1 - \sum_{i=0}^{d-1} \binom{M}{i} \epsilon^i (1-\epsilon)^{M-i} \tag{6.2}$$

La dimostrazione, che ha per presupposto quanto finora detto nel presente capitolo, per non appesantire la lettura è riportata nell'appendice A.1, laddove si possono peraltro studiare i passaggi che conducono alla formulazione integrale:

$$F_{V_M}(\epsilon) = \binom{M}{d} \int_0^{\epsilon} (1-\alpha)^{M-d} \alpha^d \mathrm{d}\alpha \tag{6.3}$$

Nota la distribuzione F_{V_M}, può essere facilmente calcolato il valore atteso di V_M (la dimostrazione è pure nell'appendice, A.2):

$$\mathbf{E}[V_M] = \frac{d}{M+1}$$

Questo risultato non sorprende. Infatti, era anch'esso noto nel contesto della programmazione convessa basata su scenario (si veda [9]), prima ancora che si pervenisse a ricavare con esattezza la distribuzione di V_M. Nella sezione 8.1 si farà inoltre cenno ad un risultato che presenta delle somiglianze profonde, noto nell'ambito delle SVM, basato sull'applicazione di un risultato di Luntz e Brailovsky [41], riguardante le proprietà statistiche dell'errore di *leave-one-out*.

6.3 Applicazioni

I risultati precedentemente esposti sono dati in una forma abbastanza astratta, in modo da fornire una cornice concettuale applicabile in un ambito diverso da quello di origine.

6.3.1 Ottimizzazione convessa

L'apparato concettuale che porta all'importante risultato della sezione 6.2 è nato nel campo dell'ottimizzazione convessa basata su scenario. In quel contesto,

- lo spazio Δ viene interpretato come lo spazio degli infiniti vincoli convessi da soddisfare e che, campionato un numero finito di volte, dà origine ad una multi-estrazione;
- l'insieme E associato ad una multi-estrazione rappresenta l'insieme dei vincoli in Δ che violano la soluzione ottenuta risolvendo il problema con i vincoli nella multi-estrazione;
- V_M è la probabilità di violazione, cioè la probabilità che un certo vincolo estratto a caso da Δ violi la soluzione del problema con gli M vincoli della multi-estrazione;
- C può essere interpretata come una funzione che rende conto del fatto che, per una classe di problemi, detti *fully supported*, il numero di *vincoli di supporto* per ogni problema basato su un numero finito di vincoli campionati è sempre lo stesso.

6.3.2 Classificazione

Nel contesto di questa tesi, il risultato è invece applicato alla classificazione e si danno le seguenti corrispondenze.

- Δ è X, lo spazio degli oggetti, sul quale si assume definita la misura di probabilità μ. Ad ogni oggetto x può essere associata (deterministicamente) la relativa etichetta $y(x)$. Anche se la funzione y è ignota, si assume che, quando si estraggono punti per formare l'insieme di addestramento, un supervisore sia in grado di fornire le relative etichette.
- La funzione E è quella che individua, per ogni M-upla di campioni per la quale l'algoritmo costruisce una certa funzione di decisione $\widehat{y}_M$, l'insieme di misclassificazione $\{x \in X \text{ t.c. } y(x) \neq \mathsf{R} \text{ e } y(x) \neq \widehat{y}_M(x)\}$.
- La variabile casuale V_M può essere usata come sinonimo di $\mathrm{PE}(\widehat{y}_M)$, per comodità di scrittura e, naturalmente, totale identità di ruolo, date le precedenti corrispondenze.

- C è una funzione (procedura) che permette di estrarre da una M-upla i d punti attraverso i quali è possibile ricostruire $\widehat{y}_M$. Questi d punti devono inoltre essere tutti e soli quelli *di supporto* per quella M-upla nel senso che lo stesso algoritmo addestrato con una $M-1$-upla da cui sia stato rimosso uno di tali punti deve condurre ad un classificatore per il quale il punto rimosso appartiene all'insieme di misclassificazione.

Capitolo 7

Garanzie sull'errore per RealGPE

Si analizzeranno ora le proprietà di IdealGPE e RealGPE alla luce della teoria astratta presentata nel capitolo 6. Si comincerà col dire quali sono le proprietà che un algoritmo deve avere affinché sia valido il risultato fondamentale nella sezione 6.2 applicato al suo errore di generalizzazione. Si mostrerà come superare un'apparente discrepanza tra questi requisiti e l'effettivo funzionamento dei due algoritmi proposti, che saranno analizzati nei loro aspetti costruttivi decisivi. Il capitolo si conclude, nella sezione 7.2, con alcuni possibili spunti per la costruzione di varianti degli algoritmi proposti.

7.1 Requisiti matematici

Alla luce di quanto emerso finora nella trattazione, affinché sia possibile applicare il risultato della sezione 6.2 ad un algoritmo di classificazione, devono essere rispettate alcune condizioni sulle sue caratteristiche. In particolare, date le corrispondenze descritte nella sezione 6.3.2, si vorrebbe che l'algoritmo di classificazione soddisfacesse *quasi certamente* i seguenti requisiti

- L'algoritmo è deterministico e simmetrico, nel senso che il classificatore prodotto è invariante a fronte di permutazioni nella sequenza di addestramento (che quindi può a buon titolo essere definita *insieme* di addestramento).
- L'algoritmo ammette in ingresso un numero d, oltre all'insieme di addestramento di $M \geq d$ campioni, e garantisce la presenza nella M-upla di esattamente d punti di supporto, cioè tali che, se dalla M-upla è rimosso uno dei d punti, che chiamiamo $\widehat{x}$, l'algoritmo (con parametro

d) produce un classificatore che misclassifica $\widehat{x}$. I punti su cui un classificatore generato a partire da M campioni con parametro d commette un errore devono essere gli stessi su cui commette un errore il classificatore prodotto utilizzando per l'addestramento solo i d punti individuati dall'algoritmo.

- I classificatori prodotti non misclassificano i campioni dell'insieme di addestramento.
- I classificatori prodotti ammettono l'opzione di rigetto.

La probabilità di errore per un algoritmo con rigetto che soddisfa i summenzionati requisiti è distribuita secondo la (6.2).

7.1.1 Proprietà di IdealGPE

Una difficoltà preliminare

IdealGPE, così com'è definito nel capitolo 3 *non soddisfa* i requisiti appena elencati e per accorgersene basta una semplice osservazione: l'algoritmo non è simmetrico, in quanto utilizza sempre il primo punto dell'insieme di addestramento come punto di base iniziale e, pertanto, una permutazione della N-upla in ingresso conduce ad risultato diverso. Ciò non deve scoraggiare, perché l'ostacolo è soltanto apparente e facilmente aggirabile: alla fine del cammino, l'unica segno della difficoltà appena rilevata sarà la differenza che il lettore avrà già notato confrontando tra loro la (3.2) a pagina 31 e la (6.2) a pagina 67, e cioè che mentre nella seconda il termina M compare in entrambi i membri dell'uguaglianza, nella prima si ha che il suo omologo N compare a sinistra, ma a destra è sostituito da $N - 1$.

Soluzione

In questo capitolo, si considererà una modalità di utilizzo di IdealGPE (definito come nel capitolo 3) che conduce ad un tipo leggermente diverso di algoritmo. Si assumerà che un operatore *estragga preventivamente, casualmente e indipendentemente da ogni altro campione* un esempio (x_0, y_0) e utilizzi sempre quell'esempio come *primo campione* all'interno della multi-estrazione: esso diventa così a tutti gli effetti un parametro "saldato" all'algoritmo e ha il ruolo di punto di base iniziale. In tal modo, gli ingressi variabili, e quindi propriamente tali, dell'algoritmo così ottenuto (che chiamiamo **algoritmo ridotto basato su IdealGPE**) diventano

- Un parametro d

- Un insieme di addestramento di $N-1$ campioni $((x_1, y_1), ..., (x_{N-1}, y_{N-1}))$

A questo punto, la logica seguita da IdealGPE permette di affermare che tutti i requisiti presentati all'inizio della sezione 7.1 sono soddisfatti nell'ipotesi di campioni distribuiti secondo una densità (e ponendo la corrispondenza $N - 1 = M$). Per il resto di questa sezione, con *algoritmo* intenderemo *algoritmo ridotto basato su IdealGPE*, finché non altrimenti precisato. Passiamo dunque ad esaminare le sue proprietà, alla luce dei requisiti attesi.

Innanzitutto, si può osservare che nulla nell'algoritmo ridotto dipende più dall'ordine dei campioni in ingresso e che quindi esso soddisfa il requisito di **simmetria**.

Logica di funzionamento e selezione dei punti di supporto L'algoritmo inizia col costruire una regione intorno al primo punto di base, che è fissato, e che determina la natura dei punti interni alla regione stessa. Il punto di base successivo, attorno al quale viene costruita la nuova regione, viene scelto tra quelli che giacciono sul bordo della regione precedente. Com'è noto da [9], il numero di *vincoli di supporto* per un problema convesso è al più pari al numero di variabili decisionali. I vincoli di supporto non possono che essere un sottoinsieme dei vincoli attivi che coinvolgono punti dell'insieme di addestramento (d'ora in poi, quando si parlerà di "vincoli", ci si riferirà sempre a questo tipo di vincoli, escludendo quelli "strutturali", come per esempio $A \succeq 0$ e $k \geq 0$). Nel caso in cui i punti si distribuiscano secondo una densità, cosicché non vi siano insiemi con misura di Lebesgue nulla a probabilità non nulla, la probabilità che il numero di vincoli attivi sia maggiore del numero di punti di supporto è nulla. $P_{\frac{n(n+1)}{2}}$, P_{n+1}, P_{CMax} e $P_{\mathrm{CMax}'}$ sono pertanto problemi convessi nei quali i vincoli attivi coincidono *quasi certamente* con quelli di supporto.

I vari x_b, con l'eccezione del primo che è x_0 ed è *sui generis*, corrispondono a vincoli attivi per i problemi convessi risolti nel corso dell'esecuzione e sono quindi *punti di supporto*. Rimosso infatti uno di essi dall'insieme di apprendimento e rieseguito l'algoritmo, la regione con lo stesso punto di base di quella che aveva quel punto sul bordo si modificherebbe arrivando a coprire quello stesso punto, classificandolo in maniera opposta alla sua natura. Per quanto riguarda i punti in S_{esteso} vale un ragionamento analogo: essi sono tutti tranne uno, che è sul bordo, interni alla minima superficie sferica che racchiude i punti in $\mathcal{M}$. Rimossi i punti interni, essi, a una nuova esecuzione dell'algoritmo, ricadrebbero ovviamente nella regione di misclassificazione estesa; rimosso quello sul bordo, esso verrebbe racchiuso entro un'ipersfera più grande, venendo meno la sua funzione di "argine". I punti di supporto sono quindi quelli in $S \cup S_{\mathrm{esteso}}$ e sono d.

Terminazione L'algoritmo termina sempre. Quando in L esistono punti di natura opposta a quella del punto di base x_b, qualunque soluzione di P_{CMax}, $P_{\text{CMax}'}$, $P_{\frac{n(n+1)}{2}+n}$ o P_{d+1} avrà almeno un vincolo attivo. Nel caso di P_{CMax} o $P_{\text{CMax}'}$ ciò è banale. Per $P_{\frac{n(n+1)}{2}+d}$ o P_{n+1} ciò è conseguenza dell'aver definito una regola[1] (*tie-break rule*) che, in presenza di soluzioni multiple, richiede la minimizzazione della norma di B. Pertanto, l'unica soluzione che non comporta la presenza di vincoli attivi è quella banale $A^* = 0, B^* = 0$, che si ottiene quando in L non ci sono punti di natura diversa da quella del punto di base. Ne viene che, se l'algoritmo raggiunge la situazione per cui $S_c = \emptyset$, allora sicuramente avrà classificato tutto. Altrimenti, deve esistere un vincolo attivo e quindi un punto che entra in S_c e fa sì che l'esecuzione dell'algoritmo avanzi. Presto o tardi, si verificherà una delle seguenti condizioni di terminazione

A) $|S|$ raggiungerà il valore d

B) ad S_c verrà assegnato l'insieme vuoto

Per rendersi conto di questo, basta notare che l'insieme L contiene un numero finito di punti: ad ogni ciclo, al passo 4, viene eliminato sempre almeno un punto (x_b): è quindi impossibile che uno stesso punto venga scelto più volte come punto di base e ciò rende impossibile l'instaurarsi di un ciclo infinito. Se $S_c = \emptyset$ ma $|S| < d$, allora entra in gioco la procedura EXTMISC che pone $|S| - d$ punti in S_{esteso}. Poiché $N - 1 \geq d$, è *quasi certamente* possibile trovare un'ipersfera al cui interno vi siano esattamente $d - |S| - 1$ punti e uno sul bordo, presi tra quelli all'interno della $N - 1$-upla iniziale che non siano già entrati in S: trovare tale ipersfera è ciò che essenzialmente consente di fare la risoluzione di $P_{\text{CMax}'}$. Si noti che si sono, ancora una volta, esclusi insiemi di punti a misura nulla (non devono esservi campioni su una stessa superficie sferica), in caso contrario si potrebbe avere $|S|_{\text{esteso}} > d$: l'algoritmo terminerebbe comunque (con un avvertimento).

[1] Convincersi di questo fatto è semplice. Consideriamo una soluzione $\tilde{A}, \tilde{B}$ per la quale la traccia di $\tilde{A}$ è minima. Se $\tilde{A} = 0$, $\tilde{B}$ non può essere il vettor nullo, altrimenti qualsiasi vincolo coinvolgente punti della multi-estrazione (almeno uno esiste per ipotesi) sarebbe violato. Ma se non esistono vincoli attivi, allora si può modificare di "poco" e con continuità il vettore $\tilde{B}$ al fine di minimizzare ulteriormente la sua norma. Ad un certo punto, tale procedura porterà ad avere un vincolo attivo, poiché $\tilde{B}$ non può annullarsi. Se invece $\tilde{A}$ non è nulla e non esistono vincoli attivi, di nuovo si può intervenire con con continuità su $\tilde{B}$. In questo caso, $\tilde{B}$ potrebbe infine annullarsi. A questo punto, se ancora non ci fossero vincoli attivi, significherebbe che, modificando in modo sufficientemente piccolo gli elementi di $\tilde{A}$ si potrebbe ottenere una traccia minore di quella di partenza, in contraddizione con l'ipotesi.

Determinismo È importante essere certi che ogni soluzione A, B trovata nell'esecuzione sia unica. Una regola utilizzata per individuare un'unica soluzione in presenza di molteplici possibilità è detta *tie-break rule*. In $P_{\frac{n(n+1)}{2}+n}$ e P_{n+1}, utilizzando la minimizzazione del quadrato della norma euclidea di B come tie-break rule, si ha che, se la matrice A è nulla, la soluzione individuata minimizzando $||B||$ è unica: infatti i vincoli costituiti dalle diseguaglianze si riducono a vincoli lineari, che individuano come spazio di ammissibilità un politopo: minimizzare $||B||$ equivale a minimizzare il quadrato di $||B||$ che è una cifra di merito *strettamente* convessa e quindi ammette, su ogni (sotto)insieme convesso, un unico minimo[2]. Tuttavia, la minimizzazione di $||B||$

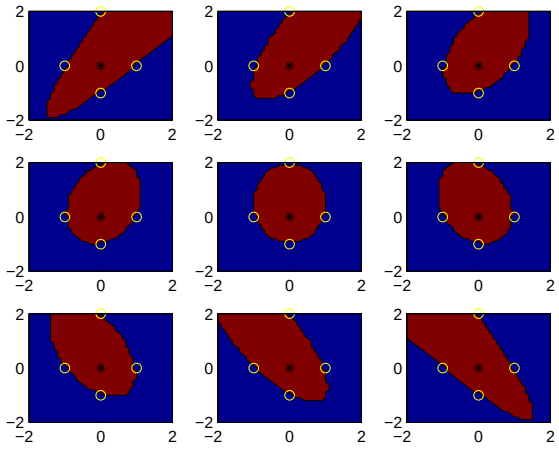


Figura 7.1: Alcune delle infinite possibili soluzioni (regione rossa) ottenute dalla soluzione di $P_{\frac{n(n+1)}{2}+n}$ a parità di $||B||$, in $X = \mathbb{R}^2$. I quattro vincoli sono rappresentati dai cerchietti.

potrebbe non bastare a individuare sempre un'unica soluzione. La figura 7.1 è un esempio che si riferisce alla risoluzione di $P_{\frac{n(n+1)}{2}+n}$ e rappresenta alcune delle infinite possibili regioni risultanti dalla presenza dei quattro vincoli evidenziati con i cerchietti (l'asterisco scuro rappresenta x_b). Per garantire l'unicità della soluzione è necessaria un'altra tie-break rule. Mentre quella su B presente nell'algoritmo è caratterizzante, in quanto fa sì che le regioni generate tocchino i punti di supporto, quest'altra può essere in una certa misura lasciata all'arbitrio del lettore, purché sia convessa. Più in dettaglio, essa può consistere anche di più regole da applicare in cascata e, infatti, nel-

[2]Si veda per esempio [17], pagine 185-186.

l'algoritmo della sezione 3.3 è suggerita la seguente:

☞ Chiamiamo t_0 la funzione che da una soluzione seleziona la norma L2 di B ($t_2(A,B) = ||B||$); t_1 quella che seleziona $a_{1,1}$ ($t_1(A,B) = a_{1,1}$); t_2 quella che seleziona $a_{1,2}$ e così via fino a $t_{\frac{n(n+1)}{2}}$.

- Si individuano le soluzioni che minimizzano t_0.
- SE ce n'è soltanto una, la si sceglie come soluzione
- ALTRIMENTI, a parità di t_0, si individuano le soluzioni che minimizzano t_1.
  - SE ce n'è soltanto una, la si sceglie come soluzione
  - ALTRIMENTI, a parità di t_1, si individuano le soluzioni che minimizzano t_1,... (e così via, fino eventualmente a $t_{\frac{n(n+1)}{2}}$)

☜

Ricostruzione del classificatore a partire da d punti di supporto La necessità di una tie-break rule basata su ottimizzazione convessa, di cui al paragrafo precedente, deriva dal voler garantire che l'algoritmo eseguito con i d punti di supporto produca lo stesso classificatore ottenuto con gli $N-1$ campioni in ingresso. Ripercorriamo brevemente il funzionamento dell'algoritmo: il primo punto si assume sempre fissato, si costruirà quindi la regione "centrata" su di esso, ottenendola mediante la risoluzione di un problema convesso. Poiché tutti i vincoli attivi ottenuti con l'esecuzione dell'algoritmo su $N-1$ campioni sono a disposizione, la soluzione, se con tutti i vincoli era unica, non può che essere la stessa una volta rimossi vincoli non attivi. Se la soluzione invece non era unica, è importante sapere che quella eletta è stata individuata sulla base di un criterio che garantisca l'esatta ricostruibilità, una volta rimossi vincoli non attivi. Una regola come quella descritta subito prima di questo paragrafo soddisfa questo requisito. Infatti, supponiamo per assurdo che l'eliminazione di uno o più vincoli non associati a punti di supporto conduca ad una soluzione diversa. Tale soluzione (la chiamiamo $\mathcal{S}_d$, ed $\mathcal{S}_{N-1}$ quella ottenuta dalla $N-1$-upla), poiché è scelta, dev'essere tale che $t_0(\mathcal{S}_d) < t_0(\mathcal{S}_{N-1})$ oppure $t_0(\mathcal{S}_d) = t_0(\mathcal{S}_{N-1})$. Nel primo caso, per la convessità, deve esistere una soluzione $\mathcal{S}_\lambda = \lambda t_0(\mathcal{S}_d) + (1-\lambda) t_0(\mathcal{S}_{N-1})$, con

$0 \leq \lambda \leq 1$. Poiché nessuno dei vincoli rimossi era attivo, per λ sufficientemente piccolo $\mathcal{S}_\lambda$ deve rientrare nell'insieme di ammissibilità anche per il problema con $N-1$ punti, ma ciò è assurdo, in quanto allora $\mathcal{S}_\lambda$ avrebbe dovuto essere preferita a $\mathcal{S}_{N-1}$. Se $t_0(\mathcal{S}_d) = t_0(\mathcal{S}_{N-1})$, il ragionamento si ripete per t_1 e così via.

La soluzione del primo problema con punto di base x_0 trovata dall'algoritmo eseguito a partire dai d punti dev'essere dunque la stessa prodotta dall'esecuzione a partire da $N-1$ punti. Costruita la prima regione, si eliminano i punti interni ad essa. I punti sul bordo della regione sono gli stessi in entrambe le esecuzioni e ciò consente di fissare allo stesso modo il successivo x_b. Il ragionamento fatto fin qui si ripete invariato per la regione costruita su x_b. E così via fino al momento in cui, in entrambe le esecuzioni, si esegue EXTMISC. Di nuovo, per ricostruire ogni circonferenza costruita attorno ad x_0, è sufficiente la presenza del punto che ne limita l'espansione, che sicuramente è tra i d punti selezionati. Le due esecuzioni conducono quindi alla stessa soluzione.

Nessuna misclassificazione nell'insieme di addestramento Nel corpo dell'algoritmo i punti di supporto individuati e posti in S sono sempre ai bordi di una regione di natura a loro opposta: esaminando la struttura dei problemi risolti e tenendo conto di come viene definita la funzione di decisione risultante è facile convincersi che nessun punto nell'insieme di addestramento può essere misclassificato dalle regioni generate. Nella fase di costruzione di $\mathcal{M}$, pure si osservi che i punti di supporto già posti, ad uno stadio precedente, in S o in S_{esteso} vengono esclusi dall'area di misclassificazione estesa.

"Prudenza" nella crescita di $|\mathbf{S}|$ Come si osserva ad un'analisi degli algoritmi RealGPE (pag. 19) e IdealGPE (pag. 35), non si prende mai in considerazione l'ipotesi che l'algoritmo torni sui propri passi per il verificarsi della condizione $|S| > d$. Infatti, l'algoritmo cerca di costruire regioni con k gradi di libertà, risolvendo problemi che conducono ad al massimo k potenziali nuovi punti di supporto (cioè vincoli attivi), solo quando ha $|S| + k \leq d$ (k potrà essere $\frac{n(n+1)}{2} + n$ oppure $(n+1)$). In altri termini: non prova "ottimisticamente" a costruire regioni complesse contando sul fatto che poi i punti di supporto introdotti possano non essere più di $d - |S|$: se c'è la possibilità di eccedere d, l'algoritmo scala il problema accontentandosi di costruire una regione meno complessa. Questo non è un pavido o comodo eccesso di prudenza, ma ha una ragione ben precisa che risiede nella volontà di poter ricostruire la funzione di decisione $\widehat{y}_{N-1}$ a partire solo dai d punti individuati. Infatti, supponiamo (per comodità) che già al primo passo dell'algoritmo

(con $|S| = 0$) $|S| + k > d$, dove per k s'intende il numero di punti di supporto che si possono ottenere costruendo una regione attraverso P_k. Si immagini che, risolto P_k, si ottengano realmente k punti di supporto. A questo punto, occorrerebbe tornare sui propri passi e risolvere un problema P_h, con $h < k$: supponiamo che questa volta si raggiunga esattamente la condizione $|S| = d$. La domanda è: sono sufficienti i d punti ottenuti dalla soluzione di P_h affinché l'algoritmo pervenga a ricostruire lo stesso classificatore ottenuto a partire da $N - 1$ punti? Evidentemente, l'algoritmo con l'insieme di addestramento ridotto a d punti inizierà tentando di costruire una regione attraverso la soluzione di P_k e la regione così costruita non potrà avere sul bordo più di d punti, quindi verrà "accettata", pur non essendovi nessun motivo per credere che essa coincida con quella costruita da P_h: il classificatore prodotto a partire da d punti potrà differire da quello ottenuto con $N - 1$ punti. In termini più intuitivi, *nel passaggio da $N - 1$ a d campioni sono stati scartati $k - d$ punti che portavano l'informazione circa il fatto che l'algoritmo prendesse una strada anziché un'altra* (risolvesse cioè P_h anziché P_k).

Punto iniziale L'applicazione della teoria presentata nella sezione 6.1 ad IdealGPE può dunque avvenire solo condizionatamente alla conoscenza preventiva di un campione che serve come primo x_b. Si è visto che un qualsiasi algoritmo ridotto basato su RealGPE, con un insieme di addestramento di $N - 1$ campioni, soddisfa i requisiti all'inizio della sezione 7.1 e aderisce alle corrispondenze presentate nella sezione 6.3.2 a pagina 68. Per qualsiasi algoritmo ridotto basato su IdealGPE con un insieme di addestramento di $N - 1$ campioni vale allora, per il suo errore di generalizzazione V_{N-1}, il risultato della sezione 6.2 (con $M = N - 1$) e quindi:

$$\mu^{N-1}(V_{N-1} > \varepsilon) = \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1}$$

Ora indichiamo con $\widetilde{V}_N$, per distinguerlo, l'errore di generalizzazione di IdealGPE con un insieme di addestramento di N campioni, per il quale, lo ripetiamo, il risultato della sezione 6.2 *non è* direttamente applicabile.

Giacché gli N campioni sono estratti casualmente e indipendentemente, una qualsiasi esecuzione di IdealGPE corrisponde all'esecuzione di uno tra i $|X|$ possibili algoritmi ridotti basati su IdealGPE, scelto casualmente secondo μ e indipendentemente dai restanti $N - 1$ campioni. Condizionando il risultato ad una certa estrazione del primo campione si ottiene il risultato che si

è visto valere per un qualsiasi algoritmo ridotto:

$$\mu^N(\widetilde{V}_N > \varepsilon | x_0) = \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1}$$

ed è un risultato indipendente dalla scelta di x_0. Di conseguenza, integrando su X si ottiene ancora:

$$\mu^N(\widetilde{V}_N > \varepsilon) = \sum_{i=0}^{d-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-i-1}$$

che è il risultato presentato nel capitolo 3.

7.1.2 Proprietà di RealGPE

Per RealGPE, si può procedere del tutto analogamente a quanto fatto per IdealGPE e notare che, in questo caso, l'esatta conoscenza a priori della distribuzione dell'errore non è garantita in quanto non è soddisfatto il requisito secondo cui debbano essere individuabili *quasi certamente* d punti di supporto. L'algoritmo può in effetti terminare prima che $|S|$ raggiunga d anche per classi di multi-estrazioni di probabilità non nulla. Questo avviene quando le regioni generate arrivano a coprire l'intero spazio $\mathbb{R}^n$: anziché eseguire EXTMISC come farebbe IdealGPE, RealGPE termina. Siccome i due algoritmi fino al verificarsi della situazione in cui $S_c = \emptyset$ e $|S| < d$ si comportano in maniera identica e EXTMISC comporta solo l'introduzione di un'area di misclassificazione estesa, si ha evidentemente che, a parità di insieme di addestramento e di parametri, ad un qualsiasi classificatore prodotto da RealGPE resterà associato un insieme di misclassificazione incluso (non necessariamente strettamente) in quello associato all'analogo classificatore prodotto da IdealGPE.

In parole più semplici, RealGPE non può sbagliare più di IdealGPE. Vale allora la disequazione (2.1) a pagina 23.

7.2 Varianti

7.2.1 Gradi di libertà

Nella formulazione degli algoritmi proposti, se i campioni si trovano in uno spazio n-dimensionale, i problemi da risolvere possono coinvolgere più o meno variabili decisionali:

- $\frac{n(n+1)}{2} + n$
- $n + 1$
- 1

A patto che le proprietà descritte continuino a valere, è possibile naturalmente modificare l'algoritmo in modo da scalare in maniera diversa la complessità delle regioni costruite al crescere di $|S|$. Per esempio, è possibile vincolare la matrice A ad essere diagonale e risolvere un problema come il seguente con $2n$ variabili decisionali

$$
\begin{array}{ll}
P_{2n}\ (x_b, y(x_b)): & \\
\quad \min_{A=\in\Re^{n\times n}, B\in\Re^{1\times n}} & \text{Traccia}(A) = \sum_{i=1}^{n} a_{ii} \\
\quad \text{con vincoli:} & (x_j - x_b)^T A (x_j - x_b) + B(x_j - x_b) \geq 1, \\
 & \qquad \forall x_j \in \mathrm{L}, y(x_j) \neq y(x_b) \\
 & A \text{ diagonale, } a_{ii} \geq 0,\ 1 \leq i \leq n
\end{array}
$$

Nel quale, in breve, si rinuncia alla libertà di orientare le regioni costruite a piacere. L'idea è che vincolare solo certe variabili può evitare che l'algoritmo passi troppo in fretta ad utilizzare regioni sferiche costruite risolvendo problemi unidimensionali, problema già mitigato nell'attuale versione dell'algoritmo grazie al problema con $n + 1$ variabili. D'altro canto, l'utilizzo *soltanto* d'ipersfere ha un vantaggio teorico di cui si rende conto nella seguente sezione.

7.2.2 Rilassamento delle ipotesi

È possibile costruire un algoritmo per il quale le proprietà teoriche siano garantite anche nel caso di distribuzioni dei campioni particolari, per le quali non è definibile una densità. L'ostacolo cui si deve far fronte è la possibilità che sul bordo di una regione vi siano più vincoli attivi di quanti siano i punti di supporto. Si supponga di procedere apportando le seguenti variazioni ad IdealGPE (per ora non si consideri EXTMIX):

- Per costruire regioni si utilizzi solamente P_{CMax}. Si noti che ciò fa sì che ogni regione sia l'interno di un'ipersfera. Generalmente, se sul bordo dell'ipersfera risultante dalla soluzione di P_{CMax} esistono più punti di natura opposta a x_b, nessuno di essi può dirsi di supporto, in quanto, se rimosso, non modifica la soluzione, che resta determinata dagli altri. D'altra parte, la presenza di uno qualsiasi di essi è sufficiente a ricostruire esattamente l'ipersfera.

- Trovata la soluzione k^* di $P_{\mathrm{CMax}}(x_b, y(x_b))$, supponendo che t_h sia la funzione definita su $\mathbb{R}^n$ che seleziona la h-esima componente di un punto, si proceda così:
  - Sia $\mathcal{B}$ il bordo dell'ipersfera; $l_1 := \min\{t_1(x) \text{ t.c. } x \in \mathcal{B} \cap L\}$; si inizi a "costruire" un insieme ξ' inizializzato come
    $$\xi' := \{x \in X \ t.c. \ k^*||x - x_b||^2 < 1\}$$
  - Si aggiunga a ξ' l'insieme $\{x \in \mathcal{B} \text{ t.c. } t_1(x) < l_1\}$
  - Sia $l_2 := \min\{t_2(x) \text{ t.c. } x \in \mathcal{B} \cap L,\ t_1(x) = l_1\}$, e si aggiunga a ξ' l'insieme
    $$\{x \in \mathcal{B} \text{ t.c. } t_1(x) = l_1 \text{ e } t_2(x) < l_2\}$$
  - Si proceda analogamente fino ad n, aggiungendo da ultimo a ξ' l'insieme
    $$\{x \in \mathcal{B} \text{ t.c. } t_1(x) = l_1,\ t_2(x) = l_2, ..., t_{n-1}(x) = l_{n-1} \text{ e } t_n(x) < l_n\}$$
  - Si definisce infine la regione m-esima (ξ_m, κ_m),

    $\xi_m := \xi'$
    $\kappa_m := y(x_b)$

Seguendo il procedimento descritto si otterrà una regione di classificazione che può occupare anche il bordo dell'ipersfera. È immediato notare che essa sarà determinata da *un solo* punto sulla superficie: quello la cui prima componente è più piccola o, se più d'uno condividono la prima componente, quello che ha la seconda più piccola e così via fino ad esaurire le componenti e individuare necessariamente un punto. Tale punto è di supporto poiché, se venisse rimosso, la regione costruita (di natura opposta alla sua) andrebbe a coprirlo sia nel caso in cui esso fosse l'unico punto sulla superficie (il solito caso), sia nel caso in cui vi fossero altri punti sulla superficie. Per fissare le idee, supponiamo che il punto che ora stiamo rimuovendo $\widehat{x}$ fosse quello con la prima componente minore; risolvendo lo stesso problema senza $\widehat{x}$, si troverebbe sul "confine" della nuova regione costruita un x' tale che $l_1 := t_1(x') > t_1(\widehat{x})$. Ma allora, per costruzione, $\widehat{x}$ apparterrebbe ad una regione classificata con natura opposta alla sua. Il discorso sarebbe analogo anche se fosse stato $t_1(x') = t_1(\widehat{x})$ ma $t_1(x') > t_2(\widehat{x})$.

Seguendo un approccio del tutto analogo, è possibile modificare anche EXTMISC in modo da garantire che possa sempre essere trovato il numero voluto di punti di supporto anche qualora, per esempio, tutti i campioni dell'insieme di addestramento si trovassero su una circonferenza centrata attorno ad x_0. Dopo aver risolto $P_{\mathrm{CMax}'}$, basta "portare nel bordo", procedendo come prima, la regione di misclassificazione estesa; individuare il punto che "frena" la sua espansione e metterlo S_{esteso} (escludendolo, come ogni altro punto individuato da questa procedura, da $\mathcal{M}$, per evitare di misclassificare nell'insieme di addestramento). Se $|S_{\mathrm{esteso}}|$ è ancora minore di d, si risolve di nuovo $P_{\mathrm{CMax}'}$. Non è detto che l'ipersfera aumenti di raggio rispetto a prima: potrebbe semplicemente portare all'individuazione di un nuovo punto sul suo bordo.

Capitolo 8

Problematiche affini e note bibliografiche

8.1 La ricerca di garanzie sull'errore

Il punto di forza di RealGPE è nella possibilità di stabilire un maggiorante (*bound*), rispettato con alta probabilità e non lasco, dell'errore di generalizzazione su un insieme di campioni di test. Questo è un risultato notevole laddove si richieda un controllo stretto sull'errore commesso nella classificazione. Nell'ambito della letteratura sull'apprendimento induttivo, i maggioranti finora noti, per quanto teoricamente interessanti, si rivelano decisamente troppo spesso molto laschi per problemi reali. La teoria più studiata che conduce a maggiorare l'errore di generalizzazione a priori è certamente quella dell'apprendimento statistico dovuta a Vapnik e Chervonenkis (si veda [62, 60]). Il maggiorante che è possibile ricavare attraverso la teoria dell'apprendimento statistico fa uso della nozione di *dimensione VC* di una classe di funzioni $\mathcal{H}$ tra le quali è scelta, in seguito all'addestramento, l'ipotesi. La dimensione VC fornisce in qualche modo una misura della "flessibilità" della classe $\mathcal{H}$ e permette di tenere sotto controllo la capacità di generalizzazione. Per avvicinare intuitivamente il lettore all'impostazione che conduce al maggiorante a priori, senza introdurre rigorosamente il concetto di dimensione VC ed i passaggi matematici necessari alla dimostrazione, si ipotizza di avere a che fare con uno spazio delle ipotesi di dimensione finita[1]. Sia $h \in \mathcal{H}$ un'ipotesi che classifica correttamente gli N campioni estratti indipendentemente che costituiscono l'insieme di addestramento. Ci chiediamo quale sia la probabilità che tale ipotesi sia "cattiva", cioè abbia un errore di generalizzazione PE(h)

[1] Questo scelta espositiva è quella adottata in [16].

superiore ad una certa soglia ε. Ciò equivale a chiedersi la probabilità che la N-upla di campioni estratti cada nel sottoinsieme di quelle che conducono

- alla scelta dell'ipotesi h consistente gli N campioni
- ad un errore di generalizzazione maggiore di ε

L'errore di generalizzazione non è altro che la misura del sottoinsieme di X per cui $h(x) \neq y(x)$. Ne viene che i campioni estratti appartengono ad un insieme di misura $1-\varepsilon$ e, per l'indipendenza, si ottiene che la probabilità cercata è $(1-\varepsilon)^N$. Questo risultato riguarda però solo *una certa* ipotesi h consistente con i dati. Si vuole invece un maggiorante valido per l'algoritmo che parte da N campioni e produce *una qualche* ipotesi consistente con l'insieme di addestramento. Si noti che la generazione di un classificatore con un errore di generalizzazione maggiore di ϵ *implica* l'esistenza un'ipotesi "cattiva" $\widehat{h} \in \mathcal{H}$, cioè consistente con i dati dell'insieme di addestramento e tale che $\mathrm{PE}(\widehat{h}) > \varepsilon$. Ciò comporta che la probabilità che l'errore di generalizzazione sia maggiore di ε non può superare quella che esista un'ipotesi "cattiva" $\widehat{h} \in \mathcal{H}$. In formule (i d_i rappresentano gli esempi (x_i, y_i))

$$\mathbb{P}(h \text{ } \textit{"cattiva"}) := \mathbb{P}((d_1, ..., d_N) \text{ t.c. } \mathrm{PE}(h_{\textit{prodotta a partire da } d_1,...,d_N}) > \varepsilon) \leq$$

$$\leq \mathbb{P}(\exists \widehat{h} \text{ } \textit{"cattiva"}) \leq |\mathcal{H}|(1-\varepsilon)^N \tag{8.1}$$

In questo caso, la misura della ricchezza (complessità) della classe delle ipotesi in campo è data dalla cardinalità $\mathcal{H}$. All'aumentare della complessità delle ipotesi si perde la capacità di generalizzare: questo è un concetto anche intuitivo che rende conto del fenomeno detto del sovradattamento. Il maggiorante trovato mostra inoltre che è possibile tenere sotto controllo l'errore *uniformemente* sullo spazio delle ipotesi: aumentando N si può sempre garantire che l'errore di generalizzazione si faccia piccolo, qualsiasi ipotesi consistente sia scelta dall'algoritmo. La teoria di Vapnik e Chervonenkis generalizza gli aspetti visti introducendo la dimensione VC come parametro per il controllo della complessità di classi di ipotesi anche infinite. Se la dimensione VC di $\mathcal{H}$ è finita, allora sarà possibile, attraverso la scelta di N sufficientemente grandi, tenere sotto controllo l'errore di generalizzazione o, meglio, l'errore che si commette confondendo l'*errore empirico* sull'insieme di addestramento con l'errore di generalizzazione. Ciò significa, fissato un η tra 0 e 1, ottenere una disequazione del tipo

$$\mathrm{PE}(h) \leq \text{errore empirico} + \text{confidenzaVC}(\mathrm{dimVC}(\mathcal{H}), \eta) \tag{8.2}$$

che vale con probabilità almeno $1-\eta$. Dove l'*errore empirico* è il tasso di errore che si commette sui campioni dell'insieme di addestramento e la

confidenza VC un termine[2] che ha una dipendenza dalla dimensione VC di $\mathcal{H}$.

Ora, si noti peraltro che, laddove entra in gioco una dipendenza dall'errore empirico, si perde la possibilità di avere un maggiorante sull'errore definito a priori. Il maggiorante (8.2) col termine empirico a zero vale per qualsiasi ipotesi consistente selezionata dall'algoritmo, il che vuol dire che, a priori, non vale *sempre*, poiché potrebbero non esistere ipotesi in grado di classificare correttamente tutti i campioni osservati, così come non si può supporre di trovare sempre entro un *fissato* insieme $\mathcal{H}$ di cardinalità finita un'ipotesi in grado di classificare qualsiasi multi-estrazione di campioni di addestramento, a meno di non fare ipotesi aggiuntive sulla realtà da apprendere. Anche ignorando queste precisazioni, sarebbe perlopiù illusorio pensare di utilizzare i maggioranti così trovati per stimare l'accuratezza di un algoritmo. Essi infatti, a priori, hanno più che altro la funzione di fornire una giustificazione teorica alla pretesa che un algoritmo sia in grado di apprendere, garantendo che, anche nel caso peggiore, sia possibile in qualche modo avere sicurezze sull'andamento dell'errore. Essi sono utilizzati anche come guide nella selezione dei parametri per certi algoritmi. Un'idea, basata su questi concetti, è quella della SRM (Structural Risk Minimization): pur con possibili varianti, si tratta di costruire (a priori[3]) una gerarchia di classi $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq ... \subseteq \mathcal{H}_k \subseteq ...$ di ipotesi di complessità (cioè dimensione VC) crescente. Per ogni classe si individua il classificatore che minimizza l'errore empirico sul training set. Il classificatore scelto sarà quello che minimizza la somma di errore empirico e confidenza VC per la relativa classe di appartenza.

Concludendo, il fatto che il maggiorante (8.2) valga per qualsiasi distribuzione lo rende, in generale, molto lasco e perlopiù inservibile nel tentativo di stimare le prestazioni effettive di un algoritmo. Non stupisce allora che esistano studi volti all'utilizzo di maggioranti che cerchino di trarre informazioni dai dati a disposizione. In questi casi, si abbandona la speranza di garantire a priori certe prestazioni, ma si assume di poter, guardando all'insieme di addestramento e all'ipotesi prodotta, trarre informazioni, anche dettagliate, sulla capacità di generalizzazione dell'ipotesi prodotta. Intuitivamente, l'idea di approcci simili è che, se un algoritmo è costretto a scegliere un'ipotesi molto complessa per spiegare i dati visti, è forte l'attesa che la realtà sia

[2] Un'espressione della confidenza VC, tratta da [7] è $\sqrt{\left(\frac{\dim\mathrm{VC}(\mathcal{H})\left(\ln\left(\frac{2N}{\dim\mathrm{VC}(\mathcal{H})}\right)+1\right)-\ln\left(\frac{\eta}{4}\right)}{N}\right)}$

[3] Per una generalizzazione di questo aspetto, si veda [57].

complessa; se invece si è scelta un'ipotesi molto semplice, allora è lecito attendersi che essa sia in grado di generalizzare bene. Metodi legati a questa considerazione si ispirano spesso all'importante lavoro di Littlestone e Warmuth [40] circa la correlazione tra schemi di compressione e apprendimento: è il caso per esempio della Set Covering Machine (si vedano [43, 42, 35]), ma certo l'approccio originario non è estraneo neppure al presente lavoro. Un altro esempio di come a partire dall'ipotesi risultante si possano dedurre informazioni sulla capacità di generalizzare si ha nell'ambito delle SVM, dove è possibile stimare il valore atteso dell'errore di generalizzazione. Se $N+1$ è il numero di campioni di addestramento e la soluzione è ricostruibile a partire da v vettori di supporto, si può affermare per certo che effettuando una procedura di *leave-one-out* si otterrà una percentuale di errore pari al più a $\frac{v}{N+1}$. Ora, l'errore di leave-one-out con $N+1$ campioni è una stima dell'errore di generalizzazione ottenuto da un addestramento con N campioni (si vedano [41, 20]): abbiamo quindi una stima dell'errore che dipende da un parametro del classificatore ottenuto[4]. Anche il classico algoritmo per la costruzione di SVM che mira ad incrementare il margine trova una giustificazione teorica nella volontà di massimizzare la capacità di generalizzazione della sottoclasse di ipotesi cui appartiene il classificatore *risultante*[5].

8.2 Panoramica sull'opzione di rigetto

Il classificatore binario proposto in questo lavoro di tesi ha la caratteristica di poter rifiutare di classificare alcuni campioni. Tale opportunità, la quale non è nuova nella letteratura, apre certo nuovi orizzonti a livello teorico, ma è già di riconosciuta utilità in molti ambiti pratici: la convinzione che talvolta sia meglio sospendere il giudizio anziché sbagliare, si trova esplicitamente espressa in diversi lavori di autori che si sono occupati di classificazione e di apprendimento automatico in generale; tant'è che molteplici sono i lavori nei quali appaiono considerazioni simili a quelle svolte nella sezione 5.2.1 del presente lavoro, nella letteratura statistica quanto in quella ingegneristica. Si riporta a puro titolo di esempio una considerazione contenuta in un recente articolo [38] ma che si ripresenta nella stessa identica sostanza in molti articoli precedenti, come sarà facile intuire dal prosieguo della presente panoramica:

[4] L'idea qui presentata è stata notevolmente affinata in [61] con l'introduzione del concetto di *span* dei vettori di supporto che consente di catturare l'idea che non tutti i vettori di supporto, se rimossi, conducono ad un errore in fase di leave-one-out.

[5] L'esposizione rigorosa del fatto che la minimizzazione del margine è legata ad una migliore generalizzazione si ha in [57]. Approcci precedenti non erano rigorosi, per quanto potessero essere "convincenti", si veda per es. l'analogia suggerita in in [7] circa l'applicazione della SRM ai classificatori "gap-tolerant".

> In many real applications, it is more convenient to withhold making a decision than making a wrong assignment, e.g. in medical diagnosis where a false negative outcome can be much more costly than a false positive. Reject options have been proposed to overcome these difficulties and to reduce misclassification risk[6].

In questa sezione si forniscono i riferimenti ad alcuni filoni di ricerca, legati anche ad ambiti che esulano dal problema della classificazione così com'è finora stato impostato, che si appoggiano sulla convinzione che non sempre decidere "ad ogni costo" sia una buona cosa e che dunque talvolta sia meglio rigettare un campione piuttosto che classificarlo.

Decisoni bayesiane Chow in [12, 13] propone un'estensione del metodo di decisione (multiclasse) basato sulla massimizzazione della probabilità a posteriori calcolata secondo la regola che porta il nome del reverendo Thomas Bayes. In un contesto bayesiano, il classificatore è visto, nel complesso, come uno strumento in grado di calcolare la probabilità che un certo campione appartenga ad una classe e, in funzione di ciò, di operare delle decisioni. Il punto di partenza è la conoscenza (incompleta) della distribuzione dei dati, che viene a meglio precisarsi a seguito di osservazioni. In particolare, per ogni classe k (da 1 a M), si può assumere di conoscere la probabilità che un qualsiasi dato estratto appartenga a quella classe. Le probabilità $\mathbb{P}(classe = 1) = \pi_1, ..., \mathbb{P}(classe = M) = \pi_M$ sono dette probabilità a priori, così come sono probabilità a priori quelle che definiscono, in assoluto, le probabilità con cui si presentano i campioni: $\mathbb{P}(x)$. La regola di Bayes lega le probabilità $\mathbb{P}(classe = k|x)$, dette *a posteriori*, alle probabilità $\mathbb{P}(x|classe = k)$, secondo la ben nota formula:

$$\mathbb{P}(classe = k|x) = \frac{\mathbb{P}(x|classe = k)\pi_k}{\mathbb{P}(classe = k)}$$

Assumendo di conoscere esattamente le probabilità a posteriori (le quali nei casi reali vanno stimate a partire dall'insieme di apprendimento, per esempio grazie ad una rete bayesiana, cui s'è accennato nella sezione 1.5), occorre risolvere il problema di come decidere circa l'appartenenza di un campione ad una classe: è possibile, definito il "rischio" e quindi la probabilità di commettere un errore, individuare la regola di decisione ottimale. L'approccio di Chow introduce la possibilità di non classificare dei campioni qualora

[6] In molte applicazioni reali, è più opportuno rifiutare di prendere una decisione piuttosto che sbagliare, per esempio nelle diagnosi mediche, dove un esito falso negativo può pesare assai più di un falso positivo. Opzioni di rigetto sono state proposte per superare queste difficoltà e per ridurre l'errore di misclassificazione.

la probabilità a posteriori non dia un'indicazione considerata soddisfacente, contestualmente al superamento di una soglia. Questo metodo consente di ottenere una relazione matematicamente rigorosa che lega la soglia di rigetto e il tasso di errore a patto però di conoscere, per ogni soglia, la probabilità di rigetto che essa comporta. Nella pratica, il tasso di rigetto può essere facilmente stimato (a classificazione avvenuta) semplicemente contando i campioni non classificati.

Un'ulteriore estensione dovuta a T.M. Ha (si veda [32]) permette al classificatore di assegnare ai campioni non *una* classe, bensì *un insieme* di classi (naturalmente, mettere un campione in corrispondenza con l'insieme di tutte le classi sostanzialmente equivale a non classificarlo). Anche in questo caso si introduce una soglia che permette di privilegiare l'assegnamento dei campioni a classi per le quali si ha una buona probabilità a posteriori. Un risultato, che *in nuce* è analogo a quello di Chow, permette di legare matematicamente il tasso di errore al numero medio di regioni a cui un campione viene assegnato.

L'idea di Chow è stata pure estesa da [19] per problemi in cui si ha una conoscenza incompleta circa le classi di appartenenza dei campioni, anche circa il loro numero.

Fumera, Roli e Giacinto in [28] dimostrano che la regola di Chow non fornisce il bilanciamento ottimo tra tasso di errore e di rigetto quando nella stima delle probabilità posteriori interviene un errore significativo e propongono in sostituzione una regola di decisione che coinvolge l'utilizzo di più soglie.

Regole plug-in [33] e [3], pubblicati rispettivamente nel 2006 e nel 2008, sono due articoli nei quali, peraltro, si lamenta la carenza di risultati teorici nella letteratura statistica circa il tema del rigetto dimostratosi tanto valido nella pratica. Nel primo, di Herbei e Wergkamp, al centro della trattazione viene posto lo studio di che cosa avviene quando, com'è giocoforza nelle applicazioni, nella regola di decisione ottima con rigetto (di Chow [13]), la probabilità a posteriori viene sostituita da uno stimatore, per esempio basato su regressione logistica o finestra di Parzen [1, 47]. Regole di decisione di questo tipo, in cui lo stimatore prende il posto della funzione di regressione, sono dette "innestate" (*plug-in*). È naturalmente di grande importanza conoscere quanto l'errore relativo alla regola di decisione ottima si discosta da quello ottenuto utilizzando la decisione plug-in e risultati in tal senso erano già noti in assenza di rigetto (ad es. si veda il Teorema 2.2 in [18]) . [33] tratta quindi delle condizioni a cui una decisione plug-in con rigetto può dirsi "vicina" a quella ottimale, ma considera anche un altro approccio, per il quale esistono ora risultati che suggeriscono implementazioni

efficienti (si veda ad esempio proprio [3]): si tratta di considerare, all'interno di un insieme $\mathcal{F}$ di classificatori, quello che minimizza il valore di una funzione empirica (cioè applicata ai dati disponibili) di rischio. Chiamando R il "valore" in uscita al classificatore quando un campione è rigettato, Y la classe di un campione X, $\widehat{f}$ il classificatore che minimizza la funzione empirica di rischio, $\widehat{L}_r(f)$ la funzione di rischio associata ad un $f \in \mathcal{F}$ (definita uguale a $r\mathbb{P}\{f(X) = R\} + \mathbb{P}\{f(X) \neq Y, f(X) \neq R\}$ in modo da penalizzare gli errori e, secondo un parametro r, i rigetti), si vorrebbe che la differenza $\Delta(\widehat{f}) = L_r(\widehat{f}) - \min_{f:\mathbb{R}^k \to \{0,1,R\}(\in \mathcal{F})} L_r(f)$ soddisfacesse la seguente diseguaglianza

$$\Delta(\widehat{f}) \leq C \inf_{f \in \mathcal{F}} \Delta(f) + R_n$$

con C costante e R_n che dipendesse solo dal numero di campioni, una condizione sulla funzione di regressione e la dimensione di $\mathcal{F}$. Nello stesso articolo si mostrano risultati che vanno in questa direzione. Lo stesso contesto concettuale regge qualora si volessero introdurre dei costi diversi per errori di "tipo I" ed errori di "tipo II": talvolta un falso positivo può essere ben più grave di un falso negativo, o viceversa (questa terminologia sarà ripresa in un successivo paragrafo).

Diversi tipi di rigetto In letteratura, sono utilizzati i termini

- *distance rejection*(rigetto per distanza) per indicare l'assegnamento di un campione a nessuna classe. Si fa riferimento a questo concetto quando si vuole poter vedere un campione come un *outlier*, cioè un campione con valore aberrante.

- *exclusive classification*(classificazione esclusiva) per indicare l'assegnamento "classico" di un campione ad una classe.

- *ambiguity rejection*(rigetto per ambiguità) per indicare l'assegnamento di un campione a più classi. Si fa riferimento a questo concetto quando le classi di appartenenza dei campioni si possono "sovrapporre" o comunque risulta difficile distinguere tra l'appartenenza all'una o all'altra.

Le scelte tra i possibili tipi di rigetto, nonché le loro combinazioni, sono state oggetto di studi e proposte. Rilevante in tal senso è il lavoro di Frélicot [25] che mira a una trattazione indipendente dal modello matematico su cui il classificatore opera (probabilistico, fuzzy,...), proponendo un modello secondo cui un campione viene dapprima sottoposto ad una strategia di accettazione, che può sfociare in una mancata assegnazione a qualsivoglia classe, oppure concludersi in una classificazione univoca o a più classi.

Apprendimento affidabile, probabilmente quasi sempre utile Rivest e Sloan, in [54], si ispirano all'approccio PAC per introdurre il concetto di algoritmo in grado di apprendere*affidabilmente e probabilmente quasi sempre utilmente* (*reliably, probably almost always usefully*) una classe di concetti. Richiamiamo in sintesi l'idea dell'apprendimento PAC (probabilmente approssimativamente corretto), introdotto da Valiant in [59]. Sia $\mathcal{C}$ una classe di concetti definiti su un universo U (un concetto può essere visto come una funzione $c : U \rightarrow [0, 1]$) che possono essere estratti secondo una distribuzione D. Un algoritmo A è in grado di apprendere una classe di concetti $\mathcal{C}$ in maniera probabilmente approssimativamente corretta se l'algoritmo per qualsiasi distribuzione di probabilità D, concetto c e parametri $\delta > 0$ ed $\epsilon > 0$, è in grado di computare in maniera efficiente (polinomiale) un'ipotesi c' tale che con probabilità $1 - \delta$ valga

$$\sum_{c'(x) \neq c(x)} D(x) < \epsilon$$

L'algoritmo di apprendimento deve processare gli esempi in modo computazionalmente efficiente e deve essere in grado di produrre con alta probabilità una buona approssimazione del concetto da apprendere.

Ammettendo la possibilità di non classificare alcuni campioni, un algoritmo apprende *affidabilmente e probabilmente quasi sempre utilmente* nel senso di Rivest e Sloan se produce in uscita, in un tempo polinomiale, un programma Q che riceve in ingresso un elemento di $x \in U$ e che ha il compito di classificare x in modo da riprodurre la funzione (il concetto) c che si vuole apprendere. L'algoritmo proposto nell'articolo è

- *affidabile* (reliable), in quanto se $Q(x)$ vale 0, allora $c(x)$ vale 0; se $Q(x) = 1$, allora $c(x) = 1$. Ciò significa che Q non deve sbagliare mai: se fornisce una risposta, è quella giusta

- *probabilmente quasi sempre utile* (probably almost always useful), in quanto Q può rifiutarsi di dare una risposta (rigetto), ma si deve poter affermare, con una confidenza $(1 - \delta)$, che la probabilità che l'insieme degli $x \in U$ per cui Q non fornisce risposta sia minore di un certo parametro $\epsilon > 0$ scelto a priori. Si deve cioè avere con probabilità $1 - \delta$ che

$$\sum_{Q(x)=R} D(x) < \epsilon \quad \text{(Q vale R laddove rigetta)}$$

Tale approccio è possibile per l'apprendimento di qualsiasi concetto esprimibile come funzione booleana di dimensione polinomiale[7] e utilizza una forma di apprendimento gerarchico per far fronte alla complessità di apprendimento di alcune classi di funzioni che, altrimenti, sarebbero verosimilmente[8] intrattabili, come mostrato da Valiant.

Il costo del rigetto In [14] si considera la possibilità di servirsi dell'opzione di rigetto nel campo delle reti neurali, mediante regole per la definizione di opportune soglie. La solita considerazione per cui, a seconda dei domini di applicazione, si può essere disposti a ridurre il rischio di misclassificazione, anche a costo di aumentare la percentuale di risposte mancate, viene matematicamente tradotta introducendo nel contesto una funzione P che descriva la bontà delle prestazioni ottenute dalla rete neurale in termini non solo di tasso di errori commessi (R_m) e classificazioni corrette (R_c), ma anche di risposte non date (R_r). Su tale funzione non sono poste restrizioni particolari, se non quelle ovvie adatte a renderla un ragionevole indice della qualità del classificatore:

$$\begin{aligned} \frac{\partial P}{\partial R_c} &> 0 \\ \frac{\partial P}{\partial R_r} &< 0 \\ \frac{\partial P}{\partial R_m} &< 0 \\ \left|\frac{\partial P}{\partial R_r}\right| &< \left|\frac{\partial P}{\partial R_m}\right| \end{aligned}$$

L'ultima equazione suggerisce il maggiore impatto (negativo) di un aumento del tasso di errore rispetto a quello del tasso di rigetto. Tale approccio, di carattere generale, può essere in particolare applicato al caso delle reti neurali per estendere, con l'introduzione del rigetto, il metodo detto del *winner*

[7] Sia c una formula booleana (che esprime un concetto come funzione di n variabili booleane). Si definisce la dimensione di $c$come il numero di bit necessari a rappresentarla secondo un qualche predefinito schema di rappresentazione. Per "dimensione polinomiale" si intende che la dimensione della formula è polinomiale nel numero di variabili.

[8] Ciò è vero se esiste una funzione "one-way" (computazionalmente non invertibile). Tale congettura è ritenuta verosimile. Se dimostrata, porterebbe immediatamente a concludere che $P \neq NP$, mentre se falsa avrebbe effetti devastanti nel ramo della criptografia, anche se non implicherebbe immediatamente $P = NP$ (per ulteriori approfondimenti si veda per esempio il capitolo 9 di "Handbook of Applied Cryptography", di A. J. Menezes et al., CRC Press, 1997).

takes all (cioè del "chi vince piglia tutto"), secondo cui un campione viene assegnato ad una classe corrispondente al neurone di uscita che assume il massimo valore. L'idea è sempre quella di individuare una soglia ottimale che, bilanciando misclassificazioni e rigetti, consenta di massimizzare la qualità del classificatore espressa da P. Per individuare la soglia occorrerebbe conoscere delle distribuzioni che vengono, ancora una volta, soltanto stimate a seguito dell'attività di classificazione sull'insieme di apprendimento.

Come valutare un classificatore In statistica, si usa definire un'ipotesi nulla H_0, che solitamente rappresenta lo "stato normale" delle cose, per esempio l'assenza di una certa patologia in medicina. Sulla base di osservazioni si decide se accettare o rifiutare l'ipotesi nulla. A seconda che essa venga accettata o rifiutata e che sia vera o falsa, si distinguono i quattro casi presenti nella seguente tabella:

	H_0 Vera	H_0 Falsa
Accetto H_0	Corretto (TN)	Falso negativo (FN) (tipo II)
Rifiuto H_0	Falso positivo (FP) (tipo I)	Corretto (TP)

Quando in ogni casella è riportato il numero di campioni classificati nel modo corrispondente, la tabella viene chiamata **matrice di confusione**. Dal punto di vista di un classificatore si parla normalmente di *classe obbiettivo* (*target class*), per indicare la classe corrispondente alla negazione dell'ipotesi nulla (es. la classe degli individui malati) e di *classe di sfondo o background*, per indicare la classe corrispondente all'ipotesi nulla (individui che non presentano la patologia in esame).

Dato un classificatore (senza rigetto), indicato con TP il numero dei casi positivi classificati correttamente e con TN il numero dei negativi classificati correttamente, si definisce *accuratezza* il valore

$$\frac{TP + TN}{Numero\ dei\ campioni}$$

Pare opportuno osservare che, talvolta, l'accuratezza può essere un parametro davvero poco o per nulla soddisfacente, specialmente nel caso in cui le classi abbiano una probabilità molto diversa. Si pensi per esempio ad un classificatore che ha il compito di pronunciarsi circa la presenza di una patologia abbastanza rara in un paziente. Immaginiamo di studiare i casi di 1000 pazienti, supponiamo che soltanto due di essi siano davvero malati e che il classificatore si comporti così: dia soltanto un falso positivo e 997 veri negativi, ma nessun vero positivo e 2 falsi negativi. L'accuratezza sarebbe del

99.7%, ma certo è che ben pochi si potrebbero dire soddisfatti della prestazione! Altri parametri e metodi per valutare la bontà dei classificatori binari sono quindi stati oggetto di studi. I due rapporti che riguardano il tasso di veri positivi ($\frac{TP}{TP+FN}$) e dei veri negativi ($\frac{TN}{TN+FP}$) sono detti rispettivamente *sensitività* (nell'esempio, un inequivocabile 0%) e *specificità* (nell'esempio quasi il 99.9%) e danno com'è evidente maggiore informazione. Talvolta la classificazione si basa sulle probabilità stimate che un certo oggetto appartenga alla classe target o comunque su un numero che suggerisce quanto sia verosimile l'appartenenza ad una classe (tali classificatori sono detti *ranker*): oltre una certa soglia, un campione non viene più assunto come negativo ma sarà classificato come positivo. Il tasso dei falsi positivi ($\frac{FP}{FP+TN}$, cioè $1 - $ *specificità*) in ascissa viene utilizzato in coppia con la sensitività in ordinata per individuare i punti di una curva detta **ROC** (receiver operating characteristic), tracciata al variare della soglia del classificatore. Tali curve, che nella realtà sono stime costruibili con opportuni metodi, iniziano sempre all'origine (laddove si accetta sempre, indiscriminatamente, l'ipotesi nulla) e terminano all'angolo in alto a destra del quadrante unitario (quando si rifiuta sempre l'ipotesi nulla): più la curva sta verso l'alto e a sinistra, maggiore è la capacità del classificatore di discriminare gli oggetti appartenenti alla classe target da quelli della classe di background. Da un classificatore che effettua scelte casuali, ci si attende, d'altro canto, un grafico a cavallo della bisettrice. Un buon "riassunto" della ROC è il numero che rappresenta l'area sottesa da tale curva, detto AUC (o AUROC) (si veda [5]). Esso gode di buone proprietà e può essere calcolato con algoritmi che non prevedono l'effettivo computo dell'area sottesa dalla ROC. Vi sono risultati che mostrano come classificatori che hanno come obiettivo di massimizzare l'AUC si rivelino migliori anche sotto il punto di vista dell'accuratezza (si veda [39]).

La curva ROC può essere estesa al caso di classificatori multi-classe (si veda per esempio [22]) ed è al centro dell'interesse di alcuni metodi anche per i classificatori con rigetto. In [50], che riprende ed estende alcune idee in [58] viene proposto un *metaclassificatore* con rigetto basato su due classificatori binari (senza rigetto) e appartenenti ad una stessa famiglia che condivide una stessa ROC (possono essere lo stesso *ranker* che utilizza soglie diverse), tali che sia rispettata una condizione di interdipendenza che riguarda le rispettive uscite. Sulla struttura della matrice di confusione, può essere definita una *matrice di costo*, che assegna appunto un costo ai falsi positivi o ai falsi negativi, nonché, con una semplice estensione, ai positivi e ai negativi su cui il classificatore sospende il giudizio. Minimizzando tale costo è possibile ricavare i parametri che definiscono i due classificatori componenti: ciò significa individuare due punti sulla ROC. Poiché non sempre si sa quantificare il costo dei tipi di rigetto, sono proposti anche altri due metodi, interessanti poiché

mettono in luce i due aspetti del problema di bilanciare opportunamente tasso di rigetto e misclassificazione: si può fissare un tetto massimo al tasso di errore di misclassificazione e minimizzare la percentuale ottima di campioni rigettati; oppure si può fissare un tetto alla percentuale di rigetto e quindi minimizzare il tasso di misclassificazione. In entrambi i casi il risultato è ancora costituito dai parametri relativi ai due classificatori componenti ottimi. Operativamente, si parte da un certo insieme di campioni, si costruisce (per esempio mediando opportunamente tra i risultati ottenuti effettuando test attraverso validazioni incrociate) la ROC corrispondente alla famiglia di classificatori scelti e, attuando una delle tre strategie, si individuano i parametri dei due classificatori componenti. Si noti che, anche se si utilizza una cifra di merito che fa sì che il classificatore individuato possa dirsi "ottimo", tale ottimizzazione si basa pur sempre su una ROC individuata a partire da un certo insieme di campioni. Dati empirici mostrano correlazioni di tipo tendenzialmente lineare tra alcuni parametri, per esempio tra la frazione di campioni rigettati e il *miglioramento* del costo (espresso nell'articolo come percentuale da sottrarre a un valore di riferimento per lo stesso: maggiore è il miglioramento, minore è il costo).

Su un'estensione tridimensionale della ROC si basa l'analisi svolta in [37], nella quale si investiga la relazione tra bontà della classificazione e rigetto in classificatori utilizzati per problemi nei quali si ha una classe target ben definita e una classe *outlier* per la quale non si hanno campioni; si pensi per esempio al riconoscimento di segnali stradali nel quale occorre distinguere i segnali tra loro ma anche capire se un certo oggetto è o non è un segnale stradale: è evidente la difficoltà di campionare opportunamente la classe outlier di tutto ciò che non è un segnale stradale.

Cautious classifiers Classificatori che utilizzano il rigetto sono pure i cosiddetti *cautious classifiers*, i "classificatori cauti" introdotti da Ferri e Hernandez-Orallo in [21] che hanno il vantaggio di poter essere utilizzati facilmente in classificazioni multi-classe. Ferri e Flach in [23] introducono i classificatori deleganti (*delegating classifiers*) che utilizzano più classificatori cauti "in cascata" in modo da ottenere un classificatore completo (senza rigetto). I campioni non classificati ad un primo stadio sono utilizzati per addestrare un vero e proprio classificatore allo stadio successivo[9].

Altri approcci In [48], senza pretese di ottimalità, si mostra come introdurre, su algoritmi specifici, il rigetto al fine di aumentare l'accuratezza.

[9]Tale idea ha forti affinità con la strategia già menzionata nella nota 2 a pagina 22.

I *Confirmation rule sets*, insiemi di regole di conferma, sono proposti in [30]. L'approccio è multiclasse: si individuano insiemi molto specifici di regole (si veda la sezione 1.5.2) che rivelano campioni solo di una certa classe target e che non decidono circa campioni che non rivelano come appartenenti ad essa. Una regola, per essere accettata, deve soddisfare un certo grado di "copertura" sui campioni nell'insieme di apprendimento e, tendenzialmente, non deve misclassificarne nessuno, anche se tale vincolo può essere rilassato per far fronte a dati rumorosi. L'algoritmo di classificazione finale si basa sul numero di regole che scattano segnalando l'appartenenza di un campione all'una o all'altra classe, ammettendo il rigetto nel caso in cui nessuna regola scatti o in presenza di disaccordi.

Nell'ormai vasta letteratura sulle tecniche di costruzione di *combinazioni di classificatori*, il rigetto è preso in considerazione al fine di dirimere la discordia tra le uscite dei classificatori componenti che, pesate e aggregate, danno luogo all'uscita finale (un approccio del genere è seguito anche nel già citato [50]). Un'analisi alla luce della teoria sviluppata da Chow del bilanciamento errore-rigetto di classificatori combinati linearmente è svolta in [26].

Per quanto la teoria possa essere rimasta un po' indietro circa lo studio delle opportunità fornite dal rigetto, si è da tempo evidenziato anche sperimentalmente come, per esempio nel caso delle **SVM**, l'introduzione del rigetto conduca a prestazioni notevolmente migliori. Tradizionalmente, nelle SVM il rigetto è introdotto in funzione della distanza di un campione dall'iperpiano separatore ottimo: quando la distanza è minore di una certa soglia, il campione non viene classificato. In [27] si segue una linea teoricamente più controllabile, nel solco della SRM (Support Risk Minimization, si veda il cenno nella sezione 8.1) di Vapnik, secondo cui il classificatore binario si ottiene come risultato di un problema di minimizzazione di una cifra di merito (che penalizza il rigetto e le misclassificazioni) che conduce alla costruzione nello spazio dei campioni di due piani paralleli in grado di dividere lo spazio in tre regioni: due classificate secondo ciascuna delle due classi e una (quella tra i due piani) corrispondente all'area di rigetto.

L'algoritmo in [27] è inefficiente e articoli più recenti riprendono il problema. Nel 2008, in [31], Grandvalet e altri mostrano che, per opportuni nuclei, addestrare una SVM con la funzione costo da essi proposta (che può tenere conto di costi diversi per falsi positivi e negativi, nonché per rigetti di campioni positivi e negativi) conduce ad un classificatore con rigetto universalmente consistente, cioè tale che, al crescere del numero di campioni

nell'insieme di addestramento, il rischio della funzione di decisione appresa tende al rischio della regola ottima di Chow [13]. Essi presentano la propria implementazione come la prima fondata su solidi principi ed efficiente.

Capitolo 9

Conclusioni e prospettive

In questa tesi è stato proposto un innovativo algoritmo per la costruzione di classificatori con opzione di rigetto dalle interessanti proprietà teoriche. Come si è visto nella sezione 8.1, e per quanto ci è noto, nel ricercare garanzie circa l'errore di classificazione, si è finora sempre tentato da un lato di tutelarsi nel caso peggiore, ottenendo maggioranti molto laschi, dall'altro di sfruttare informazioni a posteriori per ottenere previsioni sulla generalizzazione. L'approccio introdotto con IdealGPE, e quindi RealGPE, ammette invece che la realtà possa talvolta essere complicata, talvolta semplice; il grado di complessità del classificatore è correlato ad parametro dell'algoritmo (d), che stabilisce il numero di punti dell'insieme di addestramento necessari per la ricostruzione della funzione di decisione, e a tale grado di complessità è legato l'errore di generalizzazione, sul quale si ha pertanto un forte controllo in probabilità. Si perviene così ad un disaccoppiamento tra l'*accuratezza* dell'algoritmo, che è sotto controllo, e la sua *efficacia*, che l'algoritmo massimizza euristicamente, trovando un limite, che si manifesta nella parziale copertura dello spazio dei campioni, nell'intrinseca difficoltà della realtà oggetto di apprendimento.

9.1 Possibili direzioni di ricerca

Per quanto i risultati sperimentali ottenuti suggeriscano la competitività di RealGPE, è opportuno continuare a verificarne l'efficacia in casi reali, anche per dataset di grosse dimensioni e con molti attributi. Insieme al lavoro pratico di test, restano aperte possibilità di migliorare l'algoritmo, per esempio provando a renderlo più flessibile. Nuove opportunità possono ovviamente venire dall'indagine teorica, sia per ciò che concerne le basi matematiche offerte da teorie sviluppate inizialmente nell'ambito dell'ottimizzazione con-

vessa, sia per la possibilità di studiare algoritmi che offrano garanzie sull'errore sotto ipotesi ancora più lasche, in particolare per i quali non si richieda necessariamente di estrarre campioni distribuiti secondo una densità.

Appendice A

Dimostrazioni

A.1 Dimostrazione del risultato fondamentale

D'ora in poi, M sarà un numero fissato $M \geq d$, e l'insieme Z delle M-uple in Δ^M contenenti elementi ripetuti, o per le quali C non è in grado di selezionare tutti e soli i punti di supporto, deve avere probabilità nulla ($\mu^M(Z) = 0$). A questo punto, è possibile partizionare l'insieme $\Delta^M \setminus Z$, considerando come appartenenti alla stessa classe di equivalenza le M-uple per le quali i d punti di supporto, individuati da C, occupano lo stesso posto all'interno della M-upla. Poiché ci sono $\binom{M}{d}$ modi per scegliere d posti all'interno di una sequenza di M elementi, definiamo $\binom{M}{d}$ classi di equivalenza che costituiscono una partizione di Δ^M a meno di un insieme di misura nulla: le chiamiamo S_i, con $i = 1, ..., \binom{M}{d}$. Per fissare le idee, concentriamo l'attenzione sulla classe S_1 che assumiamo così definita:

$$S_1 = \{(\delta_1, ...\delta_M) \in \Delta^M \setminus Z \text{ t.c. } C(\delta_1, ..., \delta_d, ..., \delta_M) = C(\delta_1, ..., \delta_d)\}$$

Per la proprietà "**d**", per qualsiasi $(\delta_1, ..., \delta_M) \in S_1$ vale che

$$E_{(\delta_1,...,\delta_M)} = E_{(\delta_1,...,\delta_d)}$$

e, per la proprietà "**c**", che

$$\delta_{d+1} \notin E_{(\delta_1,...,\delta_d)}; \delta_{d+2} \notin E_{(\delta_1,...,\delta_d)}; ...; \delta_M \notin E_{(\delta_1,...,\delta_d)}.$$

Se definiamo

$$\tilde{S_1} = \{(\delta_1, ...\delta_M) \in \Delta^M \text{ t.c. } \delta_j \notin E_{(\delta_1,...,\delta_d)}, j = d+1, ..., M\}$$

e analogamente gli altri $\tilde{S}_i$ (cioè come gli insiemi di M-uple in Δ^M in cui agli elementi che occupano i posti selezionati secondo l'i-esimo modo è associato un insieme E al quale gli altri elementi della M-upla non appartengono), possiamo quindi banalmente concludere che $S_i \subseteq \tilde{S}_i$. È pur vero che $\tilde{S}_i \subseteq S_i$. Sia infatti, sempre per fissare le idee, $(\delta_1, ..., \delta_d, ..., \delta_M) \in \tilde{S}_1 \setminus Z$. Se tale M-upla appartenesse a un qualsiasi S_j con $j \neq 1$, uno dei punti δ_k (scegliamo un k tra $d+1$ e M), dovrebbe essere di supporto e quindi, per definizione, si avrebbe

$$\delta_k \in E_{(\delta_1,...,\delta_d,...,\delta_{k-1},\delta_{k+1}...,\delta_M)}.$$

Per definizione di $\tilde{S}_1$, qualsiasi δ_j $(j = d+1, ..., M)$ *non* deve appartenere a $E_{(\delta_1,...\delta_d)}$. Ora, immaginiamo di prendere un qualsiasi $\delta_j \neq \delta_k$: applicando la proprietà "**e**" si deduce che $\delta_j \notin E_{(\delta_1,...\delta_d,\delta_j)}$. Prendiamo ora un altro (sempre ammesso che ci sia) $\delta_{j'} \neq \delta_k$. Siccome $E_{(\delta_1,...\delta_d,\delta_j)} = E_{(\delta_1,...\delta_d)}$, è evidente che $\delta_{j'} \notin E_{(\delta_1,...\delta_d,\delta_j,\delta_{j'})}$. Si prosegue così finché degli $M-d$ elementi da aggiungere resta solo δ_k, e si arriva alla contraddittoria conclusione $\delta_k \notin E_{(\delta_1,...,\delta_d,...,\delta_{k-1},\delta_{k+1}...,\delta_M)}$.

Quindi, un elemento in $\tilde{S}_1 \setminus Z$ non può che appartenere a S_1. Dunque si può affermare che $\mu^M(S_i) = \mu^M(\tilde{S}_i)$.

Il prossimo obbiettivo è quello di trovare un modo per descrivere $\mu^M(\tilde{S}_1)$. Cerchiamo di ottenere una formula per $\mu^M\{(\delta_1, ..., \delta_d, ...\delta_M) \in \tilde{S}_1 \,|\, V_d(\delta_1, ...\delta_d) = \alpha)\}$. Supponiamo di fissare una certa d-upla $(\bar{\delta_1}, ..., \bar{\delta_d})$ per la quale la funzione V_d assuma il valore α. Per definizione di $\tilde{S}_1$,

$$(\bar{\delta_1}, ..., \bar{\delta_d}, \delta_{d+1}, ..., \delta_M) \in \tilde{S}_1 \textit{ se e soltanto se } \delta_j \notin E_{(\bar{\delta_1},...,\bar{\delta_d})},\ j = d+1, ..., M$$

Siccome $\mu(E_{(\bar{\delta_1},...,\bar{\delta_d})}) = V_d(\bar{\delta_1}, ..., \bar{\delta_d}) = \alpha$ e dato che su Δ^M è definita la misura di probabilità prodotto, si conclude che $\mu^{M-d}(\{(\bar{\delta_1}, ..., \bar{\delta_d}, \delta_{d+1}, ..., \delta_M) \in \tilde{S}_1\}) = (1 - V_d(\bar{\delta_1}, ..., \bar{\delta_d}))^{M-d} = (1-\alpha)^{M-d}$. Essendo il valore di tale probabilità indipendente dal valore esatto della d-upla per la quale $V_d = \alpha$, si ha che $\mu^M(\{(\delta_1, ..., \delta_d, \delta_{d+1}, ..., \delta_M) \in \tilde{S}_1 \,|\, V(\delta_1, ..., \delta_d) = \alpha\}) = (1-\alpha)^{M-d}$. V_d è una variabile casuale che può assumere valori in $[0, 1]$ definita su Δ^d, per cui integrare $(1-\alpha)^{M-d}$ su α tra 0 e 1 equivale ad integrare su Δ^d la funzione $(1 - V_d(\delta_1, ..., \delta_d))^{M-d}$ (un semplice cambio dello spazio di integrazione), perciò

$$\mu^M(\tilde{S}_1) = \int_0^1 (1-\alpha)^{M-d} F_{V_d}(\mathrm{d}\alpha)$$

Poiché il medesimo ragionamento si può ripetere invariato per tutti gli $\tilde{S}_i$ e $\mu^M(\tilde{S}_i \cap \tilde{S}_j) = 0$, per $i \neq j$, nonché $\mu^M\left(\bigcup_{i=1}^{\binom{M}{d}} \tilde{S}_i\right) = \mu^M(\Delta^M) = 1$, si ha

$$\binom{M}{d} \int_0^1 (1-\alpha)^{M-d} F_{V_d}(\mathrm{d}\alpha) = 1$$

Si osservi che si è pervenuti a questo risultato avendo fissato un $M \geq d$. Tale equazione deve quindi valere per infiniti valori di M. Il problema è equivalente a quello di ricavare una distribuzione a partire dalla conoscenza di tutti i suoi momenti: se la distribuzione di probabilità è definita su un intervallo finito (in questo caso $[0, 1]$), il problema ammette un'**unica** soluzione. Come si può verificare integrando per parti, una (e quindi l'unica) soluzione è

$$F_{V_d}(\alpha) = \alpha^d$$

Per ricavare la distribuzione di V_M ci concentriamo sulla probabilità dell'insieme $\Theta = \{\boldsymbol{\delta}_M = (\delta_1, ..., \delta_M) \text{ t.c. } V_M(\boldsymbol{\delta}_M) > \varepsilon\}$, che può essere scritta come

$$\mu^M(\Theta) = \mu^M\left(\bigcup_{i=1}^{\binom{M}{d}}(S_i \cap \Theta)\right) = \sum_{i=1}^{\binom{M}{d}} \mu^M(S_i \cap \Theta) = \binom{M}{d}\mu^M(S_1 \cap \Theta)$$

L'ultima eguaglianza deriva dal fatto che per tutti gli $S_i \cap \Theta$ si può applicare un ragionamento analogo al seguente:

$$\begin{aligned}\mu^M(S_1 \cap \Theta) &= \mu^M(\{(\delta_1, ..., \delta_M) \in S_1 \,|\, V_M(\delta_1, ..., \delta_M) > \epsilon)\}) = \\ &= \mu^M(\{(\delta_1, ..., \delta_M) \in S_1 \,|\, V_d(\delta_1, ..., \delta_d) > \epsilon\}) = \\ = \int_{V_d(\delta_1,...,\delta_d)>\epsilon} &(1 - V_d(\delta_1, ..., \delta_d))^{M-d}\mu^d(\mathrm{d}\delta_1, ..., \mathrm{d}\delta_d) = \int_\epsilon^1 (1-\alpha)^{M-d} F_{V_d}(\mathrm{d}\alpha)\end{aligned}$$

L'ultimo integrale può essere risolto sostituendo formalmente a $F_{V_d}(\mathrm{d}\alpha)$ il termine $d\alpha^{d-1}\mathrm{d}\alpha$ (in accordo col fatto che $F_{V_d} = \alpha^d$) e quindi integrando (iterativamente) per parti.

Un'espressione per il risultato è

$$1 - F_{V_M}(\epsilon) = \mu^M(V_M > \epsilon) = \sum_{i=0}^{d-1} \binom{M}{i} \epsilon^i (1-\epsilon)^{M-i} \tag{A.1}$$

A.2 Calcolo del valore atteso di V_M

Il calcolo di $\mathbf{E}[V_M]$ può avvenire secondo la formula (si veda sotto, la sezione A.2.1):

$$\mathbf{E}[V_M] = \int_0^1 1 - F_{V_M}(\epsilon)\mathrm{d}\epsilon$$

Per successive integrazioni per parti è semplice dimostrare la seguente proprietà della funzione Beta:

$$\beta(a-b+1,b+1)=\frac{1}{a+1}\frac{1}{\binom{a}{b}} \quad a,b\geq 0 \tag{A.2}$$

Ponendo $a=M$, $b=M-i$, si ricava dunque

$$\int_0^1 \epsilon^i(1-\epsilon)^{M-i}\mathrm{d}\epsilon = B(d+1,M-i+1)=\frac{1}{M+1}\frac{1}{\binom{M}{M-i}}$$

Con banali semplificazioni, si ha

$$\int_0^1 \sum_{i=0}^{d-1}\binom{M}{i}\varepsilon^i(1-\varepsilon)^{M-i}\mathrm{d}\varepsilon=\sum_{i=0}^{d-1}\binom{M}{i}B(d+1,M-i+1)=\frac{d}{M+1}$$

A.2.1 Nota sul calcolo del valore atteso

Dato lo spazio di probabilità $(\Delta^M,\mathfrak{F},\mu^M)$, il valore atteso di una variabile casuale $V:\Delta^M\longrightarrow[0,1]$ è definito come $\mathbf{E}[V]=\int_{\Delta^M}V(\delta_1,...,\delta_M)\mu^M(\mathrm{d}\delta_1,...,\mathrm{d}\delta_M)$. A V si può associare sempre la relativa funzione di distribuzione di probabilità $F_V=\mu^M(V(\delta_1,...,\delta_M)\leq\omega)$, $\omega\in\mathbb{R}$. Mostriamo che il valore atteso di V può essere calcolato mediante la F_V, nel seguente modo:

$$\mathbf{E}[V]=\int_{[0,1]}1-F_V(\omega)\mathrm{d}\omega$$

Se V ammette una funzione di densità, il risultato è immediato, integrando per parti. In generale, si può ragionare come segue.

Sia U la variabile casuale distribuita uniformemente su $[0,1]$ che ad un certo $\omega\in[0,1]$ associa il valore

$$1-F_V(\omega)=\mu^M(\{(\delta_1,...,\delta_M)\in\Delta^M\ t.c.\ V(\delta_1,...,\delta_M)>\omega\})$$

Si può scrivere

$$\begin{aligned}\mathbf{E}_\omega[U] &= \textstyle\int_{[0,1]}U(\omega)\mathrm{d}\omega=\int_{[0,1]}\left[\int_{\Delta^M}I_{V>\omega}(\delta_1,...,\delta_M,\omega)\mu^M(\mathrm{d}\delta_1,...,\mathrm{d}\delta_M)\right]\mathrm{d}\omega=\\ \textit{Fubini} &= \textstyle\int_{\Delta^M}\left[\int_{[0,1]}I_{V>\omega}(\delta_1,...,\delta_M,\omega)\mathrm{d}\omega\right]\mu^M(\mathrm{d}\delta_1,...,\mathrm{d}\delta_M)\end{aligned}$$

La funzione $I_{V>\omega}(\delta_1,...,\delta_M,\omega)$, per ogni $\overline{\delta}=(\delta_1,...,\delta_M)$, vale o zero o uno, e cioè (mettendo in evidenza solo la dipendenza da ω):

$$I_{V>\omega}(\omega)=\begin{cases}1 & \text{se } \omega<V(\overline{\delta})\\ 0 & \text{se } \omega\geq V(\overline{\delta})\end{cases}$$

Anche grazie all'ausilio della visualizzazione nella figura A.2.1, è evidente che l'integrale innestato nell'ultimo membro vale proprio $V(\overline{\delta})$ e, nel complesso, $\mathbf{E}[V]=\mathbf{E}[U]$.

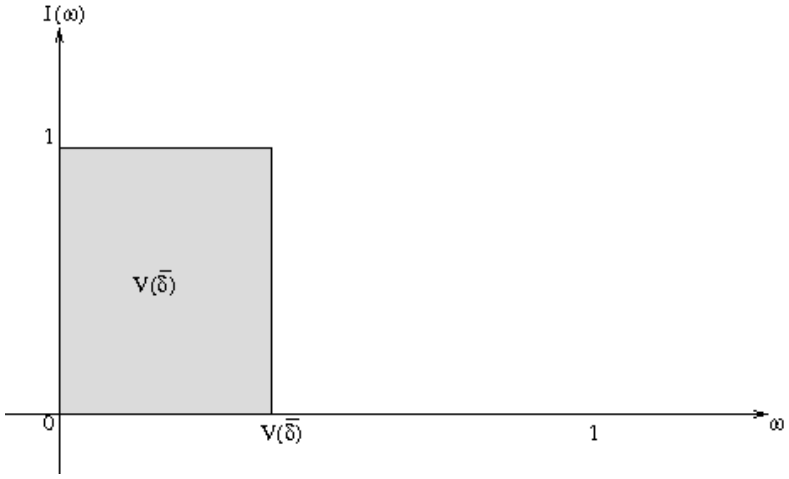


Figura A.1: Per ogni $\overline{\delta} = (\delta_1, ..., \delta_M)$, l'area sottesa dalla funzione indice $I_{V(\overline{\delta})>\omega}$ coincide col valore assunto da $V(\delta)$

A.3 Dimostrazione della proprietà "e" (sezione 6.1)

Sia C una funzione che seleziona tutti e soli i punti di supporto di qualsiasi M-upla ($M \geq d$) non contenente elementi ripetuti, allora vale la seguente proprietà

e) Se, per una data M-upla $\boldsymbol{\delta}_M = (\delta_1, ..., \delta_M)$ che non contiene elementi ripetuti, un qualsiasi $\widehat{\delta} \neq \delta_j$ (con $j = 1, ..., M$) non appartiene a $E_{\boldsymbol{\delta}_M}$, allora $C(\delta_1, ..., \delta_M) = C(\delta_1, ..., \delta_M, \widehat{\delta})$.

È utile ricordare anche che, se $C(\delta_1, ..., \delta_M) = C(\delta_1, ..., \delta_M, \widehat{\delta})$, allora $E_{(\delta_1,...,\delta_M)} = E_{(\delta_1,...,\delta_M,\widehat{\delta})}$

La dimostrazione può procedere per induzione.

Base induttiva Si prende $M = d$ come base. La dimostrazione per questo caso è banale: poiché $C(\delta_1, ..., \delta_M, \widehat{\delta})$ deve scegliere d (cioè M) punti e uno di essi, cioè $\widehat{\delta}$, non può essere essere scelto in quanto non di supporto per $(\delta_1, ..., \delta_M, \widehat{\delta})$, evidentemente $C(\delta_1, ..., \delta_M, \widehat{\delta}) = (\delta_1, ..., \delta_M) = C(\delta_1, ..., \delta_M)$

Ipotesi induttiva Supponiamo che per $M = m$ la proprietà valga. Dimostriamo che la proprietà allora vale anche per $M = m + 1$.

Per assurdo sia $\widehat{\delta} \notin E_{(\delta_1,...,\delta_m,,\delta_{m+1})}$ e $C(\delta_1,...,\delta_{m+1}) \neq C(\delta_1,...,\delta_{m+1},\widehat{\delta})$. Più esplicitamente:

$$C(\delta_1,...,\delta_{m+1},\widehat{\delta}) = (a_1,...,a_d)$$
$$C(\delta_1,...,\delta_{m+1}) = (b_1,...,b_d)$$

con almeno un a_i tale che $a_i \neq b_j$, $j = 1,...,d$. È senza perdita di generalità che possiamo ammettere che questo a_i sia δ_{m+1}.

Siccome, per quanto appena affermato, δ_{m+1} non è selezionato da $C(\delta_1,...,\delta_{m+1})$, esso non è di supporto per $(\delta_1,...,\delta_{m+1})$ e si ha che

$$\delta_{m+1} \notin E_{(\delta_1,...,\delta_m)} \tag{A.3}$$

Dall'ultima espressione, grazie all'ipotesi induttiva, si ricava che $E_{(\delta_1,...,\delta_m)} = E_{(\delta_1,...,\delta_m,\delta m+1)}$. Dato che, per ipotesi, $\widehat{\delta} \notin E_{(\delta_1,...,\delta_m,\delta_{m+1})}$, si ha dunque

$$\widehat{\delta} \notin E_{(\delta_1,...,\delta_m,\delta_{m+1})} = E_{(\delta_1,...,\delta_m)}$$

Ma applicando di nuovo l'ipotesi induttiva si può scrivere allora

$$E_{(\delta_1,...,\delta_m)} = E_{(\delta_1,...,\delta_m,\widehat{\delta})} \tag{A.4}$$

Mettendo insieme le equazioni A.3 e A.4 si ottiene

$$\delta_{m+1} \notin E_{(\delta_1,...,\delta_m,\widehat{\delta})}$$

nonostante che si fosse ipotizzato che δ_{m+1} fosse scelto da $C(\delta_1,...,\delta_m,\delta_{m+1},\widehat{\delta})$ e fosse quindi di supporto per $(\delta_1,...,\delta_m,\delta_{m+1},\widehat{\delta})$, il che, per definizione, comporterebbe

$$\delta_{m+1} \in E_{(\delta_1,...,\delta_m,\widehat{\delta})}$$

Quindi l'assunto è dimostrato per assurdo.

A.4 Dimostrazione del risultato negativo

La *realtà* risulta determinata una volta fissata la coppia (X,Y), la misura μ e la funzione $y(\cdot)$. Ci restringiamo a casi particolare di distribuzione μ: X sia distribuito uniformemente sull'insieme (discreto) $\{1,2,\ldots,k\}$, cosicché

$$\mathbb{P}(X=i) = \begin{cases} 1/k & \textit{se } i \in \{1,\ldots,k\} \\ 0 & \textit{altrimenti} \end{cases}$$

con $k \in \mathbb{N}$.

Fissato k, abbiamo quindi una distribuzione di probabilità per $X = \mathbb{R}$. Per quanto riguarda y, si osservi che è inutile precisare il valore assunto da ciascun $x \in \mathbb{R}$, in quanto la distribuzione di probabilità discreta rende possibili solo osservazioni in cui compaiano $x \in \mathbb{N}$ (possiamo assegnare cioè un valore arbitrario, per esempio sempre zero, a qualsiasi x non intero). Agli x interi si possono assegnare valori in tanti modi quanti sono i punti contenuti nell'intervallo $[0, 1]$ (cioè in $2^{\aleph_0}$ modi [1]). Questo fatto può essere intuitivamente afferrato considerando lo sviluppo binario di un qualsiasi numero reale $b \in [0, 1]$:

$$b \equiv 0, b_1 b_2 b_3 \ldots$$

Ciascun b_i ($i \geq 0$) può quindi essere associato all'intero i. Viceversa, fissate le etichette per ciascun numero intero maggiore di 1, resta univocamente determinato un solo sviluppo binario corrispondente ad un solo numero tra 0 e 1. Come lo sviluppo di 0 è quello che determina l'assegnamento dell'etichetta 0 a tutti gli interi, così si è assunto come sviluppo di 1 quello che determina l'assegnamento di 1 a tutti, cioè [2]

$$1 \equiv 0, 111111 \ldots$$

In realtà, quanto scritto non è del tutto esatto[3]. Infatti, esiste un insieme numerabile di numeri (razionali) che ammette due diversi sviluppi binari. Si pensi, ad esempio, a $\frac{1}{2}$ che può essere scritto come $0, 10000 \ldots$ ma anche $0, 01111 \ldots$. Come si appena è detto, l'insieme per cui la corrispondenza non è biunivoca è numerabile. Per questo motivo, si può dimostrare facilmente che è ancora possibile costruire una corrispondenza *biunivoca* tra sviluppi binari e numeri tra $[0, 1]$. Il ragionamento resta pertanto inalterato, considerando b_i come l'i-esimo numero dello sviluppo binario associato al numero b mediante una corrispondenza *biunivoca* opportunamente definita.

D'ora in poi, fissare un $b \in [0, 1]$ significherà quindi fissare y.

Fissato k, per ogni b si potrà quindi effettuare una multi-estrazione e ottenere così una certa $\widehat{y}_N$, cui resta associato un numero indice della sua

[1] $\aleph_0 = |\mathbb{N}|$. Si ricorda che tutti gli intervalli reali (aperti o chiusi che siano) sono tra loro equipotenti e hanno la medisima cardinalità dell'insieme delle parti di $\mathbb{N}$.

[2] Chiaramente, $\lim_{N\to\infty} \sum_{i=1}^{N} 2^{-i} = \lim_{N\to\infty} \dfrac{1 - (\frac{1}{2})^{N+1}}{1 - \frac{1}{2}} - 1 = 1$

[3] La dimostrazione in [36] non si cura di questo aspetto, dando per scontata la dimestichezza del lettore con l'aritmetica transfinita che effettivamente rende superflua, nella sostanza, la precisazione.

bontà: $\mathrm{PE}(\widehat{y}_N)$. Sempre considerando fissato b, si può, come nel caso della (1.1), calcolare il valore atteso, nel quale però, questa volta, sottolineiamo il fatto che nel calcolo entra in gioco b

$$\mathbf{E}\left[\mathrm{PE}(\widehat{y}_N)_{(b)}\right] = \int_{X^N} \mathrm{PE}(\widehat{y}_N)_{(x_1,b_1\ldots,x_N,b_N)}\mu^N(\mathrm{d}x_1,...,\mathrm{d}x_N)$$

In base al ragionamento iniziale, l'osservatore si trova in una certa realtà (caratterizzata da X, μ e y) ed effettua un certo numero di osservazioni indipendenti. Da tali osservazioni, l'algoritmo di classificazione ricava una funzione di decisione $\widehat{y}_N$. Ora, per dimostrare il risultato, dobbiamo dimostrare che è sempre possibile che esista una realtà "sfortunata", in cui l'algoritmo, qualunque sia, arrivi ad un "cattivo" risultato. Si è già detto come una misura della bontà della classificazione sia $\mathrm{PE}(\widehat{y}_N)$ e come essa dipenda da b.

Appare quindi abbastanza naturale, dopo aver limitato le distribuzione μ ad una tra quelle appartenenti ad un insieme circoscritto ma infinito di possibilità, agire su un altro grado di libertà. Ora assumeremo infatti che b non sia determinata una volta per tutte. Possiamo pensare di trovarci non più di fronte ad *una* realtà, bensì ad un insieme di mondi possibili (di etichettature possibili), tutti equiprobabili. In termini matematici, possiamo dire che le etichette dei numeri interi vengono assegnate in accordo col valore di una variabile casuale B, indipendente da X e dalle osservazioni $(x_1, \ldots, x_N)$ e la cui distribuzione di probabilità è uniforme su $[0, 1]$. A tal punto, ha senso chiedersi in media (cioè rispetto a tutti i mondi possibili) *come* opera un certo algoritmo di classificazione. Definiamo:

$$V_N(b) := \mathbf{E}\left[\mathrm{PE}(\widehat{y}_N)|B = b\right]$$

che, dal momento che b è il valore della variabile casuale B, è pure una variabile casuale di cui è possibile calcolare il valore atteso $\mathbf{E}[V_N(B)]$.
È elementare[4] notare che deve esistere una particolare etichettatura $\overline{b} \in B$

[4]

PROVA. Per assurdo, se fosse vero che

$$\forall \overline{b}, V_N(\overline{b}) < \mathbf{E}[V_N(B)]$$

allora

$$f(\overline{b}) := \int_{[0,1]} V_N(\overline{b}) - V_N(b)\mathrm{d}b < 0$$

ma si verifica immediatamente che $\mathbf{E}[f]$ deve essere zero, infatti

$$\int_{[0,1]} f(\overline{b})\mathrm{d}\overline{b} = \int_{[0,1]}\int_{[0,1]} V_N(\overline{b}) - V_N(b)\mathrm{d}b\mathrm{d}\overline{b} = 0$$

tale che

$$V_N(\bar{b}) \geq \mathbf{E}[V_N(B)]$$

Pertanto, la dimostrazione si concentrerà dapprima sulla caratterizzazione di $\mathbf{E}[V_N(B)]$.

Prima di procedere, si noti che un'estrazione di $b \in [0,1]$ con densità di probabilità uniforme equivale ad un'estrazione di infinite variabili casuali (quelle del suo sviluppo binario) B_i (con $i \in \mathbb{N}$) a valori in $\{0,1\}$, indipendenti l'una dall'altra e tali che $\mathbb{P}\{B_i = 1\} = \mathbb{P}\{B_i = 0\} = \frac{1}{2}$.

Scriviamo per esteso $\mathbf{E}[V_N(B)]$ come:

$$\begin{aligned}
\mathbf{E}[V_N(B)] &= \mathbf{E}_B\left[\mathbf{E}_{(X_1,...,X_N)}\left[\mathrm{PE}(\widehat{y}_N)|B=b\right]\right] = \\
&= \mathbf{E}_B\left[\mathbf{E}_{(X_1,...,X_N)}\left[\mathbf{E}_X[I_{\{\widehat{y}_N(X)_{(X_1,...,X_N,B_1,...,B_N)}\neq B_X\}}\right]\right] = \\
&= \mathbf{E}_{(X_1,...,X_N,X)}\left[\mathbf{E}_B\left[I_{\{\widehat{y}_N(X)_{(X_1,...,X_N,B_1,...,B_N)}\neq B_X\}}\right]\right] = \\
&= \mathbf{E}_{(X_1,...,X_N,X)}\left[\mathbb{P}\{\widehat{y}_N(X) \neq B_X|X_1,...,X_N,X\}\right]
\end{aligned}$$

Consideriamo l'insieme:

$$H = \{(x_1,...,x_n,x) \in X^{N+1} \ t.c. \ X = X_i \ per \ qualche \ i, \ 1 \leq i \leq N\}$$

A questo punto, si può scrivere $\mathbf{E}_{(X_1,...,X_N,X)}\left[\mathbb{P}\{\widehat{y}_N(X) \neq B_X|X_1,...,X_N,X\}\right]$ come

$$\begin{aligned}
&\mathbf{E}_{(X_1,...,X_N,X)\in H}\left[\mathbb{P}\{\widehat{y}_N(X) \neq B_X|X_1,...,X_N,X\}\right] + \\
&\quad \mathbf{E}_{(X_1,...,X_N,X)\in \overline{H}}\left[\mathbb{P}\{\widehat{y}_N(X) \neq B_X|X_1,...,X_N,X\}\right]
\end{aligned}$$

Concentriamoci sul secondo membro. La probabilità entro l'operatore di valore atteso non può che valere $\frac{1}{2}$: qualunque sia il valore assegnato dall'algoritmo ad un certo $x \neq x_i$ $(i = 1,...,N)$ sulla base dei campioni $(x_1,...,x_N)$, esso avrà una probabilità di $\frac{1}{2}$ di sbagliare, dal momento che l'effettiva etichetta di x è determinata dalla variabile casuale B_x, il cui valore può essere con la stessa probabilità 0 o 1, indipendentemente da quello dei campioni visionati dall'algoritmo. Portata fuori la costante dal valore atteso, resta da calcolare $\mathbb{P}\{\overline{H}\}$

$$\mathbb{P}\{\overline{H}\} = \mathbf{E}_X\left[\mathbb{P}\{ \ X \neq X_i \qquad \forall i, 1 \leq i \leq N|X\}\right] =$$

poiché $\int_{[0,1]}\int_{[0,1]} V_N(b)\mathrm{d}b\mathrm{d}\bar{b} = \int_{[0,1]}\int_{[0,1]} V_N(\bar{b})\mathrm{d}b\mathrm{d}\bar{b} = \mathbf{E}[V_N(B)]$

□

$$= \textit{indipendenza} = \left(1 - \frac{1}{k}\right)^N$$

Essendo certamente il primo membro non negativo, si può dunque scrivere

$$\mathbf{E}\left[V_N(B)\right] \geq \frac{1}{2}\left(1 - \frac{1}{k}\right)^N \tag{A.5}$$

Aumentando il valore di k (cambiando così la distribuzione di X) si può quindi ottenere un risultato mediamente (per tutte le possibili etichettature scelte con probabilità uniforme) "brutto" a piacere, in quanto una probabilità di sbagliare pari ad $\frac{1}{2}$ è quella che si ottiene classificando un campione mediante il lancio di una monetina. Inoltre, sappiamo che

$$\exists\, \bar{b} \text{ t.c. } V_N(\bar{b}) \geq \mathbf{E}\left[V_N(B)\right] \geq \frac{1}{2}\left(1 - \frac{1}{k}\right)^N \tag{A.6}$$

Abbiamo dunque dimostrato che $\forall \rho > 0$ piccolo a piacere $\exists \mu$ (cioè k) e y (cioè $\bar{b}$) t.c.

$$\underline{V_N(\bar{b}) \geq} \mathbf{E}\left[V_N(B)\right] \underline{\geq \frac{1}{2} - \rho} \geq \frac{1}{2}\left(1 - \frac{1}{k}\right)^N$$

Da qui al teorema 1 il passo è brevissimo. Supponiamo di fissare un ρ ($0 < \rho < \frac{1}{2}$) e di trovare pertanto un k opportuno che definisce μ e una funzione y che assegna etichette sulla base di $\bar{b}$ (d'ora in poi quindi la dipendenza dei calcoli da $\bar{b}$ sarà sottintesa) per cui valga

$$V_N(\bar{b}) \geq \frac{1}{2} - \rho$$

Definiamo gli insiemi:

$$L := \{(x_1, ..., x_n) \text{ t.c. } \mu(\widehat{y}_N(x) \neq y(x)) > \varepsilon\}$$

$$M := \{(x_1, ..., x_n) \text{ t.c. } \mu(\widehat{y}_N(x) \neq y(x)) \leq \varepsilon\}$$

Siccome $L \cup M = X^N$, dalla definizione di $V_N(\bar{b})$ si può scrivere:

$$\begin{aligned} V_N(\bar{b}) &= \int_L \mu(\widehat{y}_N(x) \neq y(x)|x_1, ..., x_n)\mu^N(\mathrm{d}x_1, ..., \mathrm{d}x_N) + \\ &\qquad + \int_M \mu(\widehat{y}_N(x) \neq y(x)|x_1, ..., x_n)\mu^N(\mathrm{d}x_1, ..., \mathrm{d}x_N) \leq \\ &\leq \int_L \mu^N(\mathrm{d}x_1, ..., \mathrm{d}x_N) + \varepsilon \end{aligned}$$

Il primo addendo di quest'ultima equazione è proprio $\mu^N(L)$ e quindi

$$\mu^N(\mu(\widehat{y}_N(x) \neq y(x)) > \varepsilon) =$$

$$\underline{\mu^N(\mathrm{PE}(\widehat{y}_N)_{(\bar{b})} > \varepsilon)) \geq} V_N(\bar{b}) - \varepsilon \underline{\geq \frac{1}{2} - \rho - \varepsilon} \quad \square$$

Un'ultima considerazione: il risultato è stato dimostrato facendo uso di una classe di distribuzione di probabilità su insiemi finiti e dimostrando che è sempre possibile trovare una distribuzione e una natura per i punti sufficientemente "cattive". Naturalmente, si possono trovare distribuzioni e nature egualmente malvagie definite su insiemi non numerabili, ricalcando le orme della dimostrazione precedente. Per esempio, per ragionare in analogia con la dimostrazione precedente, si potrebbe definire come classe di distribuzioni la seguente (per k naturale):

$$\mathbb{P}(X \leq x) = \begin{cases} x/k & \textit{se } x \in [0, k] \\ 0 & \textit{altrimenti} \end{cases}$$

Le etichette potrebbero essere ancora definite in $2^{\aleph_0}$ modi così:

$$y(x) = \begin{cases} b_1 & \textit{se } x \in (0, 1] \\ b_2 & \textit{se } x \in (1, 2] \\ \ldots & \\ b_j & \textit{se } x \in (j-1, j] \\ \ldots & \end{cases}$$

dove b_j è il j-esimo elemento dello sviluppo binario di un numero $b \in [0, 1]$.

Bibliografia

[1] Alan Agresti: *Categorical Data Analysis*. Wiley-IEEE, 2 edizione, 2003.

[2] A. Asuncion e D.J. Newman: *UCI Machine Learning Repository*, 2007. `www.ics.uci.edu/~mlearn/MLRepository.html`.

[3] Peter L. Bartlett e Marten H. Wegkamp: *Classification with a Reject Option using a Hinge Loss*. J. Mach. Learn. Res., 9:1823–1840, 2008, ISSN 1533-7928.

[4] Gregory Bateson: *Verso un'ecologia della mente*. Adephi, 2005.

[5] Andrew P. Bradley: *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. Pattern Recognition, 30(7):1145–1159, July 1997.

[6] L. Breiman, J. Friedman, R. Olshen e C. Stone: *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[7] Christopher J. C. Burges: *A tutorial on support vector machines for pattern recognition*. Data Mining and Knowledge Discovery, 2:121–167, 1998.

[8] Home page di Tom Bylander presso la University of Texas at San Antonio, `http://www.cs.utsa.edu/~bylander/`.

[9] M.C. Campi e G.C. Calafiore: *Uncertain convex programs: randomized solutions and confidence levels*. Math. Program., 102(1):25–46, 2005, ISSN 0025-5610.

[10] M.C. Campi e S. Garatti: *The exact feasibility of randomized solutions of uncertain convex programs*. SIAM Journal on Optimization, 19(3):1211–1230, 2008.

[11] Sezione dedicata a LIBSVM della home page di Chih-Jen Lin presso l'Università Nazionale di Taiwan, `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

[12] C. K. Chow: *An Optimum Character Recognition System Using Decision Functions*. IEEE Trans. Electronic Computers vol.6, pagine 247–254, 1957.

[13] C. K. Chow: *On optimum recognition error and reject tradeoff*. IEEE Transactions on Information Theory, 1970.

[14] L.P. Cordella, C. De Stefano, F. Tortorella e M. Vento: *A method for improving classification reliability of multilayer perceptrons*. IEEE Transactions on Neural Networks, 1995.

[15] C. Cortes e V.N. Vapnik: *Support Vector Networks*. Nel *Machine Learning*, pagine 273–297, 1995.

[16] Nello Cristianini e John Shawe-Taylor: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[17] Jon Dattorro: *Convex Optimization & Euclidean Distance Geometry*. Published by Meboo Publishing, USA, 2005.

[18] Luc Devroye, Laszlo Györfi e Gabor Lugosi: *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997.

[19] B. Dubuisson e M. Masson: *A statistical decision rule with incomplete knowledge about classes*. PR, 26(1):155–165, January 1993.

[20] André Elisseeff e Massimiliano Pontil: *Leave-one-out error and stability of learning algorithms with applications*. Advances in learning theory: Methods, models, and applications: IOS press., 2003.

[21] C. Ferri e J. Hernandez-Orallo: *Cautious classifers*. Proceedings of ROC Analysis in Artifcial Intelligence, 1st International Workshop (ROCAI-2004), pagine 27–36, 2004.

[22] C. Ferri, J. Hernández-orallo e M. A. Salido: *Volume under the ROC surface for multi-class problems*. Nel *Proc. of 14th European Conference on Machine Learning*, pagine 108–120, 2003.

[23] César Ferri e Peter Flach: *Delegating classifiers.* Nel *In Proc. 21st International Conference on Machine Learning*, pagine 27–36. Omnipress, 2004.

[24] R.A. Fisher: *The Use of Multiple Measurements in Taxonomic Problems.* Annals of Eugenics, v.7, pagine 179–188, 1936.

[25] Carl Frélicot: *On Unifying Probabilistic/Fuzzy and Possibilistic Rejection-Based Classifiers.* Nel *SSPR/SPR*, pagine 736–745, 1998.

[26] Giorgio Fumera e Fabio Roli: *Error rejection in linearly combined multiple classifiers.* Nel *In Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096*, pagine 329–338. Springer, 2001.

[27] Giorgio Fumera e Fabio Roli: *Support Vector Machines with Embedded Reject Option.* Nel *Proceedings of the Int. Workshop on Pattern Recognition with Support Vector Machines (SVM2002), Niagara Falls*, pagine 68–82. Springer, 2002.

[28] Giorgio Fumera, Fabio Roli e Giorgio Giacinto: *F.Roli: Multiple Reject Thresholds for Improving Classification Reliability.* Rapporto Tecnico, In Proceedings of Advances in Pattern Recognition: Joint IAPR International Workshops, 1999.

[29] Johannes Furnkranz: *Separate-and-conquer rule learning.* Artificial Intelligence Review, 13:3–54, 1999.

[30] Johannes Furnkranz: *Reducing misclassification costs.* Principles Data Mining and Knowledge Discovery, 4th European Conference, pagine 34–43, 2000.

[31] Y. Grandvalet, A. Rakotomamonjy, J. Keshet e S. Canu: *Support Vector Machines with a Reject Option.* Nel *NIPS*, 2008.

[32] Thien M. Ha: *An Optimum Decision Rule for Pattern Recognition.* Rapporto Tecnico, Advances in Pattern Recognition, SSPR'98 - SPR'98, 1995.

[33] R. Herbei e M. H. Wegkamp: *Classification with reject option.* Canadian Journal of Statistics, 34(4), pagine 709–721, 2006.

[34] Z. Hussain, S. Szedmak e J. Shawe-Taylor: *The linear programming set covering Machine.* In: PASCAL2004, URL: `http://eprints.pascal-network.org/archive/00001210/01/lp_scm.pdf`, 2004.

[35] Zakria Hussain, François Laviolette, Mario Marchand, John Shawe-Taylor, Spencer Charles Brubaker e Matthew D. Mullin: *Revised Loss Bounds for the Set Covering Machine and Sample-Compression Loss Bounds for Imbalanced Data.* J. Mach. Learn. Res., 8:2533–2549, 2007, ISSN 1533-7928.

[36] G. Lugosi L. Devroye, L. Györfi: *A Probabilistic Theory of Pattern Recognition, Springer, New York.* MIT Press, Cambridge, MA, 1996.

[37] Thomas C.W. Landgrebe, David M.J. Tax, Pavel Paclík e Robert P.W. Duin: *The interaction between classification and reject performance for distance-based reject-option classifiers*, 2005.

[38] H. Le Capitaine e C. Frélicot: *A class-selective rejection scheme based on blockwise similarity of typicality degrees.* To appear in Proc. 19th Int. Conf. on Pattern Recognition (ICPR), 2008.

[39] Charles X. Ling, Jin Huang e Harry Zhang: *AUC: a Statistically Consistent and more Discriminating Measure than Accuracy.* Nel *Proceedings of IJCAI 2003*, pagine 329–341. Morgan Kaufmann, 2003.

[40] Nick Littlestone e Manfred K. Warmuth: *Relating data compression and learnability.* Rapporto Tecnico, Technical Report, University of California, Santa Cruz., 1986.

[41] A. Luntz e V. Brailovsky: *On estimation of characters obtained in statistical procedure of recognition (in Russian).* Technicheskaya Kibernetica, 3, 1969.

[42] Mario Marchand, Mohak Shah, John Shawe-Taylor e Marina Sokolova: *The Set Covering Machine with Data-Dependent Half-Spaces.* Nel *Proceedings of the Twentieth International Conference on Machine Learning (ICML'2003)*, pagine 520–527. AAAI Press, 2003.

[43] Mario Marchand e John Shawe-Taylor: *The set covering machine.* Journal of Machine Learning Reasearch, 3, 2002.

[44] Pagina di MATLAB nel sito di The MathWorks Inc (3 Apple Hill Drive, Natick), software house produttrice: `http://www.mathworks.com/products/matlab/`.

[45] Warren Mcculloch e Walter Pitts: *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biology, 5(4):115–133, 1943.

[46] M.Grant e S.Boyd: *CVX: Matlab software for disciplined convex programming (web page and software)*, Febbraio 2009. `http://stanford.edu/~boyd/cvx`.

[47] Emanuel Parzen: *On Estimation of a Probability Density Function and Mode*. The Annals of Mathematical Statistics, 33(3):1065–1076, 1962.

[48] M. J. Pazzani, P. Murphy, K. Ali e D. Schulenburg: *Trading of coverage for accuracy in forecasts: Applications to clinical data analysis.* Proceedings of AAAI Symposium on AI in Medicine, 1994.

[49] J. Pearl: *Fusion, propagation, and structuring in belief networks.* Artificial Intelligence, 29(3):241–288, 1986.

[50] Tadeusz Pietraszek: *Optimizing Abstaining Classifiers using ROC Analysis*. Nel *Proceedings of the 22 nd International Conference on Machine Learning*, pagine 665–672. ACM Press, 2005.

[51] Platone: *Opere vol.2*. Laterza, 1967.

[52] J. R. Quinlan: *Induction of Decision Trees*. Nel *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[53] J.R. Quinlan: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[54] Ronald L. Rivest e Robert Sloan: *Learning complicated concepts reliably and usefully*. Nel *COLT '88: Proceedings of the first annual workshop on Computational learning theory*, pagine 69–79, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc., ISBN 0-55869-019-5.

[55] Stefan Rüping: *mySVM Manual*, 2000. `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[56] Home page di winSVM di Martin Sewell della University College London: `www.cs.ucl.ac.uk/staff/M.Sewell/winsvm/`.

[57] J. Shawe-Taylor e P. L. Bartlett: *Structural risk minimization over data-dependent hierarchies*. IEEE Trans. on Information Theory, 44(5):1926–1940, 1998.

[58] F. Tortorella: *An Optimal Reject Rule for Binary Classifiers*. Advances in Pattern Recognition Volume 1876/2000, pagine 611–620, 2000.

[59] L. G. Valiant: *A theory of the Learnable.* Communications of the ACM, pagine 1134–1142, 1984.

[60] V.N. Vapnik: *The Nature of Statistical Learning Theory (Information Science and Statistics).* Springer, November 1999.

[61] V.N. Vapnik e O. Chapelle: *Bounds on Error Expectation for Support Vector Machines.* Neural Computation, 12(9):2013–2036, 2000.

[62] V.N. Vapnik e A.Y. Chervonenkis: *On the uniform convergence of relative frequencies of events to their probabilities.* Theory Probab. Appl., 16:264–280, 1971.