Indice

Introduzione

L'apprendimento statistico e le reti neurali sono due argomenti apparentemente molto diversi tra loro, infatti il primo ha una formulazione più matematica, mentre il secondo è più orientato alle applicazioni.

In realtà approfondendo la loro conoscenza si può scoprire che essi hanno numerosi punti in comune, il più importante dei quali è che entrambi possono acquisire dei modelli mediante l'uso di esempi.

Tale metodologia di apprendimento può rivelarsi particolarmente utile per affrontare e risolvere dei problemi per i quali non sono già noti degli schemi matematici di risoluzione.

Anche se spesso non risulta evidente oggi l'apprendimento statistico è sempre più utilizzato in diversi settori, poiché permette una maggiore versatilità rispetto ad altri metodi, ad esempio è impiegato nei programmi che identificano e bloccano la posta indesiderata, oppure dai motori di ricerca per migliorare la precisione delle risposte fornite.

Le reti neurali hanno diverse altre proprietà che le rendono interessanti tra le quali l'elevato parallelismo dell'elaborazione oltre alla loro capacità di essere adattabili e parzialmente *fault tolerant*. Una rete neurale completamente configurata risulta inoltre estremamente rapida.

Il lavoro di questa tesi è rivolto a condurre una dettagliata analisi della teoria dell'apprendimento statistico e delle reti neurali riformulandone le espressioni in modo da uniformarle all'usuale formalismo della meccanica quantistica, introducendo anche dei concetti tipici di altri campi, in particolare relativi allo studio dei problemi inversi.

Per alcuni aspetti matematici si ricorrerà anche alla teoria dei Frame.

La tesi è sviluppata nei seguenti capitoli:

Il **primo capitolo** è dedicato all'apprendimento statistico detto supervisionato poiché richiede una fase di training in cui devono essere forniti degli esempi dall'esterno. Se ne discuteranno gli aspetti teorici e saranno analizzate le proprietà richieste alle soluzioni per poter essere ritenute soddisfacenti. Si presenteranno quindi gli spazi di Hilbert detti: *Reproducing Kernel Hilbert Spaces*, e le loro peculiarità; tali spazi funzionali sono alla base dell'intera teoria poiché è in essi che diventa possibile descrivere il processo di apprendimento. Si fornirà poi la dimostrazione del *Representer Theorem* che è di importanza centrale in quanto permette di sviluppare le funzioni da apprendere su una base formata dalle tante funzioni, dette di kernel, quanti sono gli esempi a disposizione per il training. Si analizzeranno poi gli schemi di regolarizzazione utilizzati per affrontare quei problemi che risultano mal posti secondo la definizione data da Hadamard, e si dimostrerà come tali approcci siano equivalenti. Alla fine del capitolo si esporrà una interpretazione bayesiana dell'algoritmo alla base dello statistical learning.

Il **secondo capitolo** è incentrato sui frame, si partirà alla loro definizione e si procederà poi alla descrizione dell'operatore di frame e del frame duale. Si descriverà quindi il metodo iterativo di inversione dell'operatore di frame detto di Richardson e se ne specificheranno le proprietà di convergenza in funzione delle caratteristiche della matrice da invertire. Si analizzerà poi la pseudo-inversa di Moore-Penrose che può essere di aiuto ogni volta che una matrice non risulta invertibile.

Il **terzo capitolo** è focalizzato sullo studio delle reti neurali, in particolare di quelle note come *feed-forward* in cui i segnali di input si propagano per tutta la rete, dalle unità di ingresso a quelle di output senza formare dei percorsi di *loop*. Saranno descritti i nodi di elaborazione, detti a volte neuroni artificiali, i tipi di codifiche più utilizzate per i dati, e gli algoritmi per il calcolo dell'output di tali nodi. Nelle reti neurali l'apprendimento avviene modificando le caratteristiche dei collegamenti tra le unità; ad ogni connessione è infatti associato un parametro reale, chiamato *weight* o peso, che ha la funzione di coefficiente moltiplicativo

per il valore dei dati trasmessi attraverso di essa. Tale schema di funzionamento è stato mutuato dallo studio del sistema nervoso, da cui è risultato che ciò che si apprende non modifica la struttura interna dei neuroni, ma le interfacce tra questi, chiamate sinapsi, il cui meccanismo chimico subisce delle alterazioni in seguito alla memorizzazione di informazioni; ad esempio l'energia richiesta per compiere una data azione diminuisce in base alla frequenza di utilizzo di un dato percorso neurale, per cui è come se si formassero dei circuiti privilegiati. Nonostante i tempi di risposta tipici dei neuroni dell'uomo siano dell'ordine del millisecondo il grande numero dei neuroni coinvolti in ogni azione, e quindi l'elevato parallelismo dell'elaborazione, oltre alla grande interconnessione tra gli stessi, consente di eseguire agevolmente delle operazioni che non sono ancora realizzabili con sistemi automatici computerizzati. Le reti neurali erano nate inizialmente proprio con lo scopo di emulare in qualche modo tale attività nervosa, il primo modello risale al lavoro di McCulloch e Pitts del 1943 [23], ed era piuttosto primitivo, tanto che, per esempio, non utilizzava ancora collegamenti con pesi variabili. La vera rivoluzione si è avuta con l'introduzione da parte di Rosenblatt nel 1958 [34] del perceptron che aveva indotto inizialmente un grande ottimismo sulle sue potenzialità, trasformatosi poi in disillusione a seguito della pubblicazione del libro di Minsky e Papert del 1969 [24] in cui se ne evidenziavano i problemi intrinseci, dovuti essenzialmente alla sua linearità e al fatto che non si erano ancora ben compresi i limiti del calcolo parallelo. La crisi che colpì il settore fu molto grave e l'interesse per le reti neurali poté riemergere solo nei primi anni ottanta grazie principalmente alla scoperta del metodo detto della *back-propagation* degli errori. All'interno del terzo capitolo si spiegherà come funziona un perceptron, anche fornendone una interpretazione geometrica, e si illustrerà la regola per l'aggiornamento dei pesi delle connessioni; si esporrà poi una dimostrazione del *teorema della convergenza* che assicura come ogni problema di discriminazione lineare risolubile possa essere eseguito da un perceptron con un numero limitato di aggiornamenti dei pesi. Si tratteranno quindi i limiti del perceptron fornendone delle dimostrazioni e

si passerà all'analisi delle reti neurali formate da più neuroni disposti in strati successivi, dette multi-layer che risolvono gran parte dei problemi suddetti. È stato infatti abbastanza recentemente dimostrato un teorema noto come *teorema di approssimazione universale*, secondo cui: per approssimare con precisione arbitraria una funzione che presenta un numero finito di discontinuità è sufficiente una rete neurale che abbia un solo layer di neuroni, detto hidden, tra il layer di input e quello di output, purché le funzioni di output di ogni singolo neurone, chiamate funzioni di attivazione, siano non lineari. La dimostrazione di tale teorema si può trovare con varie sfumature in: Hornik, Stinchcombe e White, 1989, [17]; Funahashi, 1989, [10]; Cybenko, 1989, [7]; Hartman, Keeler e Kowalski, 1990, [15]. Per l'aggiornamento dei pesi di tali reti neurali multi-layer si utilizza la già citata *back-propagation* degli errori e la regola che è nota come *delta rule generalizzata* entrambe spiegate ricorrendo anche a degli esempi. Nell'ultima parte del capitolo saranno evidenziati anche i principali problemi associati alla *back-propagation* e come sia possibile risolverne in parte alcuni modificando leggermente la regola di aggiornamento dei pesi; saranno infine brevemente illustrate le modalità di verifica dell'apprendimento di una rete neurale.

Il **quarto capitolo** è rivolto alla sintesi degli argomenti affrontati nei capitoli precedenti; si vedrà come essi possano essere collegati tra loro, ad esempio, nel caso in cui si debba affrontare il problema della modellizzazione di dati empirici. Sarà fornito inoltre un metodo per realizzare una rete neurale in grado di eseguire l'apprendimento di una funzione seguendo l'approccio dello statistical learning, dotando i singoli neuroni dell'hidden layer di funzioni di output di kernel. Si mostrerà poi come in tale schema i pesi delle connessioni possano essere calcolati utilizzando l'algoritmo iterativo di Richardson per l'inversione dell'operatore di frame, in modo da inserire tali valori direttamente in fase di preparazione della rete, evitando così le procedure di regolazione dei pesi. Sarà poi spiegato come si possano costruire gli operatori di kernel utilizzando i frame affinché tutta la procedura dello statistical learning risulti effettuabile con essi.

Capitolo 1

L'apprendimento statistico

La teoria dell'apprendimento è un campo di ricerca che si occupa dello studio formale dei sistemi che apprendono, ed è per sua natura interdisciplinare coinvolgendo la statistica, la fisica, la scienza dei computer, le scienze cognitive, l'ingegneria, la teoria dell'ottimizzazione e molte altre discipline scientifiche e matematiche; i risultati da essa conseguiti trovano oggi largo impiego anche in tanti altri settori, come la genetica, le telecomunicazioni e l'economia.

Usualmente i modelli di apprendimento sono raggruppati come segue:

L'*apprendimento supervisionato* consiste nel trovare una funzione che rappresenti nel modo migliore possibile la relazione tra i dati di input e di output del training set, che sia generalizzante, cioè sia in grado di fornire risposte corrette per input mai visti.

L'*apprendimento rinforzato* si ha quando il sistema può interagire con l'ambiente circostante modificandolo e ricevendone in cambio delle 'ricompense' o delle 'punizioni'. L'obiettivo è imparare ad agire in modo da massimizzare le future 'ricompense' o, equivalentemente, da minimizzare le future 'punizioni'. Tale tipo di apprendimento è strettamente collegato alla teoria delle decisioni (in statistica e nello studio del management) e alla teoria del controllo in ingegneria.

L'*apprendimento nella teoria dei giochi* generalizza l'apprendimento rinforzato poiché il sistema, chiamato a volte macchina o agente, riceve

ancora input, produce azioni e ottiene 'ricompense', ma l'ambiente può contenere altre macchine con le stesse proprietà. Lo scopo di tale modello è massimizzare le 'ricompense' complessive del sistema in oggetto, in funzione delle possibili azioni presenti e future delle altre macchine.

L'*apprendimento non supervisionato* dispone esclusivamente dei dati di input e cerca di scoprire in essi degli schemi sottostanti con cui effettuare raggruppamenti, estrarvi caratteristiche significative, o trovarne rappresentazioni utili, e tutto questo senza alcun aiuto esterno.

L'*apprendimento semi-supervisionato* è un modello più recente che si dimostra particolarmente utile quando si ha a disposizione una grande quantità di dati non classificati e le operazioni di classificazione risultano piuttosto impegnative, per cui si ha convenienza a classificarne solo una piccola parte; i dati etichettati sono utilizzati poi come punto di partenza per estendere la classificazione a tutti gli altri elementi, con procedimenti che sono tipici dell'approccio non supervisionato.

Nelle pagine seguenti sarà analizzato il metodo dell'apprendimento supervisionato, poiché tra tutti quelli disponibili oggi è quello che meglio si può applicare all'analisi dei dati sperimentali, che si ottengono per esempio in esperimenti controllati, dato che non si limita a semplici classificazioni ma può fornire delle vere e proprie funzioni che modellizzano i dati tenendo conto anche dei possibili errori associati agli stessi.

1.1 L'apprendimento supervisionato

Il problema dell'apprendimento mediante esempi, o supervisionato, sarà qui affrontato nella cornice dell'apprendimento statistico; con X si indicherà un dominio compatto in uno spazio euclideo e con Y un sottoinsieme chiuso.

I dati consistono in due insiemi di variabili casuali: un insieme chiamato di input ed indicato con $X \subseteq \mathbb{R}^d$, e uno di output $Y \subseteq \mathbb{R}^k$, usualmente con $k = 1$, collegati tramite una relazione probabilistica, quindi un elemento $x \in X$ non determina univocamente un elemento $y \in Y$, ma piuttosto una distribuzione di probabilità su Y, che può essere interpretata come il risultato della sovrapposizione di un rumore casuale al valore esatto y.

Tale distribuzione di probabilità non è conosciuta a priori, è definita sullo spazio prodotto $Z = X \times Y$, e sarà indicata con $\rho(x, y)$, inoltre occorre dire che essa non è l'obiettivo dell'apprendimento.

La $\rho(x, y)$ deve avere le proprietà di una misura di probabilità di Borel perciò, dato uno spazio topologico E separabile e localmente compatto, deve essere definita sulla più piccola σ-algebra fra quelle che contengono tutti gli aperti di E.

Generalmente i metodi di apprendimento si raggruppano in due categorie: si parla di *regressione* se i valori delle y sono reali, e di *classificazione* se invece essi appartengono ad un insieme discreto, in particolare per l'insieme $\{-1, 1\}$ si parla di *classificazione binaria*.

Il metodo standard per risolvere il problema dell'apprendimento statistico richiede la definizione di una *funzione di perdita* $V(f(x), y)$, che quantifichi quanto una funzione $f(x)$ presa a modello si discosti dai valori y del training set: $S_n \equiv \{z_i = (x_i, y_i) \in X \times Y\}_{i=1}^n$. Essa per esempio determina la sensibilità del risultato dell'apprendimento ad eventuali dati estranei al modello in esame, comunemente chiamati *outliers*, che possono pregiudicarne le caratteristiche di generalizzazione.

Nella sezione seguente verranno illustrate le proprietà di $V(f(x), y)$.

1.2 Le funzioni di perdita (loss functions)

Definizione: *Una* funzione di perdita *è una funzione non negativa di due variabili* $V : X \times Y \to \mathbb{R}^+$*; essa è indicata con* $V(f(x), y)$ *o con* $V(f, z)$*, essendo* $z = (x, y)$ *un elemento del training set formato da una coppia di dati rispettivamente di input ed output.*

1.2.1 Le funzioni di perdita per la regressione

1. *Square loss:* è la funzione più comune, il suo uso risale a Gauss e a Legendre, a volte è chiamata L_2 *loss*, ed è utilizzata nelle cosiddette *Regularization Networks (RN)*

$$V(f(x), y) := (f(x) - y)^2$$

2. *Absolute loss:* questa funzione è meno sensibile agli outliers della precedente, è chiamata anche L_1 *loss*, ed è alla base dello schema noto come *Support Vector Machines Regression (SVMR)*

$$V(f(x), y) := |f(x) - y|$$

3. *Huber loss:* come la L_1 è meno influenzata dai dati che possono essere considerati intrusi, in quanto errati o troppo disturbati dal rumore, ma a differenza di quest'ultima è ovunque differenziabile.

$$V(f(x), y) := \begin{cases} \frac{1}{4\varepsilon}(f(x) - y)^2 & \text{se } |f(x) - y| \leq 2\varepsilon \\ |f(x) - y| - \varepsilon & \text{altrimenti} \end{cases}$$

4. *Vapnik loss:* è come la L_1 meno sensibile agli outliers della L_2, ma può condurre a soluzioni dette *sparse*, poiché nella base su cui sono sviluppate hanno solo pochi termini non nulli.

$$V(f(x), y) = |f(x) - y|_\varepsilon := \begin{cases} 0 & \text{se } |f(x) - y| \leq \varepsilon \\ |f(x) - y| - \varepsilon & \text{altrimenti} \end{cases}$$

1.2.2 Le loss functions per la classificazione binaria

1. *Indicator loss:* è la funzione più intuitiva per la classificazione binaria, non è però molto utile per gli algoritmi statistici; indicando con Θ la funzione gradino di Heaviside risulta

$$V(f(x), y) = \Theta(-y\, f(x)) := \begin{cases} 1 & \text{se } y\, f(x) \leq 0 \\ 0 & \text{altrimenti} \end{cases}$$

2. *Hinge loss:* a differenza della precedente è una funzione convessa, perciò può essere usata per algoritmi pratici che conducano a soluzioni *sparse*; è detta anche *Soft Margin Loss*, ed è alla base del modello *Support Vector Machines Classification (SVMC)*

$$V(f(x), y) = (1 - y\, f(x))_+ := \begin{cases} 1 - y\, f(x) & \text{se } y\, f(x) \leq 1 \\ 0 & \text{altrimenti} \end{cases}$$

3. *Quadratic Hinge loss:* è simile alla *Hinge loss* ma la sua deviazione è quadratica

$$V(f(x), y) = \left[(1 - y\, f(x))_+\right]^2 := \begin{cases} (1 - y\, f(x))^2 & \text{se } y\, f(x) \leq 1 \\ 0 & \text{altrimenti} \end{cases}$$

4. *Logistic loss:* è anch'essa una funzione convessa, ma ha il vantaggio che ad essa può essere associato un modello probabilistico.

$$V(f(x), y) := \log\left(1 + \mathrm{e}^{-y\, f(x)}\right)$$

Nel caso in cui i dati debbano essere ripartiti in $M > 2$ classi, ogni potenziale errore di classificazione deve essere tenuto in considerazione e ciò conduce a delle loss functions che hanno la forma di matrici $M \times M$, con valori nulli sulla diagonale principale e positivi altrove; tale tipo di classificazione è tuttora oggetto di ricerche per determinare ad esempio come attribuire ad esse un appropriato livello di confidenza.

Ora che sono state presentate le funzioni di perdita è possibile definire con esse i vari tipi di errori che si incontrano nella teoria dell'apprendimento statistico.

1.3 Errori statistici e funzione obiettivo

Definizione: *L'*errore atteso*, detto anche* errore vero*, di una funzione f(x), data una funzione di perdita* $V(f(x), y)$*, ed una distribuzione di probabilità* $\rho(x, y)$ *definita su* $Z = X \times Y$ *è:*

$$\mathcal{E}(f) := \iint_{X,Y} V(f(x), y)\, \rho(x, y)\, \mathrm{d}x\, \mathrm{d}y$$

A volte l'errore atteso è indicato con $\mathcal{E}_\rho(f)$.
Definizione: *L'*errore empirico*, noto anche come* errore campionario*, di una funzione f(x), data una funzione di perdita* $V(f(x), y)$ *ed un training set* S_n *di n punti è:*

$$\mathcal{E}_S(f) := \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i)$$

L'errore empirico si trova indicato anche con $\mathcal{E}_{emp}(f)$ oppure $\mathcal{E}_z(f)$.

La definizione di *errore atteso* purtroppo si basa sulla distribuzione $\rho(x, y)$ che in generale non è nota, e quindi non è calcolabile direttamente, ma permette di definire teoricamente la funzione obiettivo f_0 dell'apprendimento:
Definizione: *La* funzione obiettivo f_0 *è quella funzione, appartenente ad una data classe di funzioni* $\mathcal{F}$*, che minimizza l'errore atteso:*

$$f_0(x) := \arg\min_{\mathcal{F}} \mathcal{E}(f)$$

1.3.1 Esempio di calcolo della funzione obiettivo

Come esempio si affronterà il caso molto comune della ricerca di una funzione di regressione mediante l'uso della funzione di perdita L_2.

Per i calcoli occorre ricordare la relazione: $\rho(x, y) = \rho(y|x)\rho(x)$, dove $\rho(x|y)$ è la probabilità condizionata di y dato x e $\rho(x)$ è la probabilità marginale di x, detta probabilità a priori di x; tale fattorizzazione è importante perché permette di guardare lo spazio del training set Z come il prodotto del dominio di input X per il dominio di output Y.

Definizione: *La* funzione di regressione $f_\rho : X \to Y$ *è data da:*

$$f_\rho(x) := \int_Y y \, \mathrm{d}\rho(y|x)$$

Tale funzione, che si supporrà sempre limitata, corrisponde $\forall x \in X$ al valor medio della coordinata y su $\{x\} \times Y$ oppure, detto in termini topologici, al valor medio di y sulla fibra di x.

Esprimendo l'errore atteso usando la funzione di perdita L_2 si ottiene:

$$\begin{aligned}
\mathcal{E}(f) &:= \iint_{X,Y} V(f(x), y)\, \rho(x,y)\, \mathrm{d}x\, \mathrm{d}y = \\
&= \iint_{X,Y} (f(x) - y)^2 \rho(x,y)\, \mathrm{d}x\, \mathrm{d}y = \\
&= \iint_{X,Y} \left[(f(x) - f_\rho(x)) + (f_\rho(x) - y)\right]^2 \rho(x,y)\, \mathrm{d}x\, \mathrm{d}y = \\
&= \int_X (f(x) - f_\rho(x))^2 \rho(x)\, \mathrm{d}x + \int_X \int_Y (f_\rho(x) - y)^2 \rho(x,y)\, \mathrm{d}y\, \mathrm{d}x + \\
&\quad + 2 \int_X \int_Y (f(x) - f_\rho(x))(f_\rho(x) - y)\, \rho(x,y)\, \mathrm{d}y\, \mathrm{d}x = \\
&= \int_X (f(x) - f_\rho(x))^2 \rho(x)\, \mathrm{d}x + \int_X \int_Y (f_\rho(x) - y)^2 \rho(x,y)\, \mathrm{d}y\, \mathrm{d}x = \\
&= \int_X (f(x) - f_\rho(x))^2 \rho(x)\, \mathrm{d}x + \mathcal{E}(f_\rho)
\end{aligned}$$

Analizzando l'ultima espressione si può notare come il primo integrale corrisponda ad una media su X dell'errore dovuto all'uso di $f(x)$ come modello per la funzione $f_\rho(x)$; tale termine può essere solo positivo o nullo, poiché le funzioni di perdita forniscono per definizione risultati reali non negativi. Il secondo termine invece è indipendente da f, per cui la funzione che fornisce il più piccolo errore possibile tra tutte le $f : X \to Y$ è la seguente:

$$f_0 := \arg\min_{\mathcal{F}} \mathcal{E}(f) = f_\rho(x)$$

L'obiettivo dell'apprendimento si riduce quindi a trovare una buona approssimazione di $f_\rho(x)$ a partire dai campioni del training set.

L'errore $\mathcal{E}(f_\rho)$, che talvolta è chiamato σ_ρ^2, è dovuto esclusivamente alla distribuzione ρ, e rappresenta un limite inferiore all'errore atteso $\mathcal{E}$; esso dà una misura di quanto ben condizionata sia la ρ stessa, analogamente alla nozione di numero di condizionamento dell'algebra lineare.

È opportuno evidenziare inoltre che, mentre $\rho(x, y)$ e f_ρ sono quasi sempre sconosciute, la distribuzione $\rho(x)$ in alcuni casi è nota, e può essere anche una semplice misure di Lebesgue su X, ereditata dagli spazi Euclidei.

1.4 La mappa di learning

Definizione: *Una mappa di learning $\mathcal{A}$ è una mappa da un training set S ad una funzione f_S nello spazio di ipotesi $\mathcal{H}$, delle funzioni tra cui l'algoritmo $\mathcal{A}$ effettua la ricerca:*

$$\mathcal{A} : S \to f_S \in \mathcal{H}$$

1.5 La minimizzazione del rischio empirico

Definizione: *Dato uno spazio di funzioni di ipotesi $\mathcal{H}$, una funzione $f_S \in \mathcal{H}$ è una* minimizzatrice del rischio empirico (ERM) *se:*

$$f_S \in \arg\min_{f \in \mathcal{H}} \mathcal{E}_S(f) := \arg\min_{f \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) \right]$$

Tale minimo non è detto che esista, e in tal caso si deve ricorrere ad una variante dell'algoritmo precedente chiamata *quasi ERM*, basata sull'infimum dell'errore empirico, che è un valore sempre determinabile:
Definizione: *Dato uno spazio di funzioni di ipotesi $\mathcal{H}$, una funzione f_S^ε di $\mathcal{H}$ è una ε-*minimizzatrice del rischio empirico *se $\forall \varepsilon > 0$:*

$$\mathcal{E}_S(f_S^\varepsilon) \leq \inf_{f \in \mathcal{H}} \mathcal{E}_S(f) + \varepsilon$$

Poiché per l'apprendimento supervisionato si hanno a disposizione solo i dati contenuti nel training set, è parso naturale cercare di ricavare

direttamente da essi le funzioni che meglio li approssimassero, mediante la minimizzazione dell'errore empirico, il cui calcolo si basa solo su valori noti. Per evitare però che tale impostazione conduca ad una banale interpolazione, alle funzioni trovate si devono imporre precisi vincoli, ognuno dei quali garantisce delle proprietà diverse alla soluzione.

L'elenco delle caratteristiche che deve possedere una soluzione per essere accettabile è il seguente:

1.5.1 Proprietà desiderate dell'algoritmo ERM

- *generalizzazione*
- *consistenza*
- *essere ben posto* (well-posedness), che include la *stabilità*.

Il significato di ciascuna delle proprietà suddette e le condizioni che le determinano saranno esposte nelle sezioni successive, insieme alle nozioni matematiche necessarie per la loro completa definizione.

1.6 La convergenza in probabilità

Poiché sarà utilizzata la convergenza in probabilità, se ne riportano qui le principali definizioni; con $\mathbb{P}\{x\}$ si indicherà la probabilità di x.
Definizione: *Sia $\{x_n\}$ una sequenza di variabili casuali. Si dice che:*

$$\lim_{n\to\infty} x_n = x \quad \text{in probabilità}$$

se:

$$\forall \varepsilon > 0 \quad \lim_{n\to\infty} \mathbb{P}\{\|x_n - x\| \geq \varepsilon\} = 0$$

Una diversa definizione che è utilizzata frequentemente è la seguente:

$$\forall n \in \mathbb{N} \quad \exists \varepsilon_n, \delta_n \geq 0 \mid \mathbb{P}\{\|x_n - x\| \geq \varepsilon_n\} \leq \delta_n \ \wedge \ \lim_{n\to\infty} \varepsilon_n = \lim_{n\to\infty} \delta_n = 0$$

1.7 La proprietà di generalizzazione

La definizione classica di generalizzazione è la seguente: *un algoritmo si dice che* generalizza *se fornisce output corretti per input mai visti;* purtroppo però essa non è stata matematicamente ben definita, e per tradurla in termini più formali si è ricorsi ad un'altra definizione:
Definizione: *Un algoritmo per trovare* f_S generalizza *se l'errore atteso è 'vicino' all'errore empirico, cioè* $|\mathcal{E}(f_S) - \mathcal{E}_S(f_S)|$ *è 'piccolo'.*

Da questa definizione ancora piuttosto imprecisa si deduce che: se la condizione di generalizzazione è soddisfatta e l'errore empirico è 'piccolo', allora l'algoritmo potrà fornire risposte accurate per dati sconosciuti, ma se l'errore empirico fosse 'grande', allora la sola condizione di generalizzazione non sarebbe sufficiente per vincolare adeguatamente il comportamento della f_S, con la conseguenza che essa risulterebbe pressoché inutilizzabile.

Al fine di evitare termini come *'piccolo'* e *'grande'*, troppo vaghi e del tutto relativi, in lavori più recenti è stata presentata una nuova definizione, molto più precisa e restrittiva delle precedenti, purtroppo però non del tutto equivalente a quella classica:
Definizione: *Dato un training set* S_n *costituito da n esempi, cioè da n coppie di elementi di input-output, un algoritmo* generalizza *se:*

$$\lim_{n\to\infty} |\mathcal{E}(f_{S_n}) - \mathcal{E}_S(f_{S_n})| = 0 \quad \text{in probabilità}$$

Se si interpreta la $\rho(x, y)$ come una misura μ sullo spazio $Z = X \times Y$, allora si può ulteriormente generalizzare la definizione suddetta:
Definizione: *Un mappa di apprendimento è* generalizzante *indipendentemente dalla distribuzione se:*

$$\forall \varepsilon > 0 \quad \lim_{n\to\infty} \sup_{\mu} \mathbb{P}\Big\{ |\mathcal{E}(f_{S_n}) - \mathcal{E}_S(f_{S_n})| > \varepsilon \Big\} = 0$$

1.8 La proprietà di consistenza

Si dice che un algoritmo soddisfa la proprietà di consistenza *se aumentando il numero di esempi esso tende a trovare la miglior funzione all'interno*

dello spazio di funzioni di ipotesi dato.
Definizione: *Un mappa di learning è* universalmente consistente *se:*

$$\forall \varepsilon > 0 \quad \lim_{n \to \infty} \sup_{\mu} \mathbb{P} \left\{ \mathcal{E}(f_{S_n}) > \inf_{f \in \mathcal{H}} \mathcal{E}(f) + \varepsilon \right\} = 0$$

Consistenza universale significa che essa vale con riferimento all'insieme di tutte le misure su Z, comunque la consistenza potrebbe essere definita anche con riferimento ad una specifica misura.

Per l'algoritmo ERM di minimizzazione del rischio empirico la consistenza è equivalente alla generalizzazione per cui in tale caso si parla spesso di consistenza intendendo generalizzazione e consistenza.

1.9 Problemi inversi e condizioni di Hadamard

Il problema dell'apprendimento può essere visto come appartenente alla classe dei *problemi inversi*, infatti con esso si cerca di risalire dai dati ad un modello. I problemi inversi sono solitamente mal posti poiché non soddisfano tutte le condizioni date da Hadamard per la definizione di un *problema ben posto*:
Definizione: *Un problema inverso è* ben posto *se valgono le condizioni:*

1. *Esistenza:* La soluzione esiste se: $\forall y \in Y \ \exists x \in X \mid F(x) = y$
2. *Unicità:* La soluzione è univocamente definita se: $\forall y \in Y \ \exists! x \in X$
3. *Stabilità:* La soluzione x della funzione inversa è continua rispetto a y. Tale condizione sarebbe opportuno però che si rendesse più restrittiva, imponendo l'uniforme continuità, come sostenuto da Liapunov; nel caso in cui la funzione inversa fosse definita su un compatto tuttavia ciò non sarebbe necessario infatti, per il teorema di Cantor-Heine, la continuità implicherebbe l'uniforme continuità.

   Definizione: *Una funzione inversa è* uniformemente continua *se:*

$$\forall \varepsilon > 0 \quad \exists \delta_\varepsilon > 0 \mid \forall y_1, y_2 \in Y, \ \|y_1 - y_2\| \leq \delta_\varepsilon \ \Rightarrow$$
$$\Rightarrow \quad \|F^{-1}(y_1) - F^{-1}(y_2)\| = \|x_1 - x_2\| \leq \varepsilon$$

Delle tre condizioni quella più importante, poiché più difficile da soddisfare, è la stabilità, tanto che in letteratura spesso, quando si fa riferimento ad un problema mal posto, in realtà si vuole intendere un problema che non è stabile.

Se F fosse un operatore lineare compatto su uno spazio di Hilbert di dimensione infinita, affinché si possa parlare di problema *ben posto* occorre che i valori singolari σ_l, risultanti dal processo di decomposizione in valori singolari (SVD), soddisfino la condizione: $\lim_{l \to \infty} \sigma_l = 0$.

Solitamente si effettua una classificazione del grado di *ill-posedness*, in modo che ad un grado più alto corrisponda una maggiore difficoltà di risoluzione del problema inverso; per quanto riguarda i valori singolari σ_l di cui sopra, se essi decrescono per $l \to \infty$ secondo una legge di potenza del tipo: $\sigma_l \sim l^{-k}$, come accade per le equazioni integrali di Volterra, allora si parla di problema *moderatamente mal posto*; se $k = \frac{1}{2}$, come nel caso della tomografia per immagini tradizionale, il problema è detto *debolmente mal posto*.

Per spazi di dimensione finita, un parametro importante di cui si deve tener conto è il *numero di condizionamento*, che è definito come segue:

$$\kappa(F) := \|F\| \, \|F^{-1}\|$$

Il suo utilizzo sarà spiegato successivamente, quando saranno presentati degli esempi di algoritmi di apprendimento.

1.10 Regolarizzazione dei problemi mal posti

Per regolarizzare un problema mal posto è necessario ripristinare le condizioni di esistenza, unicità e stabilità della soluzione, spesso mediante un opportuna scelta dello spazio delle funzioni di ipotesi $\mathcal{H}$ in cui si opera.

· Se il problema è l'*esistenza* allora si può intervenire ridefinendo il problema a partire dagli algoritmi usati; in molti casi infatti, come per il *quasi ERM*, l'esistenza di una soluzione è garantita proprio dalla definizione usata.

· Se non è soddisfatta la condizione di *unicità* invece si può agire scegliendo a caso un rappresentante all'interno di ogni classe di equivalenza, per esempio, riferendosi al caso dell'ERM, si possono considerare equivalenti le funzioni che hanno lo stesso errore empirico.

· Risolvere un problema di *stabilità* della soluzione è molto più complesso, e gli approcci standard utilizzano il *principio di regolarizzazione*, che si traduce in vincoli sullo spazio delle ipotesi $\mathcal{H}$.

Nella sezione seguente verranno presentati i principali metodi di regolarizzazione che sono utilizzati per trattare i problemi mal posti.

1.11 I metodi di regolarizzazione per l'ERM

1. *Tikhonov:* si vincola indirettamente lo spazio delle ipotesi aggiungendo un termine di penalità:

$$\min_{f\in\mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \gamma\, \Omega(f) \right]$$

2. *Ivanov:* si vincola direttamente lo spazio delle ipotesi

$$\min_{f\in\mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) \right] \quad \text{con il vincolo:} \quad \Omega(f) \leq \nu$$

3. *Phillips:* si vincola direttamente lo spazio delle ipotesi

$$\min_{f\in\mathcal{H}} \Omega(f) \quad \text{con il vincolo:} \quad \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) \right] \leq \tau$$

$\Omega(f)$ è un *funzionale di regolarizzazione*, detto anche di *smoothness*, ed è tipicamente una norma nello spazio di funzioni $\mathcal{H}$. Le proprietà di tale spazio e della norma in esso definita saranno presentate nella prossima sezione.

$\gamma, \nu, \tau \in \mathbb{R}^+$ sono parametri di regolarizzazione che controllano il trade-off tra il fitting dei dati e i vincoli sullo spazio delle ipotesi $\mathcal{H}$.

1.12 Reproducing Kernel Hilbert Spaces

Definizione: *Un* funzionale lineare di valutazione $L_t[f]$ *valuta ogni funzione f, di uno spazio di Hilbert di funzioni $\mathcal{H}$, in un punto t:*

$$\forall f \in \mathcal{H} \quad L_t[f] := f(t)$$

la proprietà di linearità *implica:*

1. $L_t[f+g] = f(t) + g(t) \qquad \forall f, g \in \mathcal{H} \qquad$ (additività)
2. $L_t[a\, f] = a\, f(t) \qquad \forall f \in \mathcal{H} \ \wedge \ \forall a \in$ c-numeri $\qquad$ (omogeneità)

Definizione: *Un funzionale lineare di valutazione $L_t[f]$ è* limitato *se e solo se, indicando con $\|f\|_{\mathcal{H}}$ la norma di f in $\mathcal{H}$, si ha:*

$$\forall t \in X \quad \exists M \in \mathbb{R}^+ \ | \ \forall f \in \mathcal{H} \quad |L_t[f]| = |f(t)| \leq M \|f\|_{\mathcal{H}}$$

Utilizzando le due definizioni suddette è possibile definire gli spazi detti Reproducing Kernel Hilbert Spaces (RKHS):
Definizione RKHS: *Se uno spazio di Hilbert ha un funzionale lineare di valutazione limitato, $L_t[f]$, allora esso è detto* Reproducing Kernel Hilbert Space (RKHS)*, e si indica usualmente con $\mathcal{H}_K$.*

Se $\mathcal{H}_K$ è un RKHS allora per il teorema di rappresentazione di Riesz-Fréchet, vale quanto segue:

$$\forall t \in X \quad \exists! K_t \in \mathcal{H}_K \ | \ \forall f \in \mathcal{H}_K \quad L_t[f] = \langle K_t | f \rangle = f(t)$$

L'elemento K_t è chiamato rappresentante della valutazione in t.

1.13 La funzione reproducing kernel (rk)

Definizione rk: *Una funzione* K reproducing kernel (rk) *deve essere una funzione a valori reali di due variabili* $s, t \in X$*, limitata, simmetrica e definita positiva:*

1. $K(s,t) : X \times X \to \mathbb{R}$
2. $K(s,t) = K(t,s)$
3. $\forall a_1, a_2, \ldots, a_n \in \mathbb{R} \ \wedge \ \forall t_1, t_2, \ldots, t_n \in X \quad \sum_{i,j=1}^{n} a_i \, a_j \, K(t_i, t_j) \geq 0$

Se nell'ultima condizione vale la disuguaglianza stretta allora $K(s,t)$ è detta *definita strettamente positiva.*

Molte volte, invece di caratterizzare la forma quadratica della funzione $K(s,t)$, si preferisce analizzare la matrice $\mathcal{K}(t)_{ij} = K(t_i, t_j)$ chiamata *matrice di Gram del kernel K rispetto al vettore t*, ed utilizzare le corrispondenti nozioni di *matrice semi-definita positiva* e di *matrice definita positiva*, in tal caso occorre tener presente la differente nomenclatura.

Teorema. *Per ogni Reproducing Kernel Hilbert Space (RKHS) esiste un'unica funzione reproducing kernel (rk) e viceversa.*

Dimostrazione. $RKHS \Rightarrow rk$:
Se $\mathcal{H}_K$ è un RKHS allora per definizione si ha che: $\forall t \in X, \ \exists! K_t \in \mathcal{H}_K$ detto rappresentante della valutazione in t; mediante tali rappresentanti si può definire in modo univoco la funzione reproducing kernel (rk):

$$K(s,t) := \langle K_s | K_t \rangle \in \mathbb{R}$$

Tale funzione di due variabili è a valori reali per definizione, ed è simmetrica, per cui risulta che:

$$K(t,s) := \langle K_t | K_s \rangle = \langle K_s | K_t \rangle = K(s,t)$$

Rimane ora da dimostrare solo che $K(s,t)$ è definita positiva, e ciò può essere fatto tenendo conto che $\forall K_t \in \mathcal{H}_K$ e $\forall a \in \mathbb{R}$ si ha:

$$0 \leq \left\| \sum_i a_i \, K_{t_i} \right\|^2 = \sum_{i,j} a_i \, a_j \left\langle K_{t_i} \,\middle|\, K_{t_j} \right\rangle := \sum_{i,j} a_i \, a_j \; K(t_i, t_j)$$

tale espressione coincide proprio con la definizione di operatore positivo, perciò $K(s,t)$ ha tutte e tre le proprietà richieste per essere una funzione reproducing Kernel, ed è unica per costruzione; i valori che essa assume inoltre non varierebbero anche se si cambiasse rappresentazione per i ket $|K_t\rangle$ o si introducessero dei fattori di fase globali, poiché i prodotti scalari non cambiano per tali operazioni. □

Dimostrazione. $rk \Rightarrow RKHS$:
La prova sarà per costruzione, cioè data una funzione rk: $K(\cdot,\cdot)$ reproducing kernel, si procederà alla costruzione di $\mathcal{H}_K$.
Se si definisce la funzione a valori reali:

$$\forall t \in X \quad K_t(\cdot) := K(t,\cdot) \in \mathbb{R}$$

lo spazio di funzioni generato dalle K_{t_i} risulta:

$$\mathcal{H} = \left\{ f \;\middle|\; f = \sum_{i \in \mathbb{N}} a_i \, K_{t_i} \quad \text{con:} \quad a_i \in \mathbb{R}, \;\; t_i \in X \right\}$$

il prodotto scalare in $\mathcal{H}$ può essere definito come segue:

$$\left\langle \sum_i a_i \, K_{t_i} \,\middle|\, \sum_j a_j \, K_{t_j} \right\rangle = \sum_{i,j} a_i \, a_j \left\langle K_{t_i} \,\middle|\, K_{t_j} \right\rangle := \sum_{i,j} a_i \, a_j \; K(t_i, t_j)$$

dato che $K(\cdot,\cdot)$ è definita positiva, ne consegue che il prodotto scalare è ben definito. A questo punto si può definire in modo univoco il seguente funzionale di valutazione:

$$\forall t \in X \;\; \wedge \;\; \forall f \in \mathcal{H} \quad L_t[f] := \langle K_t | f \rangle := f(t)$$

Tale funzionale di valutazione è lineare e limitato, a causa delle proprietà dei prodotti scalari tra elementi di $\mathcal{H}$; ora per dimostrare che lo spazio $\mathcal{H}$ è

un RKHS occorre solo dimostrare che esso è completo, cioè ogni sequenza di Cauchy in tale spazio converge; grazie al teorema di Cauchy-Schwartz si ha:

$$\forall t \in X \quad |f_n(t) - f(t)| = |\langle K_t, f_n - f \rangle| \leq \|f_n - f\| \, \|K_t\|$$

da cui si deduce che la convergenza in norma di $f_n \to f$ implica la convergenza puntuale di $f_n(t) \to f(t)$. Poiché la convergenza in norma è associata alla convergenza nello spazio delle funzioni reproducing kernel (rk), si ha che la completezza di quest'ultimo spazio implica la completezza di $\mathcal{H}$, quindi esso è un Reproducing Kernel Hilbert Space (RKHS), e può essere indicato con $\mathcal{H}_K$. □

1.14 Il Kernel di Mercer

Il kernel di Mercer è presente nella maggior parte delle formulazioni che riguardano le Support Vector Machines (SVMs) e le Kernel Machines; esso può essere visto come una specializzazione della trattazione precedente.

Teorema: *Dati gli autovalori λ_i e le autofunzioni ϕ_i dell'equazione integrale determinata dal kernel di Mercer $K : X \times X \to \mathbb{R}$, continuo, simmetrico e definito positivo:*

$$\int_X K(s,t)\, \phi(t)\, \mathrm{d}t = \lambda\, \phi(s)$$

il kernel ha la seguente espansione:

$$K(s,t) = \sum_i \lambda_i\, \phi_i(s)\, \phi_i(t)$$

la cui convergenza è nella norma dello spazio di Hilbert $\mathcal{L}^2(X)$, delle funzioni a quadrato sommabili su X.

La prova di tale teorema è data in [16] per $X = [0, 1]$, utilizzando una misura di Lebesgue su $\mathbb{R}$, ma la sua validità è generale.

Le autofunzioni ϕ_i costituiscono una base su cui si possono sviluppare tutte le funzioni che appartengono allo spazio di Hilbert $\mathcal{H}$, per esempio

si può scrivere:

$$f(s) = \sum_i a_i \, \phi_i(s)$$

Definizione: *Il* prodotto scalare *nello spazio di Hilbert* $\mathcal{H}_K$ *associato al kernel di Mercer, indicando con* λ_i *gli autovalori relativi alle autofunzioni* ϕ_i*, è il seguente:*

$$< f|g >= \left\langle \sum_i a_i \, \phi_i \,\middle|\, \sum_j b_j \, \phi_j \right\rangle_{\mathcal{H}_K} := \sum_i \frac{a_i \, b_i}{\lambda_i}$$

Ora che è stato definito il prodotto scalare, è possibile dimostrare la ***proprietà di riproduzione***, che dà il nome allo spazio RKHS:

$$\langle K_x | f \rangle_{\mathcal{H}_K} := \langle K(x,\cdot) | f(\cdot) \rangle_{\mathcal{H}_K} = \langle f(\cdot) | K(\cdot,x) \rangle_{\mathcal{H}_K} := \langle f | K_x \rangle_{\mathcal{H}_K} = f(x)$$

nella dimostrazione le funzioni saranno sviluppate su una base formata dalle autofunzioni suddette, rinormalizzate però affinché gli autovalori non negativi $\sqrt{\lambda_i}$ siano inglobati in esse, per cui $\Phi_i(\cdot) := \sqrt{\lambda_i} \, \phi_i(\cdot)$, in modo da rendere la formula del prodotto scalare più simmetrica:

$$\langle f | K_x \rangle_{\mathcal{H}_K} = \left\langle \sum_i a_i \, \Phi_i \,\middle|\, \sum_j \Phi_j \, \Phi_j(x) \right\rangle_{\mathcal{H}_K} =$$

$$= \sum_j \left\langle \sum_i a_i \, \Phi_i \,\middle|\, \Phi_j \right\rangle_{\mathcal{H}_K} \Phi_j(x) \; = \; \sum_j a_j \, \Phi_j(x) = f(x)$$

Dato il prodotto scalare, la norma risulta automaticamente definita, quindi si può procedere alla definizione delle spazio RKHS.

Definizione: *Lo* spazio di Hilbert RKHS *è lo spazio di funzioni generato dalle possibili combinazioni lineari delle autofunzioni dell'operatore integrale definito mediante il kernel di Mercer:*

$$\mathcal{H}_K = \left\{ f \;\mid\; f(s) = \sum_{i \in \mathbb{N}} a_i \, \phi_i(s) \;\; \wedge \;\; \|f\|_{\mathcal{H}_K} < \infty \quad \text{con:} \;\; a_i \in \mathbb{R}, \;\; s \in X \right\}$$

Si noti che lo spazio appena definito, in alcuni testi, è chiamato anche: *feature space* indotto dal kernel K, seguendo una consuetudine tipica della letteratura relativa a SVMs e Kernel Machines.

In tale contesto le autofunzioni del reproducing kernel sono considerate come una mappa dagli $x \in X \subseteq \mathbb{R}^d$ al *feature space*, avente dimensione pari al numero delle autofunzioni associate agli autovalori non nulli:

$$x \mapsto \Phi(x) := \{\sqrt{\lambda_1}\,\phi_1(x), \sqrt{\lambda_2}\,\phi_2(x), \ldots, \sqrt{\lambda_k}\,\phi_k(x)\}$$

inoltre, poiché è ivi soltanto richiesto che le autofunzioni siano linearmente indipendenti, esse potrebbero anche non essere ortogonali fra loro.

Dati due punti s e t mappati nello spazio delle caratteristiche nel modo suddetto, si ha che il prodotto scalare standard in $\mathbf{l}_2$ tra $\Phi(s)$ e $\Phi(t)$ può essere valutato mediante il kernel di Mercer, infatti:

$$K(s,t) = \langle \Phi(s) | \Phi(t) \rangle_{\mathbf{l}_2}$$

1.15 La norma negli spazi RKHS

La norma definita negli spazi RKHS può essere vista come una misura della complessità delle funzioni che appartengono allo spazio $\mathcal{H}_K$ delle funzioni di ipotesi, che si traduce a sua volta in un maggiore o minore grado di smoothness delle stesse.

Per vincolare lo spazio delle funzioni accettabili per un certo problema si può ricorrere quindi ad una condizione sulla norma in $\mathcal{H}_K$ come la seguente:

$$\|f\|_{\mathcal{H}_K} \leq M \quad \text{con:} \quad M \in \mathbb{R}^+,$$

a scopo illustrativo verrà dato un esempio relativo ad un caso tipico:

Esempio: Si consideri una qualunque funzione di una variabile, periodica, simmetrica e di classe $C^1(\mathbb{R})$ con coefficienti di Fourier positivi. Per la periodicità, la funzione reproducing kernel $K(s,t)$ può essere riscritta come $K(s-t) = K(z)$, inoltre essa può essere espansa in una serie di Fourier uniformemente convergente; ponendo per semplicità tutte le

costanti di normalizzazione uguali ad 1 risulta:

$$\begin{aligned} K(z) &= \sum_{n=0}^{\infty} \lambda_n \cos(nz) \\ K(s-t) &= \sum_{n=0}^{\infty} \lambda_n \cos(ns)\cos(nt) + \sum_{n=0}^{\infty} \lambda_n \sin(ns)\sin(nt) \end{aligned}$$

da cui risulta che le autofunzioni di K sono: $\{\chi_n(z) := \cos(nz)\}_{n=0}^{\infty}$, $\{\sigma_n(z) := \sin(nz)\}_{n=0}^{\infty}$ mediante esse è possibile calcolare la norma in $\mathcal{H}_K$ delle funzioni f che appartengono allo spazio RKHS:

$$\|f\|_{\mathcal{H}_K}^2 = \sum_{n=0}^{\infty} \frac{\langle f|\chi_n\rangle^2 + \langle f|\sigma_n\rangle^2}{\lambda_n}$$

Affinché tale norma sia limitata da un numero positivo M occorre che i coefficienti di Fourier della funzione $f \in \mathcal{H}_K$ diminuiscano in modo sufficientemente rapido da controbilanciare il decremento dei λ_n; in tal modo si selezionano le funzioni che hanno la maggior parte delle componenti spettrali diverse da zero a frequenze non elevate, cioè quelle che risultano più smooth.

1.16 Il Representer Theorem

Tutti i metodi di ottimizzazione presentati nella sezione 1.11 sono su uno spazio funzionale $\mathcal{H}_K$ che contiene un infinito numero di funzioni; gli elementi di tale spazio però si possono sviluppare usando le funzioni di kernel per cui:

$$\mathcal{H}_K = \left\{ f \;\mid\; f(s) = \sum_i a_i\, \phi_i(s) \right\}$$

in tal modo la procedura di ottimizzazione si trasferisce dalle funzioni ai coefficienti a_i, il cui numero potrebbe essere limitato oppure infinito, come nel caso in cui il kernel usato fosse gaussiano:

$$K(x,y) := \mathrm{e}^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

Il numero dei coefficienti $a_i \neq 0$ determina la dimensione del RKHS.

Il seguente teorema è molto generale, tanto che la procedura di regolarizzazione di Tikhonov può essere vista come un suo caso particolare:

Teorema. *Dato un set di punti* $S_n = \{(x_i, y_i)\}_{i=1}^n$*, una funzione della forma:*

$$f(x) = \sum_{i=1}^{n} a_i \, K(x_i, x)$$

è una minimizzatrice della seguente procedura di ottimizzazione:

$$\sum_{i=1}^{n} V\big(f(x_i), y_i\big) \; + \; \gamma \, g\big(\|f\|_{\mathcal{H}_K}\big)$$

in cui V *indica una arbitraria funzione di perdita,* g *una funzione monotona crescente e* $\|f\|_{\mathcal{H}_K}$ *è la norma nello spazio RKHS.*

Dimostrazione. Definendo $v(x) := \langle v|\Phi(x)\rangle_{\mathbf{l}_2}$, si assuma che la funzione $f(x)$ abbia la seguente espressione:

$$\begin{aligned}
f(x) \;&:=\; \sum_{i=1}^{n} a_i \, K(x_i, x) + \langle v|\Phi(x)\rangle_{\mathbf{l}_2} \\
&=\; \sum_{i=1}^{n} a_i \sum_j \lambda_j \phi_j(x_i)\phi_j(x) + \langle v|\Phi(x)\rangle_{\mathbf{l}_2} \\
&=\; \sum_{i=1}^{n} a_i \sum_j \Phi_j(x_i)\Phi_j(x) + \langle v|\Phi(x)\rangle_{\mathbf{l}_2} \\
&=\; \sum_j \left(\sum_{i=1}^{n} a_i \Phi_j(x_i) \right) \Phi_j(x) + \langle v|\Phi(x)\rangle_{\mathbf{l}_2} \\
&=\; \sum_j f_j \Phi_j(x) + \langle v|\Phi(x)\rangle_{\mathbf{l}_2} \quad \text{definendo:} \quad f_j := \left(\sum_{i=1}^{n} a_i \Phi_j(x) \right) \\
&=\; \langle f + v|\Phi(x)\rangle_{\mathbf{l}_2}
\end{aligned}$$

e che valga la condizione di ortogonalità:

$$\{\langle v|\Phi(x_i)\rangle_{\mathbf{l}_2}\}_{i=1}^n = 0$$

che impone a v di non appartenere allo span di: $\{\Phi(x_i)\}_{i=1}^n$.

In ogni punto x_i che appartiene al training set si ha:

$$\{\forall x_i\}_{i=1}^n \quad f(x_i) = \langle f + v | \Phi(x_i) \rangle_{\mathbf{l}_2} = \langle f | \Phi(x_i) \rangle_{\mathbf{l}_2}$$

da cui risulta che v non modifica i valori che la funzione $f(x)$ assume nei punti appartenenti al training set, quindi v non può influenzare neppure la funzione di perdita: $\sum_{i=1}^n V\big(f(x_i), y_i\big)$.

Analizzando la norma della funzione $f + v$ si trova:

$$\|f + v\| = \sqrt{\|f\|^2 + \|v\|^2} \geq \sqrt{\|f\|^2} = \|f\|$$

Poiché v incrementa la norma e non ha alcun effetto sulla funzione costo, la procedura di ottimizzazione dell'enunciato richiede che $v = 0$, perciò:

$$f_j = \sum_{i=1}^n a_i \, \Phi_j(x_i)$$

e di conseguenza:

$$f(x) = \langle f | \Phi(x) \rangle_{\mathbf{l}_2} = \sum_j \left(\sum_{i=1}^n a_i \, \Phi_j(x_i) \right) \Phi_j(x) = \sum_{i=1}^n a_i \, K(x_i, x)$$

□

1.17 Equivalenza delle tre regolarizzazioni

Ora che sono state trattate più in dettaglio le proprietà della norma in uno spazio RKHS, è possibile ritornare ai tre metodi descritti nella sezione 1.11, prendendo come funzione di regolarizzazione $\Omega(f)$ proprio la norma: $\|f\|^2_{\mathfrak{H}_K}$. Con il teorema seguente si dimostrerà come tali procedure possano essere considerate equivalenti per la ricerca della funzione obiettivo f_0:

Teorema. *Data una funzione di perdita convessa: $V(f(x), y)$, le tre forme di regolarizzazione seguenti sono equivalenti:*

$$(P1) \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \gamma \left\| f \right\|^2_{\mathcal{H}_K} \right]$$

$$(P2) \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) \right] \quad \textit{con il vincolo:} \quad \left\| f \right\|^2_{\mathcal{H}_K} \leq \nu$$

$$(P3) \qquad \min_{f \in \mathcal{H}_K} \left\| f \right\|^2_{\mathcal{H}_K} \quad \textit{con il vincolo:} \quad \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) \right] \leq \tau$$

Dove per equivalenti si intende che se $f_0(x)$ è la soluzione per uno dei problemi allora esistono dei parametri $\gamma, \nu, \tau \in \mathbb{R}^+$ per cui $f_0(x)$ è soluzione anche per gli altri.

Dimostrazione. Analizzando le varie implicazioni si ha:

- $(P2) \Rightarrow (P3)$: Se f_0 è la soluzione di $(P2)$ per un fissato ν, ed essa si ottiene in corrispondenza del valore massimo del vincolo, cioè per $\left\| f_0 \right\|^2_{\mathcal{H}_K} = \nu$, allora f_0 è soluzione anche di $(P3)$ se in essa si pone $\tau = \left[\frac{1}{n} \sum_{i=1}^{n} V(f_0(x_i), y_i) \right]$.

- $(P3) \Rightarrow (P2)$: Se f_0 è la soluzione di $(P3)$ per un fissato τ, ed essa si ottiene in corrispondenza del valore massimo del vincolo, cioè per $\left[\frac{1}{n} \sum_{i=1}^{n} V(f_0(x_i), y_i) \right] = \tau$, allora f_0 è soluzione anche di $(P2)$ se in essa si pone $\nu = \left\| f_0 \right\|^2_{\mathcal{H}_K}$.

- Se nei due casi precedenti le soluzioni non si ottengono in corrispondenza dell'estremo del vincolo, allora si può procedere cambiando il vincolo stesso fino a ricondursi alla condizione voluta, in caso contrario una soluzione trovata con un sistema rimane soluzione anche per l'altro, ma non è più garantita l'unicità.

- $(P2) \Rightarrow (P1)$: La forma di regolarizzazione $(P2)$ può essere riscritta

ricorrendo ai moltiplicatori di Lagrange nel modo seguente:

$$(P2') \qquad \min_{f \in \mathcal{H}_K} \max_{\alpha \geq 0} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \alpha \left(\|f\|^2_{\mathcal{H}_K} - \nu \right) \right]$$

il vincolo: $\|f\|^2_{\mathcal{H}_K} \leq \nu$ è ancora verificato, infatti se per ipotesi si assumesse $\|f\|^2_{\mathcal{H}_K} > \nu$, la massimizzazione su α farebbe divergere l'intera espressione, e tali casi sarebbero scartati nella fase di ricerca del minimo rispetto ad f.

Per regolarizzazioni come la $(P2')$, indicando con f_0 e α_0 i punti in cui f e α raggiungono i valori ottimali, vale la condizione di Karush-Kuhn-Tucker (KKT) di *complementary slackness*:

$$\alpha_0 \left(\|f_0\|^2_{\mathcal{H}_K} - \nu \right) = 0$$

Essa indica semplicemente che si può avere $\alpha_0 > 0$ solo quando $\|f_0\|^2_{\mathcal{H}_K} = \nu$, altrimenti non sarebbe soddisfatta la massimizzazione rispetto ad $\alpha \geq 0$ della $(P2')$, essendo sempre $\alpha \left(\|f_0\|^2_{\mathcal{H}_K} - \nu \right) \leq 0$, a causa del vincolo sulla norma di f che è stato descritto prima.

Conoscendo α_0 la $(P2')$ diventa:

$$(P2') \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \alpha_0 \left(\|f\|^2_{\mathcal{H}_K} - \nu \right) \right]$$

e poiché ν è una costante, si può eliminarla dalla minimizzazione:

$$(P2') \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \alpha_0 \|f\|^2_{\mathcal{H}_K} \right]$$

Ponendo $\alpha_0 = \gamma$ si può notare che $(P2')$ coincide con la procedura $(P1)$, per cui esse possono essere considerate equivalenti.

- $(P1) \Rightarrow (P2)$: La prova di tale implicazione si può ricavare semplicemente percorrendo a ritroso la dimostrazione precedente, perciò, per brevità, essa sarà omessa.

· $(P3) \Rightarrow (P1)$: Tale dimostrazione non sarebbe necessaria poiché $(P3) \Leftrightarrow (P2)$, ed è stato appena provato che $(P2) \Rightarrow (P1)$, tuttavia in tal modo non si conoscerebbe il valore della costante γ.

Procedendo esattamente come nel caso precedente, scrivendo però τ al posto di ν e scambiato tra loro i termini: $\left[\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i)\right]$ e $\|f\|^2_{\mathcal{H}_K}$ si trova che la forma di regolarizzazione $(P3)$ può essere scritta nei modi seguenti:

$$(P3') \qquad \min_{f\in\mathcal{H}_K} \left[\|f\|^2_{\mathcal{H}_K} + \alpha_0 \left(\left[\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i)\right] - \tau\right)\right]$$

$$(P3') \qquad \min_{f\in\mathcal{H}_K} \left[\|f\|^2_{\mathcal{H}_K} + \alpha_0 \left[\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i)\right]\right]$$

$$(P3') \qquad \min_{f\in\mathcal{H}_K} \left[\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i) + \frac{1}{\alpha_0}\|f\|^2_{\mathcal{H}_K}\right] \quad \forall \alpha_0 \neq 0$$

che corrisponde a $(P1)$ con $\gamma = \dfrac{1}{\alpha_0}$.

· $(P1) \Rightarrow (P3)$: Tale dimostrazione si può ottenere ripercorrendo la precedente dalla fine al principio, quindi anch'essa sarà omessa.

□

1.18 Le Regularization Networks

L'algoritmo delle *Regularization Networks* è il più famoso tra gli algoritmi di learning; esso è stato reinventato più volte in contesti diversi, ed è noto anche con i nomi di: *Kernel Ridge-Regression (KRR), Least Square Support Vector Machine (LSSVM), Regularized Least Square Classification (RLSC).*

Esso si basa sul metodo di regolarizzazione di Tikhonov:

$$\min_{f\in\mathcal{H}} \left[\frac{1}{n}\sum_{i=1}^{n} V(f(x_i), y_i) + \gamma\,\Omega(f)\right]$$

come *funzionale di regolarizzazione* si usa la norma di f nel RKHS:

$$\Omega(f) := \|f\|^2_{\mathcal{H}_K}$$

mentre come funzione di perdita si è scelta la *Square Loss*:

$$V(f(x), y) := (f(x) - y)^2$$

per cui l'algoritmo di ottimizzazione risultante è il seguente:

$$(RN) \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} (f(x) - y)^2 + \gamma \|f\|^2_{\mathcal{H}_K} \right] \quad \text{con:} \quad \gamma \in \mathbb{R}^+$$

Per il *Representer Theorem* la funzione minimizzatrice deve essere:

$$f(x) = \sum_{i=1}^{n} K(x, x_i)\, c_i \quad \text{con:} \quad c_i \in \mathbb{R}$$

in tale espressione $K(\cdot, \cdot)$ è una funzione di kernel, x è la variabile della $f(x)$, mentre gli $\{x_i \in X \subseteq \mathbb{R}^d\}_{i=1}^n$ rappresentano i dati di input del training set; gli unici termini non noti sono quindi i coefficienti $\{c_i\}_{i=1}^n$.

Con questa impostazione, una volta scelto un kernel K e fissato il parametro γ di cui si parlerà poi, il problema dell'apprendimento, tramite esempi, di una funzione generalizzante, inizialmente definito in uno spazio funzionale formato da infinite funzioni, si riduce ad una ottimizzazione degli n coefficienti reali c_i.

Nel seguito sarà utilizzato il simbolo $\mathbf{c}$, per indicare il vettore colonna formato dagli n coefficienti c_i, e $\mathbf{K}$ per rappresentare la matrice simmetrica $n \times n$ definita da:

$$\mathbf{K}_{ij} := K(x_i, x_j)$$

In tal modo la funzione $f(x)$, valutata nel training point x_i, può essere scritta in notazione matriciale come segue:

$$f(x_i) = \sum_{j=1}^{n} K(x_i, x_j)\, c_j = [\mathbf{K}\, \mathbf{c}]_i$$

avendo indicato con $[\mathbf{K}\,\mathbf{c}]_i$ l'i-esimo elemento del vettore $\mathbf{K}\,\mathbf{c}$.

Ora è possibile riscrivere l'algoritmo (RN) nella forma seguente:

$$(RN') \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} (\mathbf{K}\,\mathbf{c} - \mathbf{y})^2 + \gamma \, \|f\|^2_{\mathcal{H}_K} \right]$$

essendo $\mathbf{y}$ il vettore colonna degli output del training set.

Anche la norma $\|f\|^2_{\mathcal{H}_K}$ può essere valutata utilizzando l'algebra lineare, infatti per il *Representer Theorem*, indicando con $\mathbf{c}^T$ il vettore trasposto di $\mathbf{c}$, si ha:

$$(RN'f) \qquad \|f\|^2_{\mathcal{H}_K} = \mathbf{c}^T \mathbf{K}\,\mathbf{c}$$

sostituendo l'espressione suddetta nella (RN') il problema del learning diventa di ottimizzazione algebrica sul vettore $\mathbf{c}$:

$$(RN') \qquad \arg\min_{\mathbf{c} \in \mathbb{R}^n} \left[g(\mathbf{c}) := \frac{1}{n} (\mathbf{K}\,\mathbf{c} - \mathbf{y})^2 + \gamma \, \mathbf{c}^T \mathbf{K}\,\mathbf{c} \right]$$

La funzione $g(\mathbf{c})$ è una funzione a valori reali, convessa e differenziabile per cui può essere minimizzata semplicemente prendendo la derivata prima rispetto a $\mathbf{c}$ o a $\mathbf{c}^T$ e ponendola uguale a zero:

$$\frac{\partial g(\mathbf{c})}{\partial \mathbf{c}^T} = \frac{1}{n} \mathbf{K} (\mathbf{K}\,\mathbf{c} - \mathbf{y}) + \gamma \, \mathbf{K}\,\mathbf{c} = 0$$

da cui è possibile ricavare il vettore dei coefficienti $\mathbf{c}$:

$$\begin{aligned} & \mathbf{K} (\mathbf{K}\,\mathbf{c} - \mathbf{y}) + n\,\gamma \, \mathbf{K}\,\mathbf{c} = 0 \\ & \mathbf{K} (\mathbf{K} + n\,\gamma \, \mathbb{1}) \, \mathbf{c} = \mathbf{K}\,\mathbf{y} \\ (RN'\,\mathbf{c}) \qquad & \mathbf{c} = (\mathbf{K} + n\,\gamma \, \mathbb{1})^{-1} \, \mathbf{y} \end{aligned}$$

- Se $\gamma > 0$, la matrice $(\mathbf{K} + n\,\gamma \, \mathbb{1})$ è sicuramente definita positiva, perciò è invertibile, inoltre essendo simmetrica ha autovalori reali.
- Se $\gamma \to 0$ e $\mathbf{K}$ fosse invertibile, si avrebbe la classica soluzione dei minimi quadrati gaussiana.
- Se $\gamma \to \infty$, la soluzione $f(x) = \sum_{i=1}^{n} K(x, x_i)\, c_i$ diventerebbe la funzione identicamente nulla: $f(x) = 0$, dato che $\{c_i \to 0\}_{i=1}^{n}$.
- Se γ inoltre non è troppo piccolo la soluzione risulterà anche ben condizionata, come si dimostrerà nella sezione successiva.

1.19 Il numero di condizionamento

Il cambiamento che subisce la funzione f, indicato con Δf, a causa di una variazione $\Delta \mathbf{y}$ dei dati di output del training set, risulta vincolato dalla seguente disuguaglianza:

$$\frac{\|\Delta f\|}{\|f\|} \leq \|\mathbf{K} + n\,\gamma\,\mathbb{1}\| \left\|(\mathbf{K} + n\,\gamma\,\mathbb{1})^{-1}\right\| \frac{\|\Delta \mathbf{y}\|}{\|\mathbf{y}\|}$$

che può essere riscritta utilizzando il numero di condizionamento κ:

$$\frac{\|\Delta f\|}{\|f\|} \leq \kappa\big(\mathbf{K} + n\,\gamma\,\mathbb{1}\big) \frac{\|\Delta \mathbf{y}\|}{\|\mathbf{y}\|}$$

L'*errore relativo* sul modulo della funzione: $\|\Delta f\|/\|f\|$ è quindi limitato dall'*errore relativo* sul modulo dei valori di output: $\|\Delta \mathbf{y}\|/\|\mathbf{y}\|$ moltiplicato per il *numero di condizionamento*: $\kappa(\mathbf{K} + n\,\gamma\,\mathbb{1})$.

Un problema si dice *ben condizionato* quando il numero di condizionamento è piccolo, possibilmente vicino al minimo che è 1, in modo che non possa esserci un amplificazione troppo accentuata degli errori relativi.

Data una matrice non singolare $\mathbf{M}$ il numero di condizionamento può essere espresso in funzione dei suoi autovalori $\lambda_{\mathbf{M}}$, nel modo seguente:

$$\kappa(\mathbf{M}) = \frac{|\lambda_{max_{\mathbf{M}}}|}{|\lambda_{min_{\mathbf{M}}}|}$$

avendo indicato con $\lambda_{max_{\mathbf{M}}}$ e $\lambda_{min_{\mathbf{M}}}$ gli autovalori di modulo rispettivamente più grande e più piccolo. Se la matrice è definita positiva, come $\mathbf{K}$, tali moduli non occorrono, poiché gli autovalori sono tutti positivi.

Denotando con λ_{max} e λ_{min} gli autovalori: massimo e minimo di $\mathbf{K}$, il numero di condizionamento $\kappa\big(\mathbf{K} + n\,\gamma\,\mathbb{1}\big)$ risulta:

$$\kappa\big(\mathbf{K} + n\,\gamma\,\mathbb{1}\big) = \frac{\lambda_{max} + n\,\gamma}{\lambda_{min} + n\,\gamma} = 1 + \frac{\lambda_{max} - \lambda_{min}}{\lambda_{min} + n\,\gamma} = 1 + \frac{\kappa(\mathbf{K}) - 1}{1 + n\,\gamma/\lambda_{min}}$$

Da tale espressione si può vedere come sia importante il contributo del fattore $n\,\gamma$, che permette di trasformare in ben condizionato un problema che non lo era, in quanto caratterizzato da un $\kappa(\mathbf{K})$ elevato.

Il parametro γ ha quindi la funzione di regolare il cosiddetto *bias-variance trade-off*, tra l'*errore sistematico* che controlla le proprietà di generalizzazione e di smoothness, e l'*errore empirico* sul fitting dei dati.

1.20 La determinazione dei coefficienti

Dato un training set formato da n coppie di valori di input-output, per risolvere l'equazione (RN' $\mathbf{c}$) ricavata nella sezione 1.18:

$$\mathbf{c} = (\mathbf{K} + n\,\gamma\,\mathbb{1})^{-1}\,\mathbf{y}$$

è necessario invertire la matrice $n \times n : (\mathbf{K} + n\,\gamma\,\mathbb{1})$.

Benché tale matrice, per $\gamma > 0$ sia sempre invertibile, un approccio diretto per n molto grande risulterebbe impraticabile, dato che occorrerebbe acquisire l'intera matrice in memoria per poterla invertire.

In tali casi si ricorre a degli algoritmi iterativi che richiedono meno risorse per poter essere eseguiti, sia in termini di capacità di memoria che di potenza di calcolo, come quello illustrato nella sezione 2.4.

1.21 Interpretazione bayesiana

Il principio variazionale alla base dell'algoritmo (RN) di cui si è parlato nella sezione 1.18:

$$(RN) \qquad \min_{f \in \mathcal{H}_K} \left[\frac{1}{n} \sum_{i=1}^{n} (f(x) - y)^2 + \gamma \, \|f\|^2_{\mathcal{H}_K} \right] \quad \text{con:} \quad \gamma \in \mathbb{R}^+$$

può essere derivato anche utilizzando la teoria delle probabilità di Bayes e la stima del *Maximum A Posteriori (MAP)*.

Per la dimostrazione sono necessarie le seguenti definizioni:

- $S_n = \{(x_i, y_i)\}_{i=1}^n$: è l'insieme di training, formato da n coppie di dati di input-output.
- $P(f|S_n)$: è la probabilità condizionata di trovare la funzione f dato il training set S_n.
- $P(S_n|f)$: è la probabilità condizionata di ricavare il training set S_n data la funzione f; questo termine è un *modello del rumore* che si sovrappone ai valori di output generati dalla funzione f stessa.

- $P(f)$: è la probabilità a priori di ottenere la funzione f da un assegnato *random field* di funzioni. Essa incorpora la nostra conoscenza a priori delle funzioni, e può essere usata per assegnare vincoli al modello, in modo che solo le funzioni che li soddisfino abbiano delle probabilità a priori significativamente diverse da zero.

- $P(S_n)$: è la probabilità a priori di ottenere il training set S_n da un determinato *random field* di training sets. Se si assegna una probabilità uniforme per tutti i possibili outcomes, come di consuetudine, il valore costante di $P(S_n)$ risulta ininfluente per la dimostrazione.

Dimostrazione. Date le distribuzioni di probabilità: $P(S_n|f)$, $P(f)$ e $P(S_n)$, la probabilità a posteriori $P(f|S_n)$ può essere calcolata con la regola di Bayes:

$$P(f|S_n) = \frac{P(S_n|f)\,P(f)}{P(S_n)}$$

Assumendo che le y_i siano misure soggette ad un rumore gaussiano, cioè: $y_i := f(x_i) + \varepsilon_i$, essendo ε_i variabili gaussiane i.i.d. (indipendenti ed identicamente distribuite) di deviazione standard σ, la probabilità condizionata $P(S_n|f)$ risulta:

$$P(S_n|f) = \frac{1}{Z_l}\prod_{i=1}^{n} \mathrm{e}^{-\frac{1}{2\sigma^2}(y_i - f(x_i))^2} = \frac{1}{Z_l}\mathrm{e}^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - f(x_i))^2}$$

avendo denotato con Z_l la costante di normalizzazione, da *likelihood* che è il termine con cui si indica $P(S_n|f)$.

La probabilità a priori $P(f)$ può essere formalmente definita come:

$$P(f) := \frac{1}{Z_p}\mathrm{e}^{-\|f\|^2_{\mathfrak{H}_K}}$$

con Z_p costante di normalizzazione, da *prior* con cui è nota $P(f)$.

La probabilità a priori $P(S_n)$, detta *evidence*, è definita costante su tutto lo spazio dei training sets:

$$P(S_n) := Z_e$$

Dalle ipotesi suddette e dalla regola di Bayes segue:

$$P(f|S_n) = \frac{1}{Z_e\, Z_l\, Z_p} \mathrm{e}^{-\left[\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \|f\|^2_{\mathcal{H}_K}\right]}$$

Per stimare la funzione f si può applicare il principio del *Maximum A Posteriori (MAP)* alla *probabilità a posteriori* $P(f|S_n)$ appena trovata:

$$\begin{aligned} f &= \max_{f\in\mathcal{H}_K} P(f|S_n) = \min_{f\in\mathcal{H}_K}\left[\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \|f\|^2_{\mathcal{H}_K}\right] = \\ &= \min_{f\in\mathcal{H}_K}\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \frac{2\sigma^2}{n}\|f\|^2_{\mathcal{H}_K}\right] = \\ &= \min_{f\in\mathcal{H}_K}\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \widetilde{\gamma}\,\|f\|^2_{\mathcal{H}_K}\right] \quad \text{definendo:} \quad \widetilde{\gamma} := \frac{2\sigma^2}{n} \end{aligned}$$

□

Confrontando l'ultima espressione con la (RN) si può notare che esse coincidono. Va sottolineato però che $\widetilde{\gamma}$ non è una vera costante, dato che dipende dalla dimensione n del training set, infatti: $\widetilde{\gamma} = \widetilde{\gamma}(n) := \frac{2\sigma^2}{n}$.

In base alle definizioni utilizzate per la dimostrazione, risulta che alle componenti dell'algoritmo (RN) si può dare la seguente interpretazione:

- Il primo termine, detto *data term*, con la funzione di perdita quadratica, corrisponde ad un modello di rumore gaussiano additivo.
- Il secondo termine, chiamato *stabilizzatore*, con la norma di f nel RKHS, è associabile ad una probabilità a priori sulle funzioni f dello spazio di funzioni di ipotesi $\mathcal{H}_K$.

1.21.1 Interpretazione bayesiana dello stabilizzatore

Dato che ogni funzione $f \in \mathcal{H}_K$ si può sviluppare sulla base delle autofunzioni ϕ_i del kernel K, indicando con n, che potrebbe essere anche

infinito, la dimensione del RKHS, si ha:

$$f(x) = \sum_{i=1}^{n} a_i \, \phi_i(x)$$

applicando la definizione di prodotto tra funzioni appartenenti ad $\mathcal{H}_K$ data nelle sezione 1.14, lo stabilizzatore diventa:

$$\|f\|^2_{\mathcal{H}_K} = \sum_{i=1}^{n} \frac{a_i^2}{\lambda_i} = \mathbf{a}^T \mathbf{\Lambda}^{-1} \mathbf{a}$$

avendo indicato con $\mathbf{a}$ il vettore colonna dei coefficienti a_i e con $\mathbf{\Lambda}$ la matrice diagonale degli autovalori λ_i.

Cambiando sistema di riferimento e passando alla base formata dalle funzioni: $\boldsymbol{\phi'} = \boldsymbol{A}\,\boldsymbol{\phi}$ si ha:

$$\|f\|^2_{\mathcal{H}_K} = \mathbf{b}^T \mathbf{\Sigma}^{-1} \mathbf{b}$$

in tal caso $\mathbf{\Sigma}$ potrebbe essere interpretata come la matrice di covarianza nel nuovo sistema di riferimento, e lo stabilizzatore $\|f\|_{\mathcal{H}_K}$ come la distanza di Mahalanobis dalla media delle funzioni, dato che:
Definizione: *la* distanza di Mahalanobis *di un un vettore multivariato* $\mathbf{y} = (y_1, \ldots, y_n)$ *da un gruppo di valori con media* $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ *e matrice di covarianza* $\mathbf{\Sigma}$ *è:*

$$D_M(\mathbf{y}) := \sqrt{(\mathbf{y} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1} (\mathbf{y} - \boldsymbol{\mu})}$$

Ricordando che una funzione multivariata gaussiana con media $\boldsymbol{\mu}$ e varianza $\mathbf{\Sigma}$ è definita come segue:

$$(\mathbf{y}|\boldsymbol{\mu}, \mathbf{\Sigma}) := |2\pi\mathbf{\Sigma}|^{-\frac{1}{2}} \mathrm{e}^{-\frac{1}{2}\{(\mathbf{y}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\}}$$

si può notare come $P(f)$ risulti una gaussiana multivariata con media zero, nello spazio delle funzioni definito dal kernel K e generato da tutte le possibili combinazioni lineari dalle ϕ_i:

$$P(f) := \frac{1}{Z_p} \mathrm{e}^{-\|f\|^2_{\mathcal{H}_K}} = \frac{1}{Z_p} \mathrm{e}^{(\mathbf{b}^T \mathbf{\Sigma}^{-1} \mathbf{b})}$$

In tal modo lo stabilizzatore può essere collegato ad una probabilità a priori gaussiana sullo spazio di funzioni.

Esiste dunque una relazione di equivalenza tra processi gaussiani definiti da una probabilità condizionata $P(f|S_n)$ gaussiana e le *Regularization Networks* definite tramite la procedura (RN), la comunità delle *neural networks* sembrerebbe comunque aver notato tale relazione solo piuttosto recentemente; a tal proposito si veda [21].

Ora, con l'ausilio di quanto esposto in questa sezione, è possibile dimostrare che la scelta di un kernel K equivale ad assumere una probabilità a priori gaussiana su f, con covarianza uguale a $\mathbf{K}$, e quindi ad attribuire una funzione di correlazione associata alla famiglia di funzioni f.

Dimostrazione. Partendo dall'equazione $(RN'f)$ della sezione 1.18, e ricordando che la matrice $\mathbf{K}$ è simmetrica si ha:

$$\|f\|^2_{\mathcal{H}_K} = \mathbf{c}^T\mathbf{K}\,\mathbf{c} = \mathbf{c}^T\mathbf{K}\,\mathbf{K}^{-1}\mathbf{K}\,\mathbf{c} = \mathbf{c}^T\mathbf{K}^T\,\mathbf{K}^{-1}\mathbf{K}\,\mathbf{c} = \mathbf{b}^T\mathbf{K}^{-1}\,\mathbf{b}$$

avendo posto $\mathbf{b} = \mathbf{Kc}$.

Tale vettore $\mathbf{b}$ è formato dalle funzioni: $\{f(x_i)\}_{i=1}^n$ per cui può essere visto come appartenente allo spazio di funzioni $\mathcal{H}_K$.

Ci si è ricondotti quindi alla stessa espressione della norma trovata prima, con i vettori $\mathbf{b}$ appartenenti allo stesso spazio di funzioni $\mathcal{H}_K$, per cui si può trasferire interamente l'interpretazione data precedentemente al caso in questione e risulta che:

- $\mathbf{K}$ è interpretabile come una matrice di covarianza
- la probabilità a priori $P(f)$ risulta una distribuzione di probabilità multivariata gaussiana con media 0 e varianza $\mathbf{K}$.

□

Capitolo 2

La teoria dei frame

2.1 Definizione

I frame sono una generalizzazione del concetto di set overcompleto di vettori di uno spazio di Hilbert; la loro definizione è la seguente:
Definizione: *Una sequenza di vettori* $\{f_i\}$ *di uno spazio di Hilbert* $\mathcal{H}$ *è detta* frame *se:*

$$\exists \alpha, \beta \in \mathbb{R} \ , \ 0 < \alpha \leq \beta < \infty \ | \ \forall f \in \mathcal{H} \quad \alpha \|f\|^2 \leq \sum_i |\langle f_i | f \rangle|^2 \leq \beta \|f\|^2$$

per cui la serie di Bessel dei vettori $f \in \mathcal{H}$ *è limitata sia inferiormente che superiormente.*

I numeri α e β sono chiamati rispettivamente limite *inferiore* e *superiore* del frame. Il più grande $\alpha > 0$ e il più piccolo $\beta < \infty$ che soddisfano la condizione suddetta sono chiamati *limiti ottimi* del frame. Se $\alpha = \beta$ il frame è detto *tight*, se inoltre $\alpha = \beta = 1$ si parla di *tight frame normalizzato* o più semplicemente di *frame normalizzato*. Un frame è *esatto* se rimuovendo un vettore qualsiasi esso non è più un frame, in caso contrario è detto *overcompleto*.

Esempi:

· Ogni base ortonormale è un tight frame normalizzato

· I seguenti sono tutti tight frames normalizzati:

$$\{e_1,\ 0,\ e_2,\ 0,\ e_3,\ 0,\ \dots\},$$
$$\{\frac{e_1}{\sqrt{2}},\ \frac{e_1}{\sqrt{2}},\ \frac{e_2}{\sqrt{2}},\ \frac{e_2}{\sqrt{2}},\ \dots\},$$
$$\{e_1,\ \frac{e_2}{\sqrt{2}},\ \frac{e_2}{\sqrt{2}},\ \frac{e_3}{\sqrt{3}},\ \frac{e_3}{\sqrt{3}},\ \frac{e_3}{\sqrt{3}},\ \dots\}$$

· Un frame formato con i vettori di n basi ortonormali è un tight frame con $\alpha = \beta = n$

· Se ai vettori di una base ortonormale si aggiungono l vettori unitari arbitrari si ottiene un frame con bounds: $\alpha = 1$ e $\beta = 1 + l$

2.2 L'operatore di frame

I frame possono essere visti come operatori, in tal modo è possibile ricondursi ai grandi risultati ottenuti in tale campo ed utilizzare la usuale notazione di Dirac.

Sia $|e_n\rangle$ una base ortonormale; indicando con $|f_n\rangle$ i vettori dello spazio di Hilbert $\mathcal{H}$ che costituiscono il frame, si ha:
Definizione: *L'*operatore di sintesi *o* operatore di preframe $F : \mathbf{l}_2 \to \mathcal{H}$ *è quell'operatore che mappa i vettori della base* $|e_n\rangle$ *sui vettori del frame:*

$$\forall n \in \mathbb{N} \quad F\,|e_n\rangle = |f_n\rangle$$

e può essere scritto come: $F = \sum_i |f_i\rangle\langle e_i|$
Definizione: *L'operatore aggiunto di* F*, è chiamato* operatore di analisi*,* $F^\dagger : \mathcal{H} \to \mathbf{l}_2$ *e sviluppa i vettori* $f \in \mathcal{H}$ *sulla base dei vettori* $|e_n\rangle$:

$$\forall n \in \mathbb{N} \quad F^\dagger\,|f_n\rangle = \left(\sum_i |e_i\rangle\langle f_i|\right)|f_n\rangle = \sum_i \langle f_i|f_n\rangle\,|e_i\rangle = \sum_i c_{n_i}|e_i\rangle$$

L'operatore di sintesi F è limitato se e solo se il limite superiore del frame $\beta < \infty$ infatti:

$$\forall f \in \mathcal{H} \quad \left\|F^\dagger f\right\|^2 = \sum_i |\langle f_i|f\rangle|^2 < \beta\|f\|^2$$

Le seguenti tre affermazioni sono equivalenti:

1. *La sequenza di vettori* $\{f_n\}$ *è un frame per* $\mathcal{H}$.

2. *L'operatore di sintesi* F *è limitato, lineare e* onto *(per cui potrebbero esserci elementi distinti del dominio che vanno nello stesso elemento del codominio, dato che non è imposta la iniettività).*

3. *L'operatore di analisi* $F^\dagger$ *è un isomorfismo possibilmente* into *chiamato* trasformazione di frame.

inoltre $\{f_n\}$ è un tight frame normalizzato se e solo se l'operatore di sintesi F è una mappa quoziente, cioè una isometria parziale.

La dimensione del kernel di F: $\text{Null}(F) := \dim(\text{Ker}(F))$ è chiamata *eccesso* del frame.

Un frame $\{f_n\}$ per uno spazio di Hilbert $\mathcal{H}$ è esatto se e solo se è una base di Riesz.

Dimostrazione. Il frame è esatto se e solo se l'operatore di sintesi F è iniettivo; dato che F è anche limitato, lineare e onto allora esso deve essere un operatore invertibile, per cui $\{f_n\}$ risulta una base di Riesz. Le costanti associate a tale base sono i numeri: $\sqrt{\alpha}$ e $\sqrt{\beta}$. La prova del viceversa è ovvia dato che ogni base di Riesz è un frame esatto, non potendo togliervi alcun elemento senza far sì che non sia più una base. □

Dato un frame $\{f_n\}$ per uno spazio di Hilbert $\mathcal{H}$ tale che $f = \sum_i c_i f_i$, le seguenti affermazioni sono equivalenti:

1. $\{f_n\}$ è un frame esatto,

2. $\{f_n\}$ è una base di Riesz per $\mathcal{H}$, per cui i coefficienti c_i sono unici,

3. $\{f_n\}$ è l'immagine di una base ortonormale data dall'operatore limitato e invertibile di sintesi F,

4. l'operatore di analisi $F^\dagger$ mappa su l_2,

5. esistono due costanti $\widetilde{\alpha}$ e $\widetilde{\beta}$ tali che: $\widetilde{\alpha}\,\|c\| \leq \|\sum_i c_i\, f_i\| \leq \widetilde{\beta}\,\|c\|$ per tutte le sequenze finite $c \in l_2$,

6. L'operatore di Gram $G = F^\dagger F$ è positivo e invertibile su l_2.

Definizione: *Dato un frame* $\{f_i\}$ *l'operatore* $S = F\,F^\dagger$ *è un operatore definito positivo, autoaggiunto e invertibile chiamato* operatore di frame:

$$S = \sum_i |f_i\rangle\langle f_i|$$

Un frame ha limiti $0 < \alpha \leq \beta < \infty$ se e solo se $\alpha\,\mathbb{1} \leq S \leq \beta\,\mathbb{1}$, per cui si può avere:

· un tight frame se e solo se $S = \alpha\,\mathbb{1}$

· un tight frame normalizzato se e solo se $S = \mathbb{1}$

I limiti ottimi del frame F possono essere espressi in funzione dell'operatore di frame S, infatti: $\beta = \|S\|$ e $\alpha = \|S^{-1}\|^{-1}$, ne consegue che il numero di condizionamento della matrice S può essere scritto come: $\kappa(S) := \|S\|\,\|S^{-1}\| = \beta/\alpha$.

Dato un frame $\{f_n\}$, l'operatore di frame S è unico per definizione, a differenza degli operatori di sintesi F e di analisi $F^\dagger$ la cui espressione varia a seconda dei vettori $|e_n\rangle$ scelti come base.

Teorema. *Ogni frame* $\{f_n\}$*, con il relativo operatore di frame S, è equivalente al tight frame normalizzato:* $\{S^{-1/2} f_n\}$*, infatti:*

Dimostrazione.

$$\begin{aligned} f &= S\,S^{-1} f = \sum_i \langle f_i|S^{-1} f\rangle\,|f_i\rangle = \sum_i \langle S^{-1} f_i|f\rangle\,|f_i\rangle \\ &= S^{-1/2}\,S\,S^{-1/2} f = \sum_i \langle S^{-1/2} f_i|f\rangle\,|S^{-1/2} f_i\rangle \end{aligned}$$

essendo $S^{-1/2}$ l'operatore ottenuto dalla radice quadrata positiva dell'operatore positivo S^{-1}, con: $\beta^{-1}\,\mathbb{1} \leq S^{-1} \leq \alpha^{-1}\,\mathbb{1}$. □

Solitamente lo sviluppo di f dato nella prima riga della dimostrazione precedente è chiamato *formula di ricostruzione* e i termini: $\langle S^{-1} f_i | f \rangle$ sono detti *coefficienti del frame* per $|f\rangle$.

Teorema. *Per ogni frame* $\{f_n\}$ *di* $\mathcal{H}$ *c'è un unico operatore* $\widetilde{S} \in \mathcal{B}(\mathcal{H})$ *tale che:*

$$\forall f \in \mathcal{H} \quad f = \sum_i \langle \widetilde{S} f_i | f \rangle \, |f_i\rangle$$

L'operatore $\widetilde{S}$ *può essere scritto come:* $\widetilde{S} = G\, G^\dagger$*, essendo* $G^\dagger$ *un operatore invertibile* $\in \mathcal{B}(\mathcal{H}, K)$ *per qualche spazio di Hilbert* K*, tale che* $\{G^\dagger\, f_i\}$ *sia un tight frame normalizzato. Inoltre* $\widetilde{S}$ *risulta essere un operatore definito positivo e invertibile.*

Dimostrazione. Per il teorema dimostrato prima si ha: $\widetilde{S} = S^{-1}$ ed inoltre $\{G^\dagger\, f_i\} = \{S^{-1/2} f_i\}$; se S è invertibile come dovrebbe, la unicità è garantita, a meno di permutazioni dei vettori di frame. □

L'operatore $F^\dagger F$ è chiamato operatore di Gram del frame ed è una matrice definita sullo spazio di Hilbert degli $|e_i\rangle$.

2.3 I frame duali

Definizione: *Dato un frame* $\{f_n\}$*, un frame* $\{h_n\}$ *è chiamato* frame duale alterno *se soddisfa la seguente espansione:*

$$\forall f \in \mathcal{H} \quad f = \sum_i \langle h_i | f \rangle \, |f_i\rangle$$

Se un frame non è esatto allora si possono trovare infiniti duali alterni che in generale non sono frame.

Definizione: *Dato un frame* $\{f_n\}$ *con limiti* α *e* β*, il frame* $|S^{-1} f_n\rangle$*, avente come limiti inferiore e superiore rispettivamente* β^{-1} *e* α^{-1}*, è chiamato* frame duale canonico*, ed è indicato con* $\{f_n^*\}$*.*

Esempi:

· Dato il frame: $\{e_1,\ e_1,\ e_2,\ e_2,\ \dots\}$, i seguenti sono duali alterni:

$$\{e_1,\ 0,\ e_2,\ 0,\ \dots\}, \quad \{0,\ e_1,\ 0,\ e_2,\ \dots\}$$
$$\forall a_i \quad \{a_1\, e_1,\ (1-a_1)\, e_1,\ a_2\, e_2,\ (1-a_2)\, e_2,\ \dots\}$$

il frame duale canonico $\{S^{-1} f_n\}$ è dato da:

$$\{\frac{e_1}{2},\ \frac{e_1}{2},\ \frac{e_2}{2},\ \frac{e_2}{2},\ \dots\}$$

· Dato il frame: $\{e_1,\ \frac{e_2}{\sqrt{2}},\ \frac{e_2}{\sqrt{2}},\ \frac{e_3}{\sqrt{3}},\ \frac{e_3}{\sqrt{3}},\ \frac{e_3}{\sqrt{3}},\ \dots\}$, coincidente con il suo duale canonico, si ha che la seguente sequenza di vettori:

$$\{e_1,\ \sqrt{2}\, e_2,\ 0,\ \sqrt{3}\, e_3,\ 0,\ 0,\ \dots\}$$

non è un frame, non essendo superiormente limitata, ma soddisfa comunque l'espansione:

$$\forall f \in \mathcal{H} \quad f = \sum_i \langle h_i | f \rangle\, |f_i\rangle$$

Ricercare i duali dell'operatore F equivale a calcolare degli operatori H inversi a sinistra dell'operatore di analisi $F^\dagger$, per i quali quindi vale la relazione:

$$H\, F^\dagger = \mathbb{1}_{\mathcal{H}}$$

Affinché esista l'inverso a sinistra occorre che un operatore sia *iniettivo*, e nel caso dell'operatore di analisi $F^\dagger$ tale condizione è soddisfatta:

Dimostrazione. se per assurdo non valesse l'iniettività si potrebbe avere:

$$\exists f, g \in \mathcal{H}\ ,\ f \neq g \quad | \quad F^\dagger f = F^\dagger g\ \Rightarrow$$
$$\Rightarrow F^\dagger (f-g) = 0\ \Rightarrow\ \sum_i \langle f_i | (f-g) \rangle = 0$$

ma il vettore $(f-g)$ non può essere ortogonale a tutti i vettori di frame $\{f_i\}$ senza essere il vettore nullo perché i frames sottendono tutto lo spazio $\mathcal{H}$, essendo una base completa o overcompleta, per cui deve essere $f = g$ in contraddizione con l'enunciato del teorema. □

Una formula generale per calcolare gli inversi a sinistra di un operatore è la seguente:

$$\begin{aligned} H &= \left(F\,F^\dagger\right)^{-1} F \;+\; W^\dagger \left(\mathbb{1} - F^\dagger \left(F\,F^\dagger\right)^{-1} F\right) = \\ &= S^{-1}F \;+\; W^\dagger \left(\mathbb{1} - F^\dagger\, S^{-1} F\right) \end{aligned}$$

avendo indicato con S l'operatore di frame ed essendo $W \in \mathcal{B}(\mathcal{H}, \mathbf{l}_2)$ un operatore arbitrario.

Noto H, per avere la sequenza di vettori duali $\{f_n^\vee\}$ basta moltiplicare a destra per gli $\{e_n\}$ che costituiscono la base ortonormale di $\mathbf{l}_2$:

$$\begin{aligned} f_n^\vee &= H\,e_n = S^{-1} f_n \;+\; W^\dagger \left(e_n - F^\dagger\, S^{-1} f_n\right) = \\ &= f_n^* + W^\dagger e_n - W^\dagger F^\dagger f_n^* = f_n^* + g_n - \sum_i \langle f_i | f_n^* \rangle\, g_i \end{aligned}$$

avendo indicato con f_n^* il vettore duale canonico di f_n.

Se il frame f_n è esatto i coefficienti del frame sono unici, per cui: $\langle f_i | f_n^* \rangle = \delta_{i,n}$ e anche il frame duale risulta unico: $f_n^\vee = f_n^*$.

Il caso più interessante comunque è quello in cui il frame è overcompleto perché variando i vettori g_i si esplora tutto lo spazio dei vettori duali, ed è possibile effettuare tramite essi delle ottimizzazioni, alla ricerca della *sequenza di analisi ottimale* in funzione dei criteri assegnati.

2.4 L'inversione dell'operatore di frame

Quando si utilizzano i frame è spesso necessario trovare i duali ed in particolare il duale canonico. In linea di principio esso potrebbe essere ottenuto invertendo l'operatore di frame S, purtroppo però S generalmente non corrisponde a una matrice di dimensione finita. Per poter superare questo problema si sono cercati allora degli algoritmi iterativi che potessero essere eseguiti anche per via numerica, in modo da permetterne l'elaborazione elettronica. La formula ricorsiva di Richardson consente di calcolare l'azione dell'inverso dell'operatore di frame su un vettore: $S^{-1}v$, senza calcolare esplicitamente S^{-1}:

$$\forall i \geq 0, \quad v_{i+1} = v_i + \lambda\,(v - S\,v_i), \quad \text{con: } v_0 = 0, \text{ e } \lambda > 0$$

essendo λ un parametro numerico. Una volta garantite le condizioni di convergenza agendo sul λ, si può facilmente verificare che:

$$\lim_{i\to\infty} v_i = S^{-1}v$$

Definendo: $R := 1 - \lambda\, S$, la precedente ricorsione può essere riscritta come segue:

$$v_{i+1} = \lambda\, v + R\, v_i$$

da cui si ricava:

$$v_{i+1} - v_i = R\,(v_i - v_{i-1}) = R^i\, v_1 = \lambda\, R^i\, v$$

tale espressione dà il *rate* geometrico di convergenza infatti:

$$\begin{aligned} v_{i+1} &= v_i + \lambda\,(v - S\, v_i) \\ v - S\, v_i &= \tfrac{1}{\lambda}\,(v_{i+1} - v_i) \\ S^{-1}\, v - v_i &= \tfrac{1}{\lambda}\, S^{-1}(v_{i+1} - v_i) \end{aligned}$$

per cui:

$$\left\|v_i - S^{-1}\, v\right\| = \tfrac{1}{\lambda}\, \left\|S^{-1}\,(v_{i+1} - v_i)\right\| = \tfrac{1}{\lambda}\, \left\|S^{-1}\left(\lambda\, R^i\, v\right)\right\|$$

dato che R per come è stato definito commuta con S^{-1} si ha:

$$\left\|v_i - S^{-1}\, v\right\| = \left\|R^i\, S^{-1}\, v\right\| \leq \left\|R^i\right\|\, \left\|S^{-1}\, v\right\|$$

Per λ piccoli si ha $\|R\| \approx 1$, e la convergenza è lenta, per valori troppo grandi invece la ricorsione non convergerebbe; il valore ottimale λ_{opt} può essere espresso in funzione dei limiti inferiore e superiore del frame, indicati rispettivamente con α e β, e risulta:

$$\lambda_{opt} = \frac{2}{\alpha + \beta}$$

tale valore può essere ricavato con il seguente procedimento:

Dimostrazione. Per quando esposto nella sezione 2.2 vale:

$$\alpha \mathbb{1} \leq S \leq \beta \mathbb{1}$$

con: $0 < \alpha \leq \beta < \infty$. Esprimendo, S in funzione di R, si ha:

$$\alpha \mathbb{1} \leq \tfrac{1}{\lambda}(1 - R) \leq \beta \mathbb{1}$$
$$(1 - \lambda \beta) \mathbb{1} \leq R \leq (1 - \lambda \alpha) \mathbb{1}$$

se si vuole che il massimo valore assoluto degli autovalori di R sia il più piccolo possibile occorre che essi occupino un intervallo di valori simmetrico rispetto allo 0, per cui si dovrà avere:

$$\begin{aligned} -(1 - \lambda \beta) &= 1 - \lambda \alpha \\ \lambda (\alpha + \beta) &= 2 \\ \lambda &= \frac{2}{\alpha + \beta} \end{aligned}$$

sostituendo tale valore di λ nella disuguaglianza precedente risulta:

$$\frac{\alpha - \beta}{\alpha + \beta} \leq R \leq \frac{\beta - \alpha}{\alpha + \beta}$$

affinché si abbia convergenza si deve avere $\|R\| < 1$, e ciò si traduce nella seguente condizione:

$$\|R\| = \frac{\beta - \alpha}{\alpha + \beta} < 1$$

che è sempre soddisfatta essendo $0 < \alpha \leq \beta < \infty$. □

Ricordando la definizione di numero di condizionamento di S:

$$\kappa(S) := \left\|S\right\| \left\|S^{-1}\right\| = \beta/\alpha$$

la precedente espressione può essere riscritta come:

$$\|R\| = \frac{\frac{\beta}{\alpha} - 1}{\frac{\beta}{\alpha} + 1} = \frac{\kappa(S) - 1}{\kappa(S) + 1} = 1 - \frac{2}{1 + \kappa(S)} \quad \text{con: } \kappa(S) \geq 1$$

da cui si deduce che la convergenza è tanto peggiore, quanto più la matrice S è mal condizionata, cioè ha un $\kappa(S) \gg 1$, dato che si avrebbe $\|R\| \approx 1$.

Quando i limiti del frame non sono noti la convergenza risulta dipendere non soltanto dal frame in sé ma anche dalla stima dei vincoli, e dato che usualmente si stimano dei limiti che sono molto meno *tight* di quanto effettivamente non siano, si finisce per avere un $\widetilde{\kappa}(S)$ stimato molto peggiore di quello reale e quindi una convergenza più lenta del dovuto.

2.5 La pseudo-inversa di Moore-Penrose

Dato che con i frame è spesso necessario il calcolo della matrice pseudo-inversa di Moore-Penrose in questa sezione saranno richiamate le sue proprietà principali.

Dato un operatore lineare $\mathbf{X}$ in forma matriciale su una base fissata, il suo aggiunto, o hermitiano coniugato, sarà indicato con $\mathbf{X}^\dagger$, il complesso coniugato con $\overline{\mathbf{X}}$ e la trasposta con $\mathbf{X}^T$.

La pseudo-inversa di Moore-Penrose della matrice $\mathbf{X}$: $\mathbf{X}^+$ è unica e può essere definita dalle seguenti equazioni, che devono essere tutte soddisfatte:

1. $\mathbf{X}\,\mathbf{X}^+\mathbf{X} = \mathbf{X}$
2. $\mathbf{X}^+\mathbf{X}\,\mathbf{X}^+ = \mathbf{X}^+$
3. $(\mathbf{X}\,\mathbf{X}^+)^\dagger = \mathbf{X}\,\mathbf{X}^+$
4. $(\mathbf{X}^+\mathbf{X})^\dagger = \mathbf{X}^+\mathbf{X}$

Gli operatori hermitiani e idempotenti: $\mathbf{X}\,\mathbf{X}^+$ e $\mathbf{X}^+\mathbf{X}$ sono proiettori ortogonali rispettivamente sul Range($\mathbf{X}$) e sul supporto di $\mathbf{X}$: Supp($\mathbf{X}$) = Range($\mathbf{X}^\dagger$) = Ker($\mathbf{X}$)$^\perp$.

La pseudo-inversa di Moore-Penrose può essere calcolata usando la decomposizione in valori singolari (*Singular Value Decomposition* (SVD)): se $\mathbf{X} = U\,\Sigma\,V^\dagger$ allora $\mathbf{X}^+ = V\,\Sigma^+U^\dagger$, essendo U e V matrici unitarie, Σ la matrice diagonale dei valori singolari di $\mathbf{X}$, ordinati in modo decrescente, e Σ^+ la matrice ottenuta prendendo il reciproco di ogni elemento non nullo sulla diagonale di Σ.

Un metodo alternativo per determinare la pseudo-inversa di Moore-Penrose utilizza un processo di limite:

$$\mathbf{X}^{+} = \lim_{\delta \to 0} \left(\mathbf{X}^{\dagger}\mathbf{X} + \delta \, \mathbb{1} \right)^{-1} \mathbf{X}^{\dagger} = \lim_{\delta \to 0} \mathbf{X}^{\dagger} \, \left(\mathbf{X}\,\mathbf{X}^{\dagger} + \delta \, \mathbb{1} \right)^{-1}$$

Tali limiti esistono anche se $(\mathbf{X}^{\dagger}\,\mathbf{X})$ e $(\mathbf{X}\,\mathbf{X}^{\dagger})$ non sono invertibili.

Tra tutti i tipi di pseudo-inverse, quella di Moore-Penrose è quella che ha la norma minima, inoltre essa ha anche le seguenti proprietà:

· se $\mathbf{X}$ è quadrata e non singolare allora esiste $\mathbf{X}^{-1}$ e si ha:

$$\mathbf{X}^{+} = \mathbf{X}^{-1}$$

· l'operazione di pseudo-inversione commuta con le operazioni di trasposizione, coniugazione e coniugazione hermitiana:

$$(\mathbf{X}^{T})^{+} = (\mathbf{X}^{+})^{T}$$
$$(\overline{\mathbf{X}})^{+} = \overline{(\mathbf{X}^{+})}$$
$$(\mathbf{X}^{\dagger})^{+} = (\mathbf{X}^{+})^{\dagger}$$

· la pseudo-inversione è reversibile con un altra pseudo-inversione:

$$(\mathbf{X}^{+})^{+} = \mathbf{X}$$

· la pseudo-inversa di un multiplo scalare a di $\mathbf{X}$ è uguale al reciproco di a per $\mathbf{X}^{+}$:

$$(a\,\mathbf{X})^{+} = a^{-1}\mathbf{X}^{+}, \quad \forall a \neq 0$$

· La pseudo-inversa della matrice nulla è uguale alla sua trasposta:

$$\mathbf{0}^{+} = \mathbf{0}^{T}$$

· Le pseudo-inverse di $\mathbf{X}^{\dagger}\mathbf{X}$ e $\mathbf{X}\,\mathbf{X}^{\dagger}$ possono essere utilizzate per calcolare $\mathbf{X}^{+}$:

$$\mathbf{X}^{+} = (\mathbf{X}^{\dagger}\mathbf{X})^{+}\mathbf{X}^{\dagger}$$
$$\mathbf{X}^{+} = \mathbf{X}^{\dagger}(\mathbf{X}\,\mathbf{X}^{\dagger})^{+}$$

· Dato che $\mathbf{X}\,\mathbf{X}^+$ e $\mathbf{X}^+\mathbf{X}$ sono entrambi hermitiani si ha:

$$\mathbf{X}^+\mathbf{X} = (\mathbf{X}^\dagger\mathbf{X})^+\,\mathbf{X}^\dagger\mathbf{X} = \mathbf{X}^\dagger\mathbf{X}\,(\mathbf{X}^\dagger\mathbf{X})^+$$
$$\mathbf{X}\,\mathbf{X}^+ = \mathbf{X}\,\mathbf{X}^\dagger\,(\mathbf{X}\,\mathbf{X}^\dagger)^+ = (\mathbf{X}\,\mathbf{X}^\dagger)^+\,\mathbf{X}\,\mathbf{X}^\dagger$$

· La pseudo-inversa di un vettore è uguale al vettore aggiunto diviso per la sua norma al quadrato:

$$\mathbf{x}^+ = \begin{cases} \mathbf{0}^T & \text{se } \mathbf{x} = \mathbf{0}; \\ \frac{\mathbf{x}^\dagger}{\mathbf{x}^\dagger\mathbf{x}} & \text{altrimenti.} \end{cases}$$

· Se il prodotto $\mathbf{X}\,\mathbf{Y}$ è definito, allora:

$$(\mathbf{X}\,\mathbf{Y})^+ = \begin{cases} \mathbf{Y}^+\mathbf{X}^+ & \text{per } \mathbf{X} \text{ o } \mathbf{Y} \text{ unitarie;} \\ (\mathbf{X}^+\mathbf{X}\,\mathbf{Y})^+(\mathbf{X}\,\mathbf{Y}\,\mathbf{Y}^+)^+ & \text{altrimenti.} \end{cases}$$

un'altra espressione utilizzabile è la seguente [30]:
$(\mathbf{X}\,\mathbf{Y})^+ = \mathbf{Y}^\dagger(\mathbf{X}^\dagger\mathbf{X}\,\mathbf{Y}\,\mathbf{Y}^\dagger)^+\mathbf{X}^\dagger$

Capitolo 3

Le reti neurali

3.1 Caratteristiche generali

Una *rete neurale artificiale* è formata da un insieme di *unità di elaborazione* chiamate *nodi* o *neuroni*, ciascuna in grado di calcolare la funzione che le è stata assegnata ad arbitrio nella fase di impostazione. Tali unità sono collegate tra loro tramite *connessioni* che trasportano dei *dati* rappresentabili mediante valori reali; ad ogni connessione è generalmente associato un coefficiente reale chiamato *weight* o *peso*, spesso indicato con w, corrispondente al fattore di moltiplicazione che si applica ai dati immessi in essa. Quando ad un nodo arrivano più connessioni di norma esso calcola la somma dei vari input e vi applica la funzione preimpostata, il risultato poi è trasmesso dalle connessioni di output. Indicando con θ l'eventuale *bias* o *offset*, lo schema tipico di un nodo è il seguente:

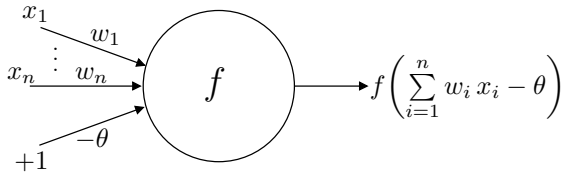

Figura 3.1: Schema di un neurone artificiale

Una rete neurale è un sistema intrinsecamente parallelo, dato che le diverse unità possono effettuare contemporaneamente le rispettive operazioni; per la sua programmazione comunemente si utilizzano degli esempi che, tramite opportuni algoritmi, provocano la modificazione dei pesi delle connessioni e quindi l'apprendimento. Tale meccanismo è stato ispirato dall'introduzione di un modello semplificato di neurone da parte di Warren McCulloch e Walter Pitts nel 1943.

Una rete neurale è caratterizzata dai seguenti elementi:

- la struttura dei nodi,
- la topologia della rete,
- l'algoritmo di apprendimento utilizzato per modificare i pesi delle connessioni.

Se i pesi delle connessioni sono regolabili, la loro variazione può avvenire in modo *sincrono* o *asincrono*, nel primo caso tutti i pesi sono modificati attraverso cicli di aggiornamento, nel secondo caso invece ogni unità ha una certa probabilità, spesso costante, di subire una modifica, per cui l'aggiornamento risulta un processo stocastico.

I vari punti che caratterizzano le reti neurali saranno trattati nelle sezioni seguenti:

3.2 La struttura dei nodi

La struttura dei nodi dipende da molti parametri:

- dal tipo di codifica dei dati
- dal modo con cui si combinano i valori di ingresso
- dalla scelta delle *funzioni di attivazione* f
- dalla presenza di soglie regolabili
- dalla esistenza di memorie interne al nodo

3.2.1 La codifica dei dati

I *dati* possono essere rappresentati mediante numeri reali qualunque, ma per l'output dei nodi spesso si impiegano solo i reali tra 0 e 1 o tra -1 e 1, oppure codifiche discrete, che di solito si adattano meglio all'hardware disponibile e consentono un miglior controllo degli errori strumentali. A volte sono utilizzati i valori binari 0 e 1, più spesso invece si ricorre alla codifica bipolare basata sui valori -1 e 1, poiché presenta numerosi vantaggi, come si può notare dal confronto diretto tra i due sistemi:

Dato un vettore v formato da n singole cifre binarie casuali, indicando con p_1 la probabilità costante di ciascun elemento di essere 1 si ha che:

- I possibili valori della norma di v hanno una distribuzione binomiale con media $\sqrt{(n\,p_1)}$ e varianza $\sqrt{n\,p_1\,(1-p_1)}$
- Il prodotto scalare di due vettori v_1 e v_2 del tipo suddetto avrà in media il valore: $(n\,p_1^2)$

Nel caso in cui il vettore v sia formato da n singole cifre bipolari casuali, indicando ancora con p_1 la probabilità costante di ciascun elemento di essere 1 si ha che:

- Ogni vettore v ha la stessa norma corrispondente a $\sqrt{n}$, quindi tutti i possibili stati si trovano in punti discreti sulla superficie della ipersfera n dimensionale di raggio $\sqrt{n}$
- Il prodotto scalare di due vettori bipolari casuali di n elementi: $v_1 \cdot v_2$ avrà mediamente il valore:

  $$n\,(p_1^2 + (1-p_1)^2 - 2\,p_1(1-p_1)) = n\,(1-2\,p_1)^2$$

  per cui se le cifre -1 e 1 sono equiprobabili, cioè $p_1 = \frac{1}{2}$, il prodotto scalare sarà in media nullo e quindi i vettori risulteranno con buona probabilità ortogonali o quasi; l'ortogonalità è una caratteristica molto importante poiché facilita la convergenza dei pesi e l'apprendimento degli schemi.

· Poiché gli stati sono simmetrici, in reti dotate di simmetria come quelle chiamate memorie associative, se un dato stato bipolare è stabile lo è anche lo stato che si ottiene scambiando 1 e -1, mentre utilizzando la codifica binaria ciò non è sempre vero, ad esempio lo stato formato da tutti 0 è sempre stabile mentre quello costituito da tutti 1 non è detto che lo sia.

· Le evidenze sperimentali confermano che utilizzando dati codificati in modo bipolare si ha una convergenza più rapida verso uno stato stabile.

Nel grafico 3.2 sono rappresentati i valori medi del prodotto scalare di due vettori casuali, in funzione della probabilità di ogni elemento di essere uguale ad 1; la linea più scura si riferisce al caso binario mentre quella più chiara al caso bipolare.

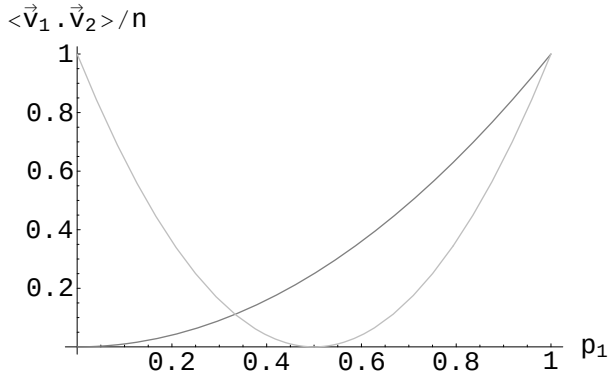

Figura 3.2: Prodotto scalare tra vettori casuali binari e bipolari

3.2.2 Le regole di combinazione degli input

Normalmente i valori di input che arrivano ad un nodo sono sommati tra di loro prima dell'applicazione della funzione di attivazione; le unità che obbediscono a tale regola di propagazione sono chiamate *sigma units*.

Indicando con:

- $s_k(t)$ l'input totale al tempo t
- w_{jk} il peso del collegamento che va dal nodo j a quello k
- y_j l'output del neurone j
- θ_k il termine di bias relativo al nodo k

si ha che:

$$s_k(t) := \sum_j w_{jk}(t)\, y_j(t) - \theta_k(t)$$

I valori di input di un neurone sono classificati generalmente in due gruppi in base al loro segno: quelli positivi sono chiamati *eccitatori*, mentre i valori negativi si dicono *inibitori*.

Non tutti gli algoritmi trattano i segnali inibitori allo stesso modo: si ha una *inibizione relativa* quando i segnali negativi sono elaborati come gli altri, e una *inibizione assoluta* quando essi provocano nei nodi che li ricevono lo stesso effetto che si avrebbe se tutti gli input fossero disattivi.

Oltre alla regola di propagazione riportata sopra ne esistono altre che oggi sono meno usate, per esempio quella introdotta da Feldman e Ballard nel 1982 che dà il nome di *sigma-pi units* ai nodi che la adottano, indicando con y_{j_i} l'i-esima componente dell'output del neurone j risulta:

$$s_k(t) := \sum_j w_{jk}(t) \prod_i y_{j_i}(t) - \theta_k(t)$$

3.2.3 La funzione di attivazione

La funzione di attivazione f può essere arbitraria, spesso però si ricorre a delle funzioni standard, diverse a seconda che si utilizzi una codifica binaria o bipolare; talvolta il valore di output di tale funzione è chiamato *valore di attivazione*.

In genere durante la fase di training di una rete si utilizzano funzioni smooth, una volta che sono stati determinati i pesi corretti è possibile utilizzare anche delle funzioni a gradino.

Le funzioni più usate nel caso binario sono le seguenti:

· La funzione gradino di Heaviside:

$$\Theta(x) := \begin{cases} 1 & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

· La sigmoide di equazione: $\dfrac{1}{1+\mathrm{e}^{-x}}$

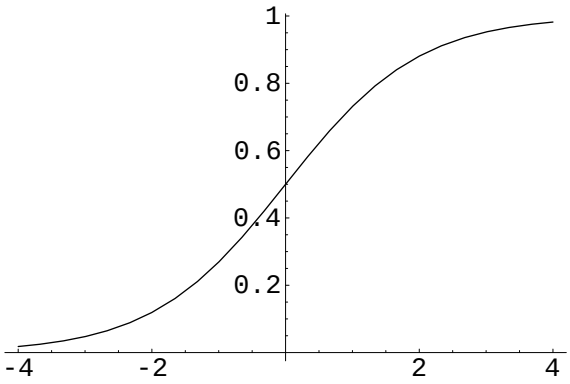

Figura 3.3: Funzione sigmoide

Le funzioni più comuni nel caso di codifica bipolare sono:

· La funzione segno:

$$\mathrm{Sign}(x) := \begin{cases} 1 & \text{se } x > 0 \\ 0 & \text{se } x = 0 \\ -1 & \text{se } x < 0 \end{cases}$$

· La tangente iperbolica:

$$\mathrm{Tanh}(x) = \frac{\mathrm{Sinh}(x)}{\mathrm{Cosh}(x)} = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}$$

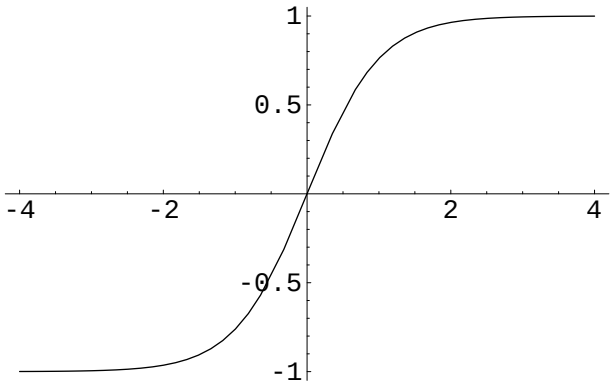

Figura 3.4: Funzione tangente iperbolica

3.3 La topologia delle reti

Per la topologia della rete è utile distinguere tre tipi di unità:

· le *unità di input*, indicate con un indice i, che ricevono i dati dall'esterno della rete neurale,

· le *unità di output*, identificate con un indice o, che inviano i risultati dell'elaborazione all'esterno della rete stessa,

· le *unità nascoste* o *hidden*, evidenziate a volte con un indice h, che sono collegate solo con altre unità della rete e non con l'esterno.

Le reti neurali in funzione della topologia sono classificate in:

· *Feed-forward networks:* In esse i nodi sono usualmente disposti in strati successivi, chiamati *layer* ed i dati fluiscono dal layer delle unità di input a quello delle unità di output senza *connessioni di*

feedback, cioè connessioni che dall'output di un'unità si dirigono all'input di altre unità dello stesso layer o di layer precedenti. Abitualmente in questo tipo di reti il layer di input non è conteggiato poiché non effettua alcuna operazione, trasferisce solo i dati alle unità dello strato successivo; se una rete è formata da: un layer di input, un hidden e uno di output si dice quindi che essa ha due layer. Un esempio tipico di rete feed-forward è il *perceptron* di cui si parlerà nella sezione 3.5.

- *Recurrent networks:* Esse contengono connessioni di feedback per cui, a differenza del caso precedente, è importante considerare le proprietà dinamiche della rete, in alcuni casi i valori di output delle unità subiscono un processo di rilassamento che fa evolvere il sistema verso uno stato stabile in cui non si hanno più variazioni, in altri casi invece il sistema risulta instabile e i suoi valori sono soggetti a continue oscillazioni. Mediante l'uso di percorsi di feedback è possibile immagazzinare nella rete delle informazioni relative a output precedenti, in modo da poterle utilizzare successivamente, in tal caso si parla a volte di *automati a stati finiti*.

3.4 Le regole di aggiornamento dei pesi

La maggior parte delle regole di aggiornamento utilizzate oggi derivano in qualche modo dalla *Hebbian learning rule* descritta da Hebb nel suo libro *Organization of Behaviour* del 1949. Tale regola si basa sul principio che se due neuroni sono contemporaneamente attivi allora il collegamento che li unisce deve essere rafforzato. Indicando con w_{ik} il peso del collegamento che va dal nodo i a quello k e con x_i e y_k i rispettivi output, la forma più semplice della regola di Hebb prescrive che tale peso debba essere modificato nel modo seguente:

$$\Delta w_{ik} := \gamma \, x_i \, y_k$$

avendo indicato con k una costante positiva chiamata *learning rate*.

Oggi una delle regole di aggiornamento più usate è la *delta rule* chiamata anche *Widrow-Hoff rule* che sarà analizzata più ampiamente nella sezione 3.8 ma di cui si può già fornire lo schema di base.

Supponendo di conoscere il valore di attivazione, cioè di output, che dovrebbe avere un dato nodo k, perché per esempio tale valore d_k è stato fornito da un teacher esterno, allora è possibile modificare tutti i collegamenti diretti a tale nodo applicando la regola:

$$\Delta w_{ik} := \gamma\, x_i (d_k - y_k)$$

3.5 Il perceptron

Il perceptron è stato proposto da Rosenblatt nel 1958 [34], ed il suo obiettivo è apprendere un trasformazione $d : \{-1, 1\}^n \to \{-1, 1\}$ utilizzando per il training degli esempi $\mathbf{x}$, a cui si vuole che corrisponda l'output $y = d(\mathbf{x})$.

Nella sua versione iniziale gli n input erano collegati ad uno layer intermedio formato da m nodi con funzioni arbitrarie ϕ_h, i quali avevano il ruolo di associatori, a loro volta collegati al nodo di uscita con output $\{-1, 1\}$; purtroppo però per tale modello non è stata trovata alcuna regola che permettesse di aggiustare i pesi tra il layer di input e quello intermedio.

3.5.1 Interpretazione geometrica di un perceptron

Un perceptron con due connessioni di input e una soglia può essere rappresentato come nella figura 3.5, dove la funzione di attivazione f è la funzione Sign, per cui è possibile utilizzare i suoi due valori di output $+1$ e -1 per ripartire gli input in due classi distinte.

Se il valore totale dell'input:

$$s := w_1\, x_1 + w_2\, x_2 - \theta$$

è positivo, all'elemento (x_1, x_2) sarà assegnata la classe $+1$, se invece s è negativo la classe -1.

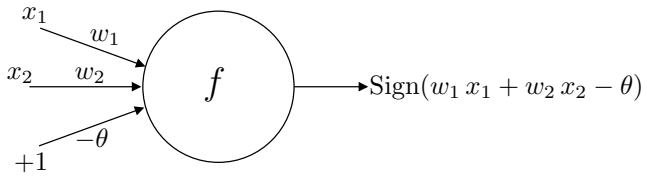

Figura 3.5: Schema di un perceptron

La linea di separazione tra le due classi è la retta di equazione:

$$w_1\,x_1 + w_2\,x_2 - \theta = 0$$

che può essere riscritta esplicitando x_2:

$$x_2 = -\frac{w_1}{w_2}\,x_1 + \frac{\theta}{w_2}$$

Analizzando tale espressione si può notare che i pesi w determinano il coefficiente angolare della retta discriminante, mentre il termine di bias θ determina l'offset, cioè la distanza della retta dall'origine degli assi; tali parametri sono evidenziati nella figura 3.6.

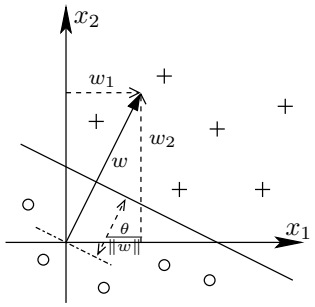

Figura 3.6: Rappresentazione geometrica del perceptron

La retta discriminante divide il piano in due semipiani, tutti i punti della figura che si trovano dalla parte verso cui punta il vettore w saranno classificati come $+1$, quelli che giacciono nell'altro semipiano saranno invece classificati con -1, in base al segno del prodotto scalare tra w e il vettore posizione r che individua il punto considerato.

3.5.2 La regola di apprendimento del perceptron

1. Si inizia attribuendo in modo casuale i pesi alle connessioni.

2. Si seleziona un vettore di input $\mathbf{x}$ del training set.

3. Se l'output del perceptron y è diverso da quello desiderato $d(\mathbf{x})$, con $d(\mathbf{x}) = \pm 1$, si procede alla modifica, di tutti i pesi w_i che vanno dall'i-esimo elemento x_i del vettore $\mathbf{x}$, al perceptron, secondo la regola: $\forall i \;\; \Delta w_i := d(\mathbf{x})\, x_i$.

4. Il bias può sempre essere visto come una connessione esterna come illustrato nella figura 3.1, si deve però tener conto del segno negativo davanti a θ, per cui la regola di modifica diventa: $\Delta\theta := -d(\mathbf{x})$.

5. Si torna al punto 2 fino a che l'analisi di tutti i punti del training set non provoca alcuna variazione dei parametri, per questo in generale è necessario utilizzare i dati di esempio più volte, in modo ciclico.

3.5.3 Esempio della regola di apprendimento

Si consideri il perceptron della figura 3.5 e si supponga di inizializzarlo con i valori: $w_1 := 1$, $w_2 := 2$, $\theta := 2$; la linea discriminante che ne risulta è raffigurata nel grafico 3.7 da una linea tratteggiata.

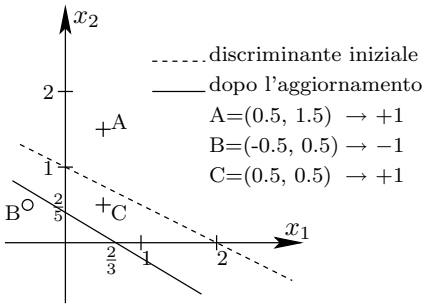

Figura 3.7: Linea discriminante prima e dopo l'aggiornamento.

Il training set è dato dai punti: $A \equiv (0.5, 1.5)$, $B \equiv (-0.5, 0.5)$ e $C \equiv (0.5, 0.5)$, a cui si vuole corrispondano rispettivamente le etichette: $+1$, -1, $+1$.

Per i primi due punti A e B, il perceptron che esegue l'operazione: $\text{Sign}(w_1\, x_1 + w_2\, x_2 - \theta)$, fornisce il risultato voluto, nel punto C però la risposta è sbagliata, infatti: $\text{Sign}(1 \cdot 0.5 + 2 \cdot 0.5 - 2) = -1$, mentre il valore desiderato per l'output: $d(\mathbf{x}) = +1$, occorre allora applicare la regola di aggiornamento dei pesi e del bias che è stata esposta prima:

$$\begin{aligned} \Delta w_1 &= d(\mathbf{x})\, x_1 = +1 \cdot x_1 = 0.5 \qquad \Rightarrow \qquad w_1 = 1.5 \\ \Delta w_2 &= d(\mathbf{x})\, x_2 = +1 \cdot x_2 = 0.5 \qquad \Rightarrow \qquad w_1 = 2.5 \\ \Delta\theta &= -d(\mathbf{x}) = -1 \qquad\qquad\quad \Rightarrow \qquad \theta = 1 \end{aligned}$$

Con i nuovi parametri la linea discriminante risulta essere quella indicata nel grafico con tratto continuo, e come si può notare essa separa i punti nel modo voluto.

3.5.4 Il teorema della convergenza

Per i perceptron esiste il seguente teorema sulla convergenza:

Teorema. *Se esiste un set di* connection weights $\mathbf{w}^*$ *che permette di realizzare la trasformazione* $y = d(\mathbf{x})$*, la regola di apprendimento del perceptron convergerà ad una soluzione, che non è detto sia* $\mathbf{w}^*$*, in un numero finito di iterazioni, per qualunque scelta dei pesi iniziali* $\mathbf{w}_0$*.*

Dimostrazione. Affinché possa esistere un $\mathbf{w}^*$ che permette di classificare i punti nel modo voluto essi devono essere linearmente separabili, in tal caso deve esistere un $\delta > 0$ tale che ogni punto da classificare deve avere una distanza dalla retta discriminante in modulo maggiore di δ per cui:

$$|\mathbf{w}^* \cdot \mathbf{x}| > \delta$$

Dato che l'attribuzione della classe ai singoli punti avviene considerando il segno del prodotto scalare tra $\mathbf{w}^*$ e il raggio vettore $\mathbf{x}$ che

individua il punto in esame, risulta che la lunghezza del vettore $\mathbf{w}^*$, purché sia maggiore di zero, è ininfluente, per cui si può considerare che tale vettore abbia norma unitaria: $\|\mathbf{w}^*\| := 1$.

Indicando con $\mathbf{w}$ un generico vettore di pesi si può definire la seguente quantità:

$$\cos(\alpha) := \frac{\mathbf{w} \cdot \mathbf{w}^*}{\|\mathbf{w}\|}$$

Se un dato punto individuato dal vettore $\mathbf{x}$ è classificato in modo errato occorre aggiornare i pesi applicando la regola: $\Delta\mathbf{w} = d(\mathbf{x})\,\mathbf{x}$; dopo la modifica si avrà: $\mathbf{w}' = \mathbf{w} + \Delta\mathbf{w}$, da cui segue:

$$\begin{aligned}
\mathbf{w}' \cdot \mathbf{w}^* &= \mathbf{w} \cdot \mathbf{w}^* + d(\mathbf{x})\,\mathbf{x} \cdot \mathbf{w}^* = \\
&= \mathbf{w} \cdot \mathbf{w}^* + \mathrm{Sign}(\mathbf{w}^* \cdot \mathbf{x})\,\mathbf{x} \cdot \mathbf{w}^* = \\
&= \mathbf{w} \cdot \mathbf{w}^* + |\mathbf{w}^* \cdot \mathbf{x}| = \\
&> \mathbf{w} \cdot \mathbf{w}^* + \delta
\end{aligned}$$

per quanto riguarda il calcolo della norma di $\mathbf{w}'$ bisogna tener conto che il perceptron deve aver effettuato una classificazione sbagliata del punto $\mathbf{x}$, altrimenti non ci sarebbe stato bisogno di aggiornare i pesi, quindi in tal caso $\mathrm{Sign}(\mathbf{w} \cdot \mathbf{x}) = -d(\mathbf{x})$ e si ha:

$$\begin{aligned}
\|\mathbf{w}'\|^2 &= \|\mathbf{w} + d(\mathbf{x})\,\mathbf{x}\|^2 = \\
&= \mathbf{w}^2 + 2\,d(\mathbf{x})\,\mathbf{w} \cdot \mathbf{x} + \mathbf{x}^2 = \\
&= \mathbf{w}^2 - 2\,\mathrm{Sign}(\mathbf{w} \cdot \mathbf{x})\,\mathbf{w} \cdot \mathbf{x} + \mathbf{x}^2 = \\
&= \mathbf{w}^2 - 2\,|\mathbf{w} \cdot \mathbf{x}| + \mathbf{x}^2 = \\
&< \mathbf{w}^2 + \mathbf{x}^2
\end{aligned}$$

Indicando con $\mathbf{x}_{max}$ il punto del training set S avente la norma più grande risulta che la seguente disuguaglianza è sempre verificata:

$$\|\mathbf{w}'\|^2 < \mathbf{w}^2 + \mathbf{x}_{max}^2$$

Dopo t aggiornamenti dei pesi iniziali $\mathbf{w}_0$ si avrà:

$$\begin{aligned}
\mathbf{w}_0(t) \cdot \mathbf{w}^* &> \mathbf{w}_0 \cdot \mathbf{w}^* + t\,\delta \\
\|\mathbf{w}_0(t)\|^2 &< \mathbf{w}_0^2 + t\,\mathbf{x}_{max}^2
\end{aligned}$$

per cui:

$$\begin{aligned} \cos(\alpha(t)) &:= \frac{\mathbf{w}_0 \cdot \mathbf{w}^*}{\|\mathbf{w}_0\|} \\ &> \frac{\mathbf{w}_0 \cdot \mathbf{w}^* + t\,\delta}{\sqrt{\mathbf{w}_0^2 + t\,\mathbf{x}_{max}^2}} \end{aligned}$$

Da tale espressione si ricava che:

$$\lim_{t\to\infty} \cos(\alpha(t)) = \lim_{t\to\infty} \frac{\delta}{\sqrt{\mathbf{x}_{max}^2}}\sqrt{t} = \infty$$

Dato che per definizione $\cos(\alpha) \leq 1$ deve esistere un limite massimo t_{max} al numero degli aggiornamenti che il sistema deve compiere prima di arrivare ad una condizione di equilibrio dei pesi delle connessioni. Se il vettore di partenza è il vettore nullo: $\mathbf{w}_0 = \mathbf{0}$, il numero massimo di modifiche da apportare ai pesi è:

$$t_{max} < \frac{\mathbf{x}_{max}^2}{\delta^2}$$

□

3.6 Le limitazioni del perceptron

3.6.1 La linearità

Uno dei problemi principali messi in evidenza da Minsky e Papert è che il perceptron essendo una funzione discriminante lineare non può rappresentare la funzione XOR detto anche OR esclusivo.

Nel grafico 3.8 sono mostrate le funzioni logiche binarie AND, OR, XOR; in esso i valori di input veri sono indicati con 1, e quelli falsi con -1, per l'output invece si sono utilizzati dischi di colore differente: il nero rappresenta un valore vero, mentre il disco bianco un valore falso. La linea tracciata corrisponde ad una delle possibili posizioni in cui può essere collocata la linea discriminante.

Le tabelle di verità degli operatori AND, OR e XOR sono:

AND

x_1	x_2	d
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

OR

x_1	x_2	d
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

XOR

x_1	x_2	d
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

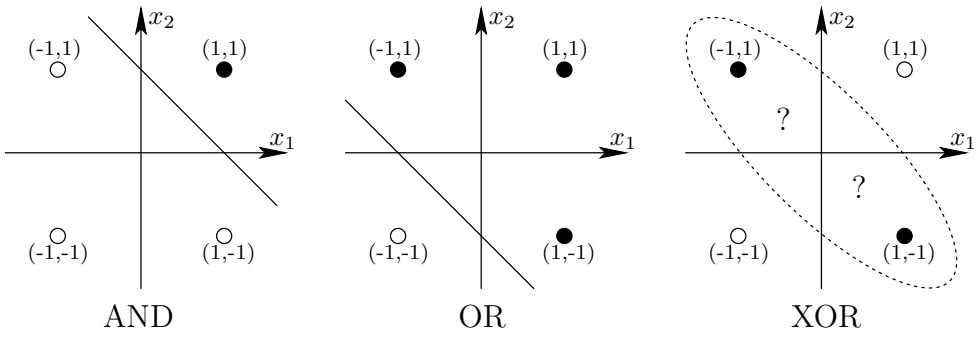

Figura 3.8: Rappresentazione geometrica degli operatori.

Dal grafico risulta chiaro che nel caso dello XOR non è possibile separare i dischi neri da quelli bianchi utilizzando una linea retta.

In realtà una soluzione a tale problema esiste, infatti Minsky e Papert hanno provato nel loro libro che per input binari è possibile effettuare una trasformazione qualunque aggiungendo un layer di neuroni collegati a tutti gli input; tale rete prende il nome di *multi-layer perceptron*.

La prova di tale teorema sarà fornita nella sezione 3.7

Il problema dello XOR aggiungendo un *neurone nascosto* diventa risolubile; dal punto di vista geometrico equivale ad immergere i quattro punti in uno spazio tridimensionale definito dai due input e dall'output del neurone aggiunto, in questo modo essi possono essere facilmente separati da un piano.

La figura 3.9 mostra come sia possibile realizzare tale separazione:

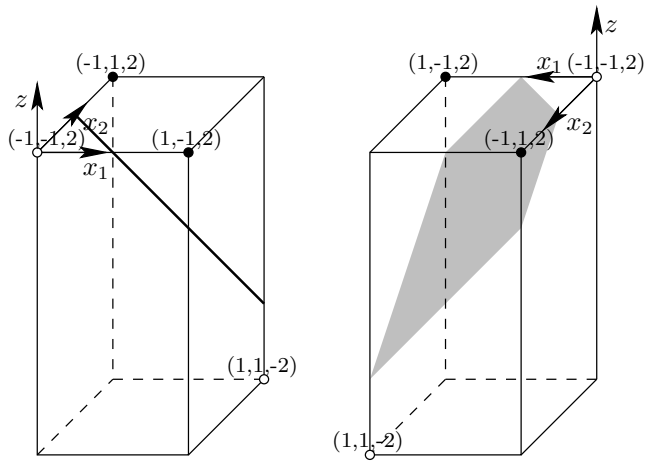

Figura 3.9: Soluzione del problema dello XOR da diverse prospettive.

La rete neurale corrispondente al grafico precedente è rappresentata nella figura 3.10; in essa i nodi contengono delle soglie, indicate al loro interno, se la somma dei valori di input che arrivano ad essi supera la soglia il corrispondente output sarà $+1$, in caso contrario si avrà -1.

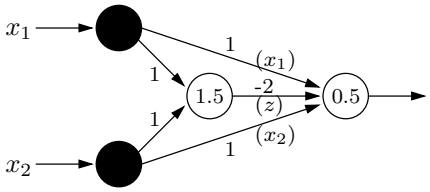

Figura 3.10: Rete neurale per lo XOR.

La soluzione del problema dello XOR non è unica infatti, ad esempio, anche la rete della figura 3.11 fornisce gli output desiderati.

Esistono inoltre delle reti che non prevedono collegamenti diretti tra il primo layer e quello di output, esse però richiedono l'impiego di più nodi di elaborazione.

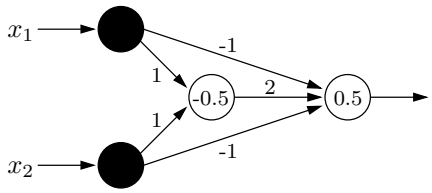

Figura 3.11: Differente rete neurale per lo XOR.

3.6.2 I limiti del parallelismo

Il perceptron nella sua forma originale deve i suoi limiti principalmente alla presenza di un unico nodo di output dotato di soglia, infatti anche se le unità che precedono tale nodo avessero una capacità di elaborazione illimitata, esse dovrebbero in qualche modo *cooperare* tra loro affinché i singoli risultati forniti possano risultare pertinenti alla decisione globale che deve essere assunta dal nodo di uscita.

Il problema può essere visto quindi nella più ampia cornice che riguarda i limiti del parallelismo, è noto infatti che quando si cerca di parallelizzare degli algoritmi, si può avere una componente sequenziale ineliminabile che limita lo *speedup* massimo ottenibile. La relazione matematica tra lo speedup e la porzione sequenziale irriducibile è nota come legge di Amdahl.

Se la risoluzione di un problema richiedesse necessariamente la presenza di una tale componente sequenziale allora il perceptron non sarebbe utilizzabile essendo il suo schema totalmente parallelo.

Determinare se una figura geometrica è connessa o meno utilizzando dei rilevatori che possono analizzare ognuno solo una parte limitata della figura stessa è un problema che non può essere in generale risolto da un perceptron, e ciò può essere dimostrato:

Teorema. *Nessun perceptron di diametro inferiore ad un valore minimo dipendente dalla figura geometrica analizzata può decidere se quest'ultima è connessa o no, indipendentemente dal numero di rilevatori utilizzati.*

Dimostrazione. La prova sarà per contraddizione, per cui si inizierà ipo-

tizzando che i perceptron possano decidere se una figura geometrica sia connessa o meno, indipendentemente dalla loro dimensione, per poi arrivare ad un risultato che falsifichi l'ipotesi.

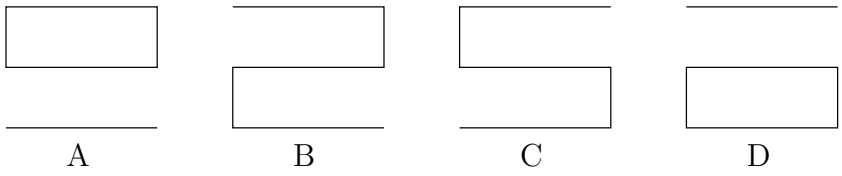

Figura 3.12: Figure utilizzate per la dimostrazione

Per fissare le idee si utilizzeranno le forme geometriche contenute nella figura 3.12, delle quali solo le due centrali sono connesse, e si supporrà che nessun singolo rilevatore sia esteso abbastanza da includere entrambi i bordi sinistro e destro, dove si trovano le discontinuità, in tal caso i singoli neuroni possono essere classificati nei tre gruppi seguenti:

- Si definiscono del *gruppo 1* i neuroni che nella loro area di rilevazione includono il bordo sinistro della figura in esame.
- Si definiscono del *gruppo 2* i neuroni che controllano il bordo destro
- Si definiscono del *gruppo 3* tutti gli altri nodi.

tali gruppi sono illustrati graficamente nella figura 3.13.

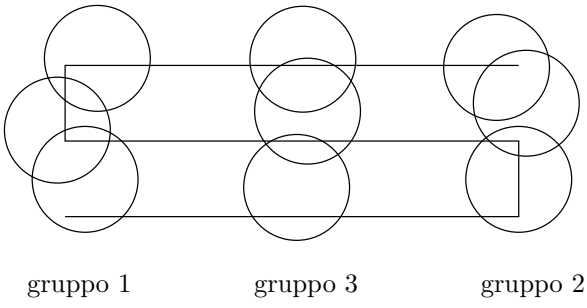

Figura 3.13: Gruppi di neuroni e relative aree di rilevazione

La risposta se una figura è connessa o no dovrà essere fornita dal nodo di output, mediante la determinazione del segno dell'input totale S che arriva a tale nodo; come al solito il valore della soglia o bias potrà essere considerato come un input esterno $+1$ associato ad un peso $-\theta$. Con w_{1_i} si indicherà il peso della connessione che arriva al nodo di output partendo dall'i-esimo neurone del gruppo 1, avente come output y_{1_i}; una notazione analoga si impiegherà per gli elementi appartenenti ai gruppi 2 e 3.

Con tali convenzioni l'input totale dell'perceptron è:

$$S := \sum_{y_i \in gruppo\,1} w_{1_i}\, y_{1_i} + \sum_{y_j \in gruppo\,2} w_{2_j}\, y_{2_j} + \sum_{y_k \in gruppo\,3} w_{3_k}\, y_{3_k} - \theta$$

Si supponga inoltre che la rete sia configurata in modo da fornire un risultato $y_{out} := \mathrm{Sign}(S) = +1$ se la figura è connessa e $y_{out} = -1$ se non lo è, ciò implica che l'input totale S dovrà essere nel primo caso positivo e nel secondo negativo.

Supporre che la rete sia configurata significa che i pesi e i bias sono tutti fissati, e gli esempi sono classificati tutti nel modo corretto.

Ora che sono stati presentati tutti gli elementi necessari è possibile iniziare la sequenza logica che porterà alla contraddizione:

- Poiché si suppone che la rete possa discriminare le forme geometriche connesse della figura 3.12, per la figura A si dovrà avere $S_A < 0$.

- Essendo la figura B connessa si dovrà avere $S_B > 0$. Per passare dalla figura A alla figura B si deve modificare un solo segmento sul lato sinistro, per cui gli unici nodi interessati saranno quelli appartenenti al gruppo 1, indicando con $\Delta_1 S_A$ la variazione subita da tali nodi si avrà:

$$S_B = S_A + \Delta_1 S_A > 0 \quad \Rightarrow \quad \Delta_1 S_A > -S_A$$

- Anche la figura C è connessa per cui $S_C > 0$. Per trasformare la figura A nella figura C occorre modificare un solo segmento sul lato

destro, quello dei nodi del gruppo 2, per cui si avrà:

$$S_C = S_A + \Delta_2 S_A > 0 \quad \Rightarrow \quad \Delta_2 S_A > -S_A$$

· La figura D non è connessa per cui si deve avere $S_D < 0$, inoltre essa può essere ottenuta partendo dalla figura A applicando entrambe le modifiche precedenti, una sul lato sinistro e l'altra sul destro. Poiché per ipotesi nessun nodo può includere entrambi i lati modificati, per i neuroni del gruppo 1 le figure B e D sono indistinguibili, per i nodi del gruppo 2 le figure C e D sono identiche, per quelli del gruppo 3 tutti i casi sono uguali, da ciò consegue che:

$$S_D = S_A + \Delta_1 S_A + \Delta_2 S_A$$

sommando le disuguaglianze ottenute per i casi B e C si ha che:

$$\Delta_1 S_A + \Delta_2 S_A > -2\, S_A$$

inserendo tale risultato nell'espressione di S_D si ricava:

$$S_D > S_A - 2\, S_A = -S_A > 0 \quad \text{essendo} \;\; S_A < 0$$

Partendo dall'ipotesi che il perceptron fosse in grado di distinguere le figure connesse da quelle che non lo sono, indipendentemente dalla superficie analizzata dai vari nodi, si è giunti ad una contraddizione poiché il punto D non può essere classificato nel modo corretto, quindi il teorema risulta dimostrato.

□

L'unico modo per eliminare tale contraddizione è richiedere che il caso D risulti differente dal caso A corretto con le variazioni che portano alle figure B e C, per cui deve esistere almeno un perceptron che agisca su entrambi i lati modificati oppure ci deve essere una qualche forma di coordinazione tra le unità del gruppo 1 e del gruppo 2, o una analisi sequenziale degli output dei vari rilevatori.

Dalla prova del teorema precedente consegue chiaramente che la proprietà di connessione è una proprietà globale che non può essere decisa localmente.

In modo analogo si può provare per esempio che un perceptron non è in grado di dire se un insieme di punti è pari o dispari se i rivelatori coprono solo parti limitate dell'insieme.

3.7 Il multi-layer perceptron

Aggiungendo un hidden layer è stato risolto il problema dello XOR. In realtà è possibile provare che per unità binarie o bipolari il layer nascosto permette di risolvere qualunque tipo di trasformazione, fissando in modo opportuno le connessioni e i pesi.

Teorema. *Ogni generica funzione a valori binari o bipolari* $y = f(\mathbf{x})$*, con* $f : \{-1, 1\}^n \to \{-1, 1\}$ *o* $f : \{0, 1\}^n \to \{0, 1\}$ *può essere calcolata con un perceptron avente un layer nascosto.*

Dimostrazione. I vettori di input $\mathbf{x}$ possono sempre essere raggruppati in due classi a seconda del risultato della funzione $f(\mathbf{x})$:

$$X^+ := \{\mathbf{x} \mid f(\mathbf{x}) = 1\} \quad \text{e} \quad X^- := \{\mathbf{x} \mid f(\mathbf{x}) \neq 1\}$$

Dato che tutti i vettori $\mathbf{x}$ hanno lunghezza n il numero totale dei possibili input è 2^n.

Una rete neurale per calcolare la generica funzione $y = f(x)$ può essere realizzata nel modo seguente: per ogni vettore appartenente ad X^+ si colloca un neurone nel layer nascosto e lo si collega da una parte a tutti gli input, e dall'altra al nodo di output, caratterizzato come al solito da un bias e da un output binario o bipolare. Si procede poi alla configurazione dei pesi dei collegamenti, iniziando da quelli che partono dai nodi di input, in modo che ciascuna unità nascosta i sia in grado di riconoscere un solo vettore di input $\mathbf{x}^{(i)} \in X^+$, e che ogni neurone hidden risulti associato ad un diverso $\mathbf{x} \in X^+$.

La configurazione di tali pesi può essere effettuata come segue:

· Indicando con w_{ij} il peso del collegamento dall'unità di input i all'unità nascosta j, e con $\mathbf{x}_i^{(j)}$ la i-esima componente del vettore di input $\mathbf{x}$, destinato alla memorizzazione nel nodo j, si possono operare le seguenti definizioni:

$$w_{ij} := \mathbf{x}_i^{(j)} \quad \text{e} \quad \theta := n - \tfrac{1}{2}$$

si ha in tal modo che l'output del j-esimo neurone hidden y_{h_j} è:

$$y_{h_j} := \text{Sign}\left(\sum_{i=1}^{n} w_{ij}\, \mathbf{x}_i^{(j)} - n + \tfrac{1}{2}\right)$$

Se a tale nodo si presenta un generico vettore di input $\widetilde{\mathbf{x}}$, l'output che si ottiene è:

$$y_{h_i} = \text{Sign}\left(\sum_{i=1}^{n} w_{ij}\, \widetilde{\mathbf{x}}_i - n + \tfrac{1}{2}\right) = \text{Sign}\left(\sum_{i=1}^{n} \mathbf{x}_i^{(j)} \widetilde{\mathbf{x}}_i - n + \tfrac{1}{2}\right)$$

analizzando l'argomento della funzione Sign si può notare che esso è positivo se e solo se è soddisfatta la condizione:

$$\forall i \quad \widetilde{\mathbf{x}}_i = \mathbf{x}_i^{(j)}$$

che è proprio quella richiesta per il riconoscimento del vettore $\mathbf{x}^{(j)}$.

Il neurone di uscita, collegato a tutti gli m nodi nascosti che corrispondono agli elementi $\mathbf{x} \in X^+$, dovrà fornire l'output 1 se e solo se $\mathbf{x} \in X^+$; per ottenere tale risultato non dovrà fare altro che verificare se in una delle sue connessioni di input è presente il valore $+1$. Fissando tutti i pesi tra le unità del layer nascosto e il neurone di output al valore $+1$, la funzione di attivazione da attribuire al nodo di uscita può essere scritta come:

$$y_{out} := \text{Sign}\left(\sum_{i=1}^{m} y_{h_i} + m - \tfrac{1}{2}\right)$$

Data una generica funzione $f(\mathbf{x})$ con la procedura suddetta è sempre possibile costruire una rete neurale del tipo multi-layer perceptron che fornisca i risultati desiderati, e ciò dimostra il teorema. □

Il problema principale di tale procedura è connesso all'alto numero di nodi che devono essere inseriti nel layer nascosto, con il procedimento presentato nel teorema precedente, per esempio, essi possono arrivare a 2^n. In realtà si potrebbe sempre scegliere l'insieme più piccolo tra X^+ e X^- e nel caso risulti X^- invertire il segno dell'output, in tal modo il numero massimo di unità si ridurrebbe a 2^{n-1}. Per trasformazioni complesse è stato dimostrato comunque che, anche applicando altri metodi, i neuroni hidden richiesti sono esponenziali in n.

3.8 La delta rule

Un importante generalizzazione della regola di learning del perceptron è la cosiddetta *delta rule* presentata da Widrow e Hoff col nome di procedura di apprendimento dei minimi quadrati (*'least mean square'* learning procedure (LMS)).

Essa si applica alle reti neurali con un singolo layer aventi come unità di output una funzione di attivazione lineare:

$$y := \sum_i w_i \, x_i - \theta$$

tali reti sono in grado di rappresentare solo relazioni lineari tra i valori di input e di output, inoltre aggiungendo un elemento a soglia al nodo di uscita si possono ottenere valori binari o bipolari, come nel caso del perceptron, così da poter essere utilizzate anche per compiti di classificazione.

In generale per valori di ingresso formati da molte componenti la rete può essere rappresentata geometricamente da un iperpiano in uno spazio multidimensionale.

Dato un training set formato dai valori di input $\mathbf{x}^p$ e dai valori desiderati di output d^p, modificando in modo opportuno i pesi delle connessioni si può trovare l'iperpiano che approssima nel modo migliore possibile i dati disponibili.

Il procedimento della *delta rule* per la modifica dei pesi richiede la definizione di una funzione costo o funzione errore; indicando con y^p l'output della rete corrispondente all'input $\mathbf{x}^p$ in generale si avrà $y^p \neq d^p$, per cui, rappresentando con E^p l'errore quadratico relativo al p-esimo vettore del training set, la somma degli errori al quadrato su tutti i *pattern* p di training risulterà:

$$E = \sum_p E^p := \tfrac{1}{2} \sum_p (d^p - y^p)^2$$

La procedura LMS permette di trovare tutti i pesi che minimizzano la funzione errore con il metodo chiamato del *gradiente discendente*, che si basa sull'idea di cambiare tutti i pesi in modo proporzionale alla derivata cambiata di segno dell'errore su ogni singolo pattern, calcolata rispetto a ciascun peso:

$$\Delta_p w_i := -\gamma \frac{\partial E^p}{\partial w_i}$$

avendo indicato con γ una costante di proporzionalità. La derivata suddetta può essere scritta come:

$$\frac{\partial E^p}{\partial w_i} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial w_i}$$

dato che le unità sono lineari si ha:

$$\frac{\partial E^p}{\partial y^p} = -(d^p - y^p)$$

e

$$\frac{\partial y^p}{\partial w_i} = x_i^p$$

essendo x_i^p la i-esima componente del vettore di input $\mathbf{x}^p$. Definendo con $\delta^p := d^p - y^p$ la differenza tra il valore di output desiderato e quello reale la regola di aggiornamento dei pesi può essere scritta come:

$$\forall i \quad \Delta_p w_i = \gamma \, (d^p - y^p) \, x_i^p = \gamma \, \delta^p \, x_i^p$$

La *delta rule* permette di modificare nel modo corretto i pesi dei collegamenti per output di entrambe le polarità, sia continui che binari o bipolari, grazie a queste caratteristiche trova applicazione in molti settori diversi.

3.8.1 La regola di aggiornamento del perceptron

Una volta ottenuta l'espressione della *delta rule* è possibile mostrare che la regola di aggiornamento dei pesi del perceptron non è altro che una specializzazione di tale regola.

Dimostrazione. Il perceptron può avere solo due output: $\{-1, +1\}$ e la modifica dei pesi avviene solo se l'output risulta errato.

Definendo come sopra $\delta := d(\mathbf{x}) - y(\mathbf{x})$ la differenza tra il valore di output desiderato e quello reale, si ha che, in presenza di un errore di classificazione, in corrispondenza del valore desiderato $d(\mathbf{x}) = +1$ si ha sempre $\delta = +2 = 2\, d(\mathbf{x})$ mentre nel caso $d(\mathbf{x}) = -1$ si può avere soltanto $\delta = -2 = 2\, d(\mathbf{x})$. L'espressione della *delta rule*:

$$\forall i \quad \Delta w_i = \gamma\, (d(\mathbf{x}) - y(\mathbf{x}))\, x_i = \gamma\, \delta\, x_i$$

può pertanto essere riscritta sostituendo $2\, d(\mathbf{x})$ al posto di δ

$$\forall i \quad \Delta w_i = \gamma\, 2\, d(\mathbf{x})\, x_i$$

a questo punto ponendo $\gamma := \frac{1}{2}$ si trova proprio la regola di aggiornamento del perceptron:

$$\forall i \quad \Delta w_i = d(\mathbf{x})\, x_i$$

che era stata a suo tempo introdotta come una definizione. □

3.9 La back-propagation

La *back-propagation* può essere considerata una generalizzazione della *delta rule*, applicabile a reti neurali multi-layer con funzioni di attivazione non lineari.

Tali tipi di reti sono estremamente importanti poiché soddisfano il: *Teorema di approssimazione universale* secondo cui per approssimare con precisione arbitraria una funzione che presenta un numero finito di discontinuità è sufficiente un solo hidden layer, purché le funzioni di attivazione siano non lineari.

La dimostrazione di tale teorema si può trovare con varie sfumature in: Hornik, Stinchcombe e White, 1989, [17]; Funahashi, 1989, [10]; Cybenko, 1989, [7]; Hartman, Keeler e Kowalski, 1990, [15].

Nel teorema precedente la richiesta di funzioni di attivazione non lineari è imprescindibile, dato che una rete multi-layer se fosse formata da nodi con funzioni di risposta lineari avrebbe le stesse capacità di elaborazione di una rete neurale a singolo layer.

L'idea centrale di tale sistema si basa sulla propagazione degli errori che si riscontrano nel layer di output, sui neuroni dell'ultimo strato hidden e così via risalendo agli strati precedenti; è tale meccanismo di propagazione all'indietro degli errori che dà il nome all'algoritmo. Una delle pubblicazioni più note sulla back-propagation è quella di Rumelhart, Hinton e Williams del 1986 [36].

L'algoritmo si applica generalmente alle reti feed-forward, configurate normalmente in modo che ogni neurone sia collegato con tutti quelli appartenenti allo strato successivo.

3.10 La delta rule generalizzata

La schema di funzionamento dell'algoritmo della *delta rule generalizzata* è il seguente:

1. Per prima cosa si fa operare normalmente la rete neurale in feed-forward, quindi si confrontano i valori ottenuti dai nodi di output con quelli desiderati e nel caso non coincidano si passa al punto successivo

2. Si calcola l'errore da attribuire ad ogni nodo della rete, partendo dai nodi dell'ultimo layer di output e risalendo progressivamente ai layer precedenti con il procedimento che sarà poi illustrato. É questa la fase chiamata di back-propagation degli errori.

3. A questo punto si calcolano tutte le variazioni da apportare ai pesi delle connessioni. É molto importante che le correzioni ai pesi sia-

no effettuate solo dopo aver determinato gli errori back-propagati per ogni unità della rete, altrimenti tali correzioni si intrecciererebbero con la back-propagation degli errori e le correzioni successive non corrisponderebbero più alla direzione negativa del gradiente sulla superficie degli errori. Purtroppo questa è una vera e propria trappola in cui a volte qualche autore cade [1].

Nel seguito della sezione si indicheranno con:

- s_k^p l'input totale del nodo k quando all'ingresso della rete è presente il p-esimo vettore del training set,
- w_{jk} il peso del collegamento che va dal neurone j a quello k,
- y_j^p l'output del nodo j, quando nel layer di input c'è il *pattern* p,
- θ_k il termine di bias relativo all'unità k,

l'input totale che interessa in nodo k è:

$$s_k^p := \sum_j w_{jk}\, y_j^p - \theta_k$$

Indicando con $\mathcal{F}$ la funzione di attivazione differenziabile che caratterizza ogni neurone, l'output del nodo k risulta:

$$y_k^p := \mathcal{F}(s_k^p)$$

In una rete feed-forward in generale nel layer di output ci possono essere più nodi, per cui si indicherà con n_o il loro numero.

Quando all'input si ha il p-esimo pattern, nell'o-esimo nodo del layer di output si avranno: un valore desiderato d_o^p e un valore effettivamente ottenuto y_o^p; l'errore quadratico associato al *pattern* p si ottiene sommando tutti i contributi relativi alle unità di output:

$$E^p := \tfrac{1}{2} \sum_{o=1}^{n_o} (d_o^p - y_o^p)^2$$

a questo punto si può scrivere:

$$\frac{\partial E^p}{\partial w_{jk}} = \frac{\partial E^p}{\partial s_k^p} \frac{\partial s_k^p}{\partial w_{jk}}$$

applicando la definizione di input totale s_k^p, il secondo termine diventa:

$$\frac{\partial s_k^p}{\partial w_{jk}} = y_j^p$$

si definisce poi il primo fattore dell'espressione precedente come:

$$\delta_k^p := -\frac{\partial E^p}{\partial s_k^p}$$

tale espressione corrisponde ad un gradiente discendente sulla superficie degli errori. Effettuando tali sostituzioni si trova una regola di aggiornamento che è uguale alla *delta rule* descritta nella sezione 3.8:

$$\Delta_p w_{jk} := -\gamma \frac{\partial E^p}{\partial w_{jk}} = \gamma \, \delta_k^p \, y_j^p$$

ora occorre stabilire che valore devono avere gli errori δ_k^p per ogni unità k della rete. Procedendo come nel caso della *delta rule* si ha:

$$\delta_k^p := -\frac{\partial E^p}{\partial s_k^p} = -\frac{\partial E^p}{\partial y_k^p} \frac{\partial y_k^p}{\partial s_k^p}$$

il secondo fattore, ricorrendo alla definizione data di y_k^p, risulta:

$$\frac{\partial y_k^p}{\partial s_k^p} = \mathcal{F}'(s_k^p)$$

essendo $\mathcal{F}'(s_k^p)$ la derivata della funzione di attivazione della k-esima unità, calcolata nel punto coincidente con l'input totale di tale nodo: s_k^p.

Per il calcolo degli errori δ_k^p occorre procedere nel modo seguente:

1. Per ogni neurone o nel layer di output, utilizzando la definizione di E^p, si trova lo stesso risultato ottenuto nel caso della *delta rule*:

   $$\frac{\partial E^p}{\partial y_o^p} = -(d_o^p - y_o^p)$$

   ne consegue che per ogni unità di output si ha:

   $$\forall o \quad \delta_o^p = (d_o^p - y_o^p) \, \mathcal{F}_o'(s_o^p)$$

2. Si considerano poi i neuroni h nell'ultimo hidden layer, formato da n_h unità; dato che non si conosce direttamente il contributo di ciascuno di essi all'errore sull'output, si utilizza la procedura seguente:

$$\begin{aligned}\frac{\partial E^p}{\partial y_h^p} &= \sum_{o=1}^{n_o} \frac{\partial E^p}{\partial s_o^p}\,\frac{\partial s_o^p}{\partial y_h^p} = \sum_{o=1}^{n_o} \frac{\partial E^p}{\partial s_o^p}\,\frac{\partial}{\partial y_h^p} \sum_{j=1}^{n_h} \left(w_{jo}\, y_j^p - \theta_o\right) = \\ &= \sum_{o=1}^{n_o} \frac{\partial E^p}{\partial s_o^p}\, w_{ho} = -\sum_{o=1}^{n_o} \delta_o^p\, w_{ho}\end{aligned}$$

per ogni unità dell'ultimo stato nascosto si ha quindi:

$$\forall h \quad \delta_h^p = \mathcal{F}_h'(s_h^p) \sum_{o=1}^{n_o} \delta_o^p\, w_{ho}$$

3. Avendo determinato le correzioni a tutti i nodi dell'ultimo strato nascosto si può ripetere la procedura del punto 2) considerando tale strato come se fosse quello di output, e così via fino a che ad ogni neurone risulti associato il rispettivo errore back-propagato.

Conclusa tale fase ad ogni collegamento della rete si applica la procedura già descritta per l'aggiornamento dei pesi:

$$\Delta_p w_{jk} = \gamma\, \delta_k^p\, y_j^p$$

3.10.1 Casi tipici di delta rule generalizzata

I risultati della sezione precedente possono essere riassunti in tre formule:

1. L'errore associato ai neuroni dello stato di output è:

$$\forall o \quad \delta_o^p = (d_o^p - y_o^p)\, \mathcal{F}_o'(s_o^p)$$

2. L'errore associato ai neuroni dell'ultimo hidden layer è:

$$\forall h \quad \delta_h^p = \mathcal{F}_h'(s_h^p) \sum_{o=1}^{n_o} \delta_o^p\, w_{ho}$$

tale formula vale anche per tutti gli altri hidden layer, reinterpretando l'indice o come un indice sugli elementi dell'ultimo hidden layer a cui sono stati assegnati gli errori, e l'indice h come quello che identifica i neuroni del layer immediatamente precedente, a cui devono essere ancora back-propagati gli errori.

3. L'aggiornamento dei pesi si effettua applicando la seguente regola:

$$\Delta_p w_{jk} = \gamma\, \delta_k^p\, y_j^p$$

Le funzioni di attivazione non lineari $\mathcal{F}$ più utilizzate sono la sigmoide e la tangente iperbolica; dato che le loro derivate si possono esprimere in funzione dell'output dei neuroni, sostituendo tali espressioni di $\mathcal{F}'$ nelle equazioni suddette, si possono determinare tutti i pesi di una qualunque rete feed-forward che utilizzi le funzioni di attivazione di cui sopra. L'unico parametro non ancora assegnato rimarrebbe infatti la costante γ detta *learning rate*.

· Se la funzione di attivazione è la sigmoide si ha:

$$y_k^p := \mathcal{F}(s_k^p) := \frac{1}{1 - \mathrm{e}^{-s_k^p}}$$

e la sua derivata risulta:

$$\begin{aligned} \mathcal{F}'(s_k^p) &= \frac{\partial}{\partial s_k^p} \frac{1}{1 - \mathrm{e}^{-s_k^p}} = \\ &= \frac{1}{\left(1 - \mathrm{e}^{-s_k^p}\right)^2} \left(-\mathrm{e}^{-s_k^p}\right) = \\ &= \frac{1}{1 - \mathrm{e}^{-s_k^p}} \frac{-\mathrm{e}^{-s_k^p}}{1 - \mathrm{e}^{-s_k^p}} = \\ &= y_k^p \left(1 - y_k^p\right) \end{aligned}$$

· Se la funzione di attivazione invece è la tangente iperbolica si ha:

$$y_k^p := \mathcal{F}(s_k^p) := \mathrm{Tanh}(s_k^p) = \frac{\mathrm{Sinh}(s_k^p)}{\mathrm{Cosh}(s_k^p)}$$

e la sua derivata risulta:

$$\begin{aligned}
\mathcal{F}'(s_k^p) &= \frac{\partial}{\partial s_k^p}\frac{\mathrm{Sinh}(s_k^p)}{\mathrm{Cosh}(s_k^p)} = \\
&= \frac{\mathrm{Cosh}^2(s_k^p) - \mathrm{Sinh}^2(s_k^p)}{\mathrm{Cosh}^2(s_k^p)} = 1 - \mathrm{Tanh}^2(s_k^p) \\
&= (1 + \mathrm{Tanh}(s_k^p))\,(1 - \mathrm{Tanh}(s_k^p)) \\
&= (1 + y_k^p)\,(1 - y_k^p)
\end{aligned}$$

3.10.2 Il learning rate e il momento

La costante γ detta *learning rate* ha la funzione di incrementare il gradiente di discesa, e può accelerare la convergenza, se però il suo valore fosse troppo grande si avrebbero delle oscillazioni e il minimo potrebbe non essere mai raggiunto; per contrastare tale fenomeno a volte si applica una modifica alla usuale regola di aggiornamento dei pesi, introducendo un termine che tiene conto del precedente cambiamento apportato agli stessi:

$$\Delta w_{jk}(t+1) = \gamma\,\delta_k^p\,y_j^p + \alpha\,\Delta w_{jk}(t)$$

l'indice t rappresenta il numero degli aggiornamenti effettuati mentre α è una costante che determina l'importanza relativa del cambiamento precedente. Il termine aggiuntivo chiamato *momento* risulta quindi avere l'effetto di una 'inerzia' che tende a smorzare le oscillazioni e potrebbe fare uscire da eventuali condizioni di minimo locale che altrimenti intrappolerebbero la soluzione.

Per evitare minimi locali è opportuno inoltre che gli elementi del training set siano ripresentati, quando occorre, con un ordine permutato rispetto a quello iniziale, in modo che il sistema non si focalizzi in particolare sui primi esempi.

3.11 I problemi della back-propagation

Purtroppo nonostante l'apparente successo della back-propagation ci sono degli aspetti di essa che fanno si che la sua utilità non sia universale,

uno dei problemi principali è ad esempio il lungo processo di training necessario, dovuto principalmente ad una scelta non ottimale del learning rate e del momento.

Un altro problema serio quando si hanno reti complesse è rappresentato dalla presenza di *minimi locali* sulla superficie degli errori, perché l'algoritmo del gradiente discendente potrebbe finirci dentro senza riuscire ad uscirne nonostante il termine di momento. In realtà l'utilizzo di metodi probabilistici potrebbe essere d'aiuto ma essi tendono ad essere molto lenti. A volte si usa il trucco di aumentare il numero delle unità nascoste in modo da avere uno spazio di dimensione maggiore, in cui risulti più difficile rimanere bloccati, ma anche tale sistema ha dei limiti per cui oltre una certa soglia non si hanno più vantaggi e si finisce ancora intrappolati come prima.

In alcuni casi poi si arriva a quella che è chiamata *paralisi della rete* in cui praticamente non si hanno più variazioni dei pesi delle connessioni e si ha una situazione di stallo. La causa di tale fenomeno è da ricercarsi prevalentemente nella presenza di valori troppo alti dei pesi che, moltiplicando eccessivamente gli output, causano nei neuroni a cui sono collegati degli input molto elevati. Dato che la *delta rule generalizzata* fa uso della derivata della funzione di attivazione nel punto corrispondente all'input totale, se questo è troppo grande, indipendentemente dal suo segno, funzioni di attivazione come la sigmoide o la tangente iperbolica tenderanno ad avere una derivata pressoché nulla che bloccherà ulteriori modifiche dei pesi in questione.

3.12 La verifica dell'apprendimento

Quando si effettua il training di una rete neurale generalmente di procede utilizzando un insieme di esempi che è chiamato *training set*, formato da coppie di dati di input ed output. Quando la rete ha imparato a trattare tale insieme, per verificare se essa è in grado di generalizzare,

si ricorre ad un altro insieme di esempi, generalmente di dimensione inferiore, chiamato *test set*.

Per poter effettuare valutazioni quantitative occorrono le seguenti definizioni di errore:

L'errore di output E^p quando in input è presente il vettore p è:

$$E^p := \frac{1}{2}\sum_{o=1}^{n_o}(d_o^p - y_o^p)^2$$

avendo indicato con n_o il numero delle unità del layer di uscita, con d_o^p i valori di output desiderati dato l'input p e con y_o^p gli output effettivi.

L'errore medio su un training set di n_p elementi, detto anche *learning error rate* è:

$$E_{learning} := \frac{1}{n_p}\sum_{p=1}^{n_p} E^p$$

Il *test error* E_{test}, se l'insieme di test è formato da n_t elementi, risulta:

$$E_{test} := \frac{1}{n_t}\sum_{p=1}^{n_t} E^p$$

Se si utilizza un training set formato da pochi elementi in generale si avrà un $E_{learning}$ piccolo ed un E_{test} molto più grande, aumentando gradualmente le dimensioni dell'insieme di addestramento, il valore dei due errori, se non si finisce intrappolati in qualche minimo locale, tende a convergere verso un limite comune legato al cosiddetto *representational power* della rete, che dipende in gran parte dal numero di unità nascoste utilizzate e dalla loro disposizione in vari layer.

Le prestazioni di una rete neurale in funzione del numero dei nodi nascosti ha in generale il seguente andamento: inizialmente si ha una diminuzione sia dell'$E_{learning}$ che dell'E_{test}, poi oltre un certo limite, dipendente dalla funzione soggiacente ai dati del campione, l'E_{test} diventa crescente, mentre l'$E_{learning}$ continua a decrescere costantemente, in modo non sufficiente però da controbilanciare l'aumento dell'altro termine. Alla fine si ottiene una curva che presenta un minimo in corrispondenza di un certo numero di hidden units. Le cause di tale andamento sono da

ricercare nel fatto che in generale tutte le misure sono affette da rumore, aumentando il numero di unità nascoste si riescono ad approssimare sempre meglio i punti del training set, ma oltre un certo limite si finisce per interpolare anche il rumore per cui si perde la proprietà di generalizzazione e gli errori sull'insieme di test crescono progressivamente. Quando si stanno utilizzando troppe unità nascoste si dice che si è in *overtraining*.

Capitolo 4

Verso l'unificazione dei problemi inversi

In questo capitolo si affronterà l'argomento dei problemi inversi, con particolare riferimento alla modellizzazione di dati empirici, ricorrendo agli strumenti introdotti nei capitoli precedenti.

Affinché si possa sperare di riuscire ad estrarre un modello dai dati a disposizione la condizione più generale e al contempo più debole che deve essere soddisfatta dalla funzione cercata è che essa debba presentare un minimo grado di smoothness, in modo che l'informazione contenuta in un punto si possa considerare come non totalmente locale, risultando in parte collegata al suo intorno.

Il risultato di quello che può essere considerato come un apprendimento tramite degli esempi, costituiti dai dati campione, dipenderà in gran parte dalla qualità e dalla quantità dei dati a disposizione, ma anche l'approccio utilizzato è spesso determinante.

A causa della loro origine, i dati sono finiti e campionati, e il campionamento spesso non è uniforme, per cui essi tendono a formare delle distribuzioni *sparse* nello spazio di input, che corrispondono quasi sempre a problemi inversi mal posti, soprattutto quando i dati sono affetti da rumore, per cui sono necessarie delle procedure di regolarizzazione.

Se si volesse ricostruire, a partire dalle coppie di dati di input e di

output di un training set, la funzione senza rumore sottostante, e si volessero utilizzare delle reti neurali, ad esempio del tipo feed-forward come quelle illustrate nel capitolo precedente, ci si troverebbe di fronte a molte scelte importanti senza avere molti aiuti da parte della teoria.

La prima riguarda il numero dei layer che si intendono impiegare, che per funzioni non lineari devono essere almeno due, non contando quello di input, poi occorre scegliere il numero di neuroni nascosti per ogni layer e le rispettive funzioni di output, dette di attivazione. Maggiore è il numero delle unità nascoste utilizzate e più grande è la complessità delle funzioni che si possono apprendere, dato che diminuisce lo smoothing, il pericolo dell'overtraining però è sempre presente per cui si potrebbero avere delle funzioni che tendono a interpolare e non generalizzano bene per nuovi input.

Anche ammesso di poter trovare la configurazione migliore, se la rete neurale deve regolare i pesi delle connessioni tra più layers occorre l'aggiornamento dei pesi dato dalla *delta rule generalizzata*, e sulla superficie associata alla funzione degli errori di output si possono avere dei minimi locali, che cambiano in base ai dati utilizzati per il training, e da cui a volte non si riesce ad uscire nonostante la presenza di un termine aggiuntivo di momento, per cui può accadere che non ci sia convergenza verso la soluzione migliore.

Oltre a tutte le difficoltà elencate sopra rimane poi sempre il problema della regolarizzazione, con particolare riferimento alla stabilità delle soluzioni, che richiede attenzioni particolari.

Per poter risolvere tali problemi si può far ricorso alla teoria dell'apprendimento statistico supervisionato, di cui si richiameranno i punti essenziali.

4.1 Richiami all'apprendimento statistico

La funzione obiettivo dell'apprendimento è sviluppata utilizzando delle funzioni di kernel note: $K_{x_i}(x)$ centrate su ogni punto di input $\{x_i\}_{i=1}^n$;

indicando con $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ un vettore di coefficienti reali da stimare si ha che:

$$f(x) = \sum_{i=1}^{n} c_i \, K_{x_i}(x)$$

con questo approccio tutte le informazioni da apprendere sono contenute quindi nel vettore dei coefficienti $\mathbf{c}$.

Indicando con $\mathbf{y}$ il vettore dei valori di output e con γ una costante positiva, il problema dell'apprendimento può essere ridotto, tenendo conto anche della regolarizzazione, al seguente sistema di equazioni in n variabili c_i espresse in forma matriciale:

$$\mathbf{y} = (\mathbf{K} + n\,\gamma\,\mathbb{1})\,\mathbf{c}$$

il termine $n\,\gamma\,\mathbb{1}$ è quello responsabile della regolarizzazione; esso rende la matrice tra parentesi positiva e non semplicemente semi-definita positiva come usualmente è $\mathbf{K}$, in tal modo è garantita la convessità, e la soluzione può essere ricavata in molti modi, per esempio con il calcolo della inversa di $(\mathbf{K} + n\,\gamma\,\mathbb{1})$, essendo invertibile:

$$\mathbf{c} = (\mathbf{K} + n\,\gamma\,\mathbb{1})^{-1}\mathbf{y}$$

Se non ci fosse il termine di regolarizzazione $n\,\gamma\,\mathbb{1}$ si potrebbe utilizzare, al posto della matrice inversa che moltiplica il vettore degli output $\mathbf{y}$, la pseudo-inversa di Moore Penrose della matrice $\mathbf{K}$, indicata con $\mathbf{K}^{+}$; la funzione $f(x)$ che si ricaverebbe con i coefficienti $\mathbf{c}$ trovati risolvendo l'equazione: $\mathbf{c} = \mathbf{K}^{+}\mathbf{y}$ sarebbe quella di norma minima che, con le funzioni di kernel assegnate, interpolerebbe nel modo migliore possibile i dati a disposizione, senza però discriminare l'eventuale rumore compreso in essi. Per ottenere una interpolazione migliore non ci sarebbe altra via che cambiare le funzioni di kernel $K_{x_i}(x)$.

4.2 L'utilizzo della pseudo-inversa di Moore-Penrose

Teorema. *Siano* $\mathbf{X}$ *e* $\mathbf{Y}$ *due matrici generiche rispettivamente* $k \times m$ *e* $n \times m$*. Indicando con* $\mathbf{X}^+$ *la pseudo-inversa di* $\mathbf{X}$*, la matrice* $n \times k$ *:* $\mathbf{W} = \mathbf{Y}\,\mathbf{X}^+$ *minimizza la norma quadratica della matrice* $\mathbf{Y} - \mathbf{W}\,\mathbf{X}$*.*

Dimostrazione. Sia $E := \|\mathbf{Y} - \mathbf{W}\,\mathbf{X}\|^2$, e $\mathbf{S} := (\mathbf{Y} - \mathbf{W}\,\mathbf{X})\,(\mathbf{Y} - \mathbf{W}\,\mathbf{X})^\dagger$, il valore di E può essere determinato calcolando la traccia della matrice $\mathbf{S}$, cioè sommando gli elementi posti sulla diagonale principale di $\mathbf{S}$:

$$E = \mathrm{Tr}(\mathbf{S})$$

la matrice $\mathbf{S}$ può essere riscritta come segue:

$$\widetilde{\mathbf{S}} := \big(\mathbf{Y}\,\mathbf{X}^+ - \mathbf{W}\big)\mathbf{X}\,\mathbf{X}^\dagger\big(\mathbf{Y}\,\mathbf{X}^+ - \mathbf{W}\big)^\dagger + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^+\mathbf{X}\big)\,\mathbf{Y}^\dagger$$

L'uguaglianza di tale espressione con quella usata in precedenza sarà provata mediante un calcolo diretto, in modo da riportare $\widetilde{\mathbf{S}}$ alla forma originale di $\mathbf{S}$; se si moltiplica a sinistra il termine $\mathbf{X}\,\mathbf{X}^\dagger$ si ricava:

$$\widetilde{\mathbf{S}} = \big(\mathbf{Y}\,\mathbf{X}^+\mathbf{X}\,\mathbf{X}^\dagger - \mathbf{W}\,\mathbf{X}\,\mathbf{X}^\dagger\big)\big(\mathbf{Y}\,\mathbf{X}^+ - \mathbf{W}\big)^\dagger + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^+\mathbf{X}\big)\,\mathbf{Y}^\dagger$$

per le proprietà della Moore-Penrose esposte nella sezione 2.5, si ha che la matrice $\mathbf{X}^+\mathbf{X}$ è hermitiana: $\mathbf{X}^+\mathbf{X} = (\mathbf{X}^+\mathbf{X})^\dagger$, per cui: $\mathbf{X}^+\mathbf{X}\,\mathbf{X}^\dagger = (\mathbf{X}^+\mathbf{X})^\dagger\,\mathbf{X}^\dagger = (\mathbf{X}\,(\mathbf{X}^+\mathbf{X}))^\dagger = \mathbf{X}^\dagger$; con questa sostituzione l'espressione

precedente diventa:

$$\begin{aligned}
\widetilde{\mathbf{S}} &= \big(\mathbf{Y}\,\mathbf{X}^{\dagger} - \mathbf{W}\,\mathbf{X}\,\mathbf{X}^{\dagger}\big)\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)^{\dagger} + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\,\mathbf{X}^{\dagger}\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)^{\dagger} + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\big(\mathbf{Y}\,\mathbf{X}^{+}\mathbf{X} - \mathbf{W}\,\mathbf{X}\big)^{\dagger} + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\Big(\big(-\mathbf{W}\,\mathbf{X}\big)^{\dagger} + \big(\mathbf{Y}\,\mathbf{X}^{+}\mathbf{X}\big)^{\dagger}\Big) + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\Big(\big(-\mathbf{W}\,\mathbf{X}\big)^{\dagger} + \mathbf{X}^{+}\mathbf{X}\,\mathbf{Y}^{\dagger}\Big) + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\big(-\mathbf{W}\,\mathbf{X}\big)^{\dagger} - \mathbf{W}\,\mathbf{X}\,\mathbf{X}^{+}\mathbf{X}\,\mathbf{Y}^{\dagger} + \mathbf{Y}\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\big(-\mathbf{W}\,\mathbf{X}\big)^{\dagger} - \mathbf{W}\,\mathbf{X}\,\mathbf{Y}^{\dagger} + \mathbf{Y}\,\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\big(-\mathbf{W}\,\mathbf{X}\big)^{\dagger} + \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\mathbf{Y}^{\dagger} \\
&= \big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)\big(\mathbf{Y} - \mathbf{W}\,\mathbf{X}\big)^{\dagger} = \mathbf{S}
\end{aligned}$$

avendo ritrovato l'espressione iniziale della $\mathbf{S}$ ne consegue che esse sono equivalenti, per cui si può applicare l'operazione di traccia ad $\widetilde{\mathbf{S}}$:

$$\begin{aligned}
E &= \mathrm{Tr}\Big[\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)\mathbf{X}\,\mathbf{X}^{\dagger}\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)^{\dagger} + \mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger}\Big] = \\
&= \mathrm{Tr}\Big[\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)\mathbf{X}\,\mathbf{X}^{\dagger}\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)^{\dagger}\Big] + \mathrm{Tr}\Big[\mathbf{Y}\,\big(\mathbb{1} - \mathbf{X}^{+}\mathbf{X}\big)\,\mathbf{Y}^{\dagger}\Big]
\end{aligned}$$

La seconda traccia rappresenta un termine costante che può essere trascurato. Dato che gli elementi sulla diagonale principale della matrice: $\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)\mathbf{X}\,\mathbf{X}^{\dagger}\big(\mathbf{Y}\,\mathbf{X}^{+} - \mathbf{W}\big)^{\dagger}$ possono essere solo positivi o nulli, essendo tale matrice della forma $\mathbf{A}\,\mathbf{A}^{\dagger}$, la prima traccia avrà come valore minimo zero, ottenibile ponendo: $\mathbf{W} = \mathbf{Y}\,\mathbf{X}^{+}$, per cui il teorema risulta dimostrato. □

La più generale soluzione del precedente problema dei minimi quadrati può essere scritta nella forma seguente:

$$\mathbf{W} = \mathbf{Y}\,\mathbf{X}^{+} + \mathbf{M}\,\big(\mathbb{1} - \mathbf{X}\,\mathbf{X}^{+}\big)$$

avendo indicato con $\mathbf{M}$ una generica matrice $n \times k$, infatti grazie alle

proprietà della pseudo-inversa si ha:

$$\begin{aligned}\mathbf{W}\,\mathbf{X} &= \mathbf{Y}\,\mathbf{X}^{+}\mathbf{X} + \mathbf{M}\left(\mathbb{1} - \mathbf{X}\,\mathbf{X}^{+}\right)\mathbf{X} \\ &= \mathbf{Y}\,\mathbf{X}^{+}\mathbf{X} + \mathbf{M}\left(\mathbf{X} - \mathbf{X}\,\mathbf{X}^{+}\mathbf{X}\right) \\ &= \mathbf{Y}\,\mathbf{X}^{+}\mathbf{X} + \mathbf{M}\left(\mathbf{X} - \mathbf{X}\right) \\ &= \mathbf{Y}\,\mathbf{X}^{+}\mathbf{X}\end{aligned}$$

tra tutte le soluzioni del problema dei minimi quadrati la pseudo-inversa di Moore-Penrose è quindi quella che ha la norma minore, è per questo che spesso si dice che è la migliore alternativa in mancanza della matrice inversa.

Naturalmente il teorema appena dimostrato, valendo per matrici generiche, può essere applicato alle matrici e ai vettori reali che rappresentano il caso tipico del learning.

4.3 Reti neurali e apprendimento statistico

Utilizzando i risultati della teoria dell'apprendimento statistico è possibile realizzare delle reti neurali con un solo hidden layer in grado di apprendere dagli esempi forniti; il particolare più rilevante che le contraddistingue dalle altre è che in esse sono modificabili solo le connessioni tra il layer nascosto e quello di output, ma nonostante ciò possono apprendere qualunque tipo di funzione con un numero finito di discontinuità, e non solo quelle che presentano una dipendenza lineare dagli input. I pesi delle connessioni tra il layer di ingresso e quello nascosto sono infatti fissati una volta per tutte nella fase iniziale e non necessitano di training, questo fa si che per tale rete neurale la regola di aggiornamento dei pesi detta *delta rule generalizzata*, si riduca semplicemente alla *delta rule*, che consente di trovare sempre una soluzione ottimale non presentando problemi di minimi locali.

Esse di basano sullo sviluppo della funzione cercata tramite delle

funzioni di kernel relative agli n punti del training set:

$$f(x) = \sum_{i=1}^{n} c_i \, K_{x_i}(x)$$

e i pesi delle connessioni da determinare corrispondono ai coefficienti c_i.

In pratica le funzioni di kernel rimappano i dati di input in uno spazio di dimensione maggiore rispetto a quello di partenza in modo che poi si possa cercare l'iperpiano interpolante che meglio approssimi i dati a disposizione.

Supponendo di cercare una funzione che abbia k variabili indipendenti di input e restituisca m valori di output, avendo a disposizione un training set formato da n esempi, è possibile costruire la rete neurale nel modo seguente:

- Nel primo layer di input si collocano tanti nodi quante sono le variabili indipendenti del problema.

- Nel secondo layer detto hidden layer, perché non in diretto contatto con l'esterno della rete, si posizionano tanti neuroni quanti sono gli esempi a disposizione, ognuno programmato con la stessa funzione di kernel ma parametrizzati con dei *bias* diversi, in modo tale che ogni funzione di output risulti 'centrata' rispetto alle coordinate di input di un esempio diverso; quindi si connette ciascuna di queste unità con tutti i nodi di input, utilizzando dei collegamenti fissi, che non prevedono cioè la possibilità di regolazione di pesi ad essi associati.

  Per chiarire questo punto si può analizzare il seguente esempio: si supponga di avere un training set formato da due coppie di dati: $(\mathbf{x}_1, \mathbf{y}_1)$ e $(\mathbf{x}_2, \mathbf{y}_2)$, avendo indicato con $\mathbf{x}$ il vettore dei dati di input e con $\mathbf{y}$ quello dei dati di output. L'hidden layer sarà formato da due nodi aventi funzioni di attivazione rispettivamente del tipo $K(\mathbf{x}-\mathbf{x}_1)$ e $K(\mathbf{x}-\mathbf{x}_2)$. Molte funzioni di Kernel calcolano l'output in funzione della norma del loro argomento, nel caso suddetto per:

$\|\mathbf{x}-\mathbf{x}_1\|$ e $\|\mathbf{x}-\mathbf{x}_2\|$, e sono chiamate *Radial Basis Functions*(RBF) a causa della loro simmetria radiale rispetto ad ogni punto di training.

· Nel layer di output si collocano tanti neuroni quanti sono i valori di output richiesti, quindi si collega ogni nodo con tutti quelli dell'hidden layer, realizzando quella che è chiamata una rete *fully connected*. Tali unità sono quasi sempre dei nodi sommatori, che restituiscono come output la somma dei valori di input, ma si potrebbero scegliere anche delle funzioni non lineari o a soglia, se si fosse interessati ad eseguire delle classificazioni invece che delle regressioni.

Purtroppo il calcolo dei coefficienti $\mathbf{c}$ dell'espansione, rappresentati dai pesi dei collegamenti ai neuroni di output, può diventare un operazione che richiede molto tempo per poter essere eseguita, dato che la complessità di tale operazione è stata stimata dell'ordine di n^3, essendo n il numero degli esempi che costituiscono il training set, paragonabile quindi alla classica inversione di una matrice $n \times n$. Inoltre se i dati sono troppo rumorosi il risultato che si otterrebbe potrebbe essere privo di significato in quanto non regolarizzato e mal condizionato.

Per poter risolvere questi problemi è possibile effettuare a parte il calcolo dei coefficienti $\mathbf{c}$ utilizzando la equazione che contiene il termine di regolarizzazione:

$$\mathbf{c} = (\mathbf{K} + n\,\gamma\,\mathbb{1})^{-1}\mathbf{y}$$

fissando il parametro γ in modo che si abbia il giusto compromesso tra interpolazione e smoothing, quindi si può configurare la rete neurale con i pesi corrispondenti ai c_i trovati.

In tal modo è possibile sfruttare tutta la potenza e la velocità intrinseche al calcolo parallelo delle reti neurali, oltre alla loro capacità di essere parzialmente *fault tolerant*, per poter eseguire compiti complessi, impostati tramite la funzione risposta così ottenuta.

4.4 L'inversione con l'algoritmo dei frame

Il vettore dei coefficienti $\mathbf{c}$ potrebbe essere determinato utilizzando il metodo iterativo di Richardson che è stato illustrato nella sezione 2.4 a proposito dei frame; per calcolare il *rate* di convergenza ottenibile con tale metodo è necessario richiamare alcuni dei risultati che sono stati ricavati in precedenza.

Nella sezione 1.19 è stato dimostrato che il numero di condizionamento della matrice: $(\mathbf{K} + n\,\gamma\,\mathbb{1})$ è:

$$\kappa\big(\mathbf{K} + n\,\gamma\,\mathbb{1}\big) = 1 + \frac{\kappa(\mathbf{K}) - 1}{1 + n\,\gamma/\lambda_{min}}$$

Mentre nella sezione 2.4 è stato calcolato che il *rate* di convergenza migliore dell'algoritmo iterativo applicato ad una matrice di frame S vale:

$$\|R\| = 1 - \frac{2}{1 + \kappa(S)} \quad \text{con: } \kappa(S) \geq 1$$

Dato che la matrice $(\mathbf{K} + n\,\gamma\,\mathbb{1})$ è definita positiva come la matrice di frame S si possono riunire i due risultati e si trova che se si applica l'algoritmo di Richardson per l'inversione si può raggiungere un rate di convergenza per il learning pari a:

$$\|R\| = 1 - \frac{1}{1 + \frac{\kappa(\mathbf{K})-1}{2\,(1+n\,\gamma/\lambda_{min})}}$$

Aumentando opportunamente il parametro γ si può ridurre il valore di $\|R\|$ ed accelerare così la convergenza, in tal modo però si aumenta anche lo smoothing, per cui occorre procedere con attenzione.

4.5 Teoria dei frame e apprendimento statistico

Teorema. *Sia $\mathcal{H}$ uno spazio di Hilbert e $\{\phi_i\}$ un frame di questo spazio a cui è associato il frame duale canonico $\{\phi_i^*\}$, sia inoltre $\mathbb{R}^\Omega$ l'insieme*

di tutte le funzioni definite su un dominio $\Omega \subset \mathbb{R}^d$ *con valori in* $\mathbb{R}$. *Se* $\{\phi_i\}$ *è un insieme finito o infinito di funzioni di* $\mathbb{R}^\Omega$, *tale che:*

$$\forall t \in \Omega \quad \left\| \sum_i \phi_i^* \, \phi_i(t) \right\|_{\mathcal{H}} < \infty$$

allora $\mathcal{H}$ *è un Reproducing Kernel Hilbert Space*

Dimostrazione. Affinché si possa parlare di Reproducing Kernel Hilbert Space, uno spazio di Hilbert deve essere dotato di un funzionale lineare di valutazione limitato.
Dato che i ϕ_i sono elementi sia di $\mathbb{R}^\Omega$ che di $\mathcal{H}$, per le proprietà dei frame duali esposte nella sezione 2.3 si ha:

$$\forall f \in \mathcal{H} \quad f = \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i$$

poiché $\mathbb{R}^\Omega$ ha la struttura di uno spazio vettoriale, $f = \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i$ vale anche in $\mathbb{R}^\Omega$, per cui risulta che $f \in \mathbb{R}^\Omega$ e ciò garantisce che sia soddisfatta la proprietà di linearità.

Definendo una seminorma per lo spazio vettoriale $\mathbb{R}^\Omega$ come segue:

$$\forall t \in \Omega \ \wedge \ \forall f \in \mathcal{H} \quad \|f\|_t := |f(t)|$$

si ha la seguente convergenza puntuale:

$$f = \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i \ \Leftrightarrow \ f(t) = \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i(t)$$

Rimane ora da dimostrare che il funzionale lineare di valutazione definito sopra è limitato, cioè:

$$\forall f \in \mathcal{H} \ \wedge \ \forall t \in \Omega, \quad \exists M_t > 0 \quad | \quad |f(t)| \leq M_t \, \|f\|_{\mathcal{H}}$$

dato che ogni elemento di $\mathcal{H}$ può essere sviluppato utilizzando i frame si ha che:

$$\forall f \in \mathcal{H} \ \wedge \ \forall t \in \Omega \quad |f(t)| = \left| \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i(t) \right| =$$

$$= \left| \left\langle f, \sum_i \phi_i^* \, \phi_i(t) \right\rangle_{\mathcal{H}} \right| \leq \|f\|_{\mathcal{H}} \left\| \sum_i \phi_i^* \, \phi_i(t) \right\|_{\mathcal{H}}$$

definendo $M_t := \|\sum_i \phi_i^* \, \phi_i(t)\|_{\mathcal{H}}$ si può concludere che $\mathcal{H}$ è un Reproducing Kernel Hilbert Space, dato che M_t è limitato per ipotesi. $\square$

Se nel teorema precedente si usasse l'espansione: $f = \sum_i \langle f, \phi_i \rangle_{\mathcal{H}} \phi_i^*$ allora si dovrebbe imporre la seguente condizione:

$$\forall t \in \Omega \quad \left\| \sum_i \phi_i^*(t) \, \phi_i \right\|_{\mathcal{H}} < \infty$$

Per quanto esposto nel primo capitolo risulta che la funzione:

$$K(s,t) = \sum_i \phi_i^*(s) \, \phi_i(t)$$

è una funzione di kernel poiché:

1. $K(s,t) : \Omega \times \Omega \to \mathbb{R}$ è a valori reali.
2. $K(s,t) = \sum_i \phi_i^*(s) \, \phi_i(t) = \sum_i \phi_i^*(t) \, \phi_i(s) = K(t,s)$ è simmetrica
3. $K(s,t)$ è definita positiva, essendo soddisfatta la condizione:

$$\forall a_1, \ldots, a_n \in \mathbb{R} \ \wedge \ \forall t_1, \ldots, t_n \in \Omega \quad \sum_{i,j} a_i \, a_j \, K(t_i, t_j) \geq 0$$

infatti:

$$K(t_i, t_j) = \sum_l \phi_l^*(t_i) \, \phi_l(t_j)$$

inoltre $\phi_l(t_j)$ può essere sviluppata come segue:

$$\phi_l(t_j) = \sum_m \langle \phi_l, \phi_m \rangle_{\mathcal{H}} \phi_m^*(t_j)$$

per cui:

$$\sum_{i,j} a_i \, a_j \, K(t_i, t_j) = \sum_{i,j} a_i \, a_j \sum_{l,m} \phi_l^*(t_i) \, \langle \phi_l, \phi_m \rangle_{\mathcal{H}} \phi_m^*(t_j) =$$

$$= \left\langle \sum_{i,l} a_i \, \phi_l^*(t_i) \phi_l \middle| \sum_{j,m} a_j \, \phi_m^*(t_j) \phi_m \right\rangle_{\mathcal{H}} = \left\| \sum_{i,l} a_i \, \phi_l^*(t_i) \phi_l \right\|_{\mathcal{H}}^2 \geq 0$$

Inoltre $K(s,t)$ soddisfa anche la proprietà di riproduzione dato che:

$$\forall f \in \mathcal{H} \ \wedge \ \forall t \in \Omega \quad f(t) = \sum_i \langle f, \phi_i^* \rangle_{\mathcal{H}} \phi_i(t) =$$

$$= \left\langle f, \sum_i \phi_i^* \, \phi_i(t) \right\rangle_{\mathcal{H}} = \langle f, K(\cdot, t) \rangle_{\mathcal{H}} := \langle f, K_t \rangle_{\mathcal{H}}$$

Con questo procedimento risulta perciò possibile costruire gli operatori di kernel utilizzando soltanto i frame.

4.6 I problemi inversi e i frame

Trovare una funzione che modellizzi dei dati è un problema che appartiene ad una classe più vasta di *problemi inversi* in cui si cerca di inferire dei parametri che caratterizzano il sistema in osservazione a partire dai dati empirici a disposizione. I problemi inversi sono fondamentali in vari ambiti, per esempio nella diagnostica per immagini, ed in tanti altri campi in cui occorre interpretare dei dati che possono essere inaccurati, o in parte inconsistenti. Nelle sezioni precedenti di questo capitolo si è potuto vedere come nel caso dell'apprendimento statistico sia possibile un approccio algebrico sviluppando la funzione obiettivo del learning mediante delle funzioni di kernel. I coefficienti di tale sviluppo si ottengono mediante l'inversione di una matrice che contiene un termine di regolarizzazione per far si che la soluzione del learning sia generalizzante. Dato che gli operatori di kernel possono essere costruiti con i frame sembrerebbe che essi possano rivelarsi come il linguaggio matematico che permetta di unificare i problemi inversi, infatti molti di tali problemi possono essere visti proprio come un apprendimento statistico. Inoltre esprimendo i kernel con i frame si aprono nuove possibilità, infatti normalmente quando un kernel si rivela inadeguato si deve sostituirlo con un altro, utilizzando i frame però potrebbe essere possibile selezionare i kernel sfruttando delle informazioni a priori che si hanno sul sistema, in modo da ottimizzare ulteriormente il processo di apprendimento.

Conclusione

Con questo lavoro di tesi è stato possibile addentrarsi in settori di ricerca piuttosto specialistici dotati ognuno di una propria letteratura tipica e ricondurli all'usuale formalismo quantistico, in modo da renderne più disponibili i risultati.

Si è fatto ricorso anche a dei concetti quali il numero di condizionamento delle matrici, tipici della letteratura sui problemi inversi, per rendere più agevole e sicuro l'approccio algebrico.

Le reti neurali sono state analizzate in funzione di un loro possibile utilizzo per l'analisi di dati sperimentali e si è descritto uno schema che permette di effettuare con esse l'apprendimento statistico.

Si è potuto mostrare poi come il problema dell'apprendimento supervisionato possa essere affrontato utilizzando soltanto i frame, inizialmente se ne è evidenziata l'utilità come ausilio matematico per calcolare l'effetto dell'applicazione di una matrice inversa ad un vettore, poi si è spiegato come poter realizzare con essi gli operatori di kernel che sono alla base dello statistical learning.

Sembrerebbe ragionevole pensare che alcuni dei risultati ottenuti nei campi suddetti possano rivelarsi utili anche nell'ambito delle scienze fisiche, in particolare nell'ambito della teoria della misura, mediante l'utilizzo, ad esempio, di tecniche di learning per la regolazione di alcuni parametri di rilevazione in sistemi a retroazione.

Si potrebbe esplorare anche la possibilità di una applicazione delle tecniche apprese alle reti quantistiche per vedere se con esse si possano ottenere delle prestazioni migliori.

Bibliografia

[1] I. Aleksander, H. Morton, *An Introduction to Neural Computing.* Chapman and Hall, London, 1990.

[2] N. Aronszajn, *Theory of reproducing kernels.* Trans. Amer. Math. Soc., 68: 337-404, 1950.

[3] D.S. Broomhead, D. Lowe, *Multivariable functional interpolation and adaptive networks.* Complex Systems, 2: 321-355, 1988.

[4] P.G. Casazza, *The art of frame theory.* Taiwanese Journal of Mathematics, 4(2): 129-201, 2000.

[5] P. Courrieu, *Fast Computation of Moore-Penrose Inverse Matrices.* Neural Information Processing - Lett. and Rev., 8(2): 25-29, 2005.

[6] F. Cucker, S. Smale, *On the mathematical foundations of learning.* Bull. Amer. Math. Soc. (N.S.), 39: 1-49, 2001.

[7] G. Cybenko, *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals and Systems, 2(4): 303-314, 1989.

[8] J. Demmel, *The geometry of ill-conditioning.* J. Complexity, 3: 201-229, 1987.

[9] T. Evgeniou, M. Pontil, T. Poggio, *Regularization Networks and Support Vector Machines.* Advances in Computational Mathematics, 13(1): 1-50, 2000.

[10] K.I. Funahashi, *On the approximate realization of continuous mapping by neural networks.*
Neural Networks, 2(3): 183-192, 1989.

[11] J.B. Gao, C.J. Harris, S.R. Gunn, *On a Class of Support Vector Kernels Based on Frames in Function Hilbert Spaces.*
Neural Computation, 13: 1975-1994, 2001.

[12] F. Girosi, M. Jones, T. Poggio *Regularization Theory and Neural Networks Architectures.*
Neural Computation, 7: 219-269, 1995.

[13] F. Girosi, *An equivalence between sparse approximation and support vector machines.*
Neural Computation, 10(6): 1455-1480, 1998.

[14] K. Grochenig, *Acceleration of the frame algorithm.*
IEEE Trans. Signal Proc., 41(12): 3331-3340, 1993.

[15] E.J. Hartman, J.D. Keeler, J.M. Kowalski, *Layered neural networks with Gaussian hidden units as universal approximators.*
Neural Networks, 2(2): 210-215, 1990.

[16] H. Hochstadt, *Integral equations.*
John Wiley & Sons, 1973.

[17] K. Hornik, M. Stinchcombe, H. White, *Multilayer feedforward networks are universal approximators.*
Neural Networks, 2(5): 359-366, 1989.

[18] B. Kröse, P. Smagt, *An introduction to Neural Networks.*, 1996.
Presso: http://citeseer.ist.psu.edu/ose96introduction.html.

[19] V. Kurkova, *Supervised learning as in inverse problem.*
Technical Report 960, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2004.

[20] S. Li, *On general frame decompositions.*
Numer. Funct. Anal. Optim., 16, (9-10): 1181-1191, 1995.

[21] D.J.C. MacKay, *Introduction to gaussian processes.*, 1997.
Presso: http://www.inference.phy.cam.ac.uk/mackay.

[22] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms.*
Cambridge University Press, 2003.

[23] W.S. McCulloch, W. Pitts, *A logical calculus of the ideas immanent in nervous activity.*
Bulletin of Mathematical Biophysics, 5: 115-133, 1943.

[24] M. Minsky, S. Papert, *Perceptrons: An introduction to Computational Geometry.*
The MIT Press, 1969.

[25] S. Mukherjee, P. Niyogi, T. Poggio, R. Rifkin, *Statistical Learning: CV_{loo} stability is sufficient for generalization and necessary and sufficient for consistency of Empirical Risk Minimization.*, 2003.
Presso: http://citeseer.ist.psu.edu/mukherjee03statistical.html.

[26] S. Mukherjee, *Statistical Learning: Algorithms and Theory.*, 2004.
Presso: http://www.stat.duke.edu/∼sayan/statlearn.pdf.

[27] T. Poggio, R. Rifkin, S. Mukherjee, P. Niyogi, *General conditions for predictivity in learning theory.*
Nature, 428: 419-422, 2004.

[28] T. Poggio, S. Smale, *The Mathematics of Learning: Dealing with Data.*
Notices of the AMS, 50(5): 537-544, 2003.

[29] T. Poggio, T. Torre, C. Koch, *Computational vision and regularization theory.*
Nature, 317(26): 314-319, 1985.

[30] M.A. Rakha, *On the Moore-Penrose generalized inverse matrix.*
Applied Mathematics and Computation, 158: 185-200, 2004.

[31] A. Rakotomamonjy, S. Canu, *Frames, Reproducing Kernels, Regularization and Learning.*
Journal of Machine Learning Research, 6: 1485-1515, 2005.

[32] R.D. Richtmyer, *Principles of advanced mathematical physics.*
Springer-Verlag, Berlin, 1978.

[33] R. Rojas, *Neural Networks, A systematic introduction.*
Springer-Verlag, Berlin, 1996.

[34] F. Rosenblatt, *The perceptron, a probabilistic model for information storage and organization in the brain.*
Psychological Review, 65: 386-408, 1958.

[35] F. Rosenblatt, *Principles of Neurodynamics.*
New York: Spartan Books, 1959.

[36] D.E. Rumelhart, G.E. Hinton e R.J. Williams, *Learning representations by back-propagating errors.*
Nature, 323: 533-536, 1986.

[37] A. Smola, B. Schölkopf, K. Müller, *The connection between regularization operators and support vector kernels.*
Neural Networks, 11: 637-649, 1998.

[38] A. Tikhonov, V. Arsénin. *Solutions of Ill-posed problems.*
W.H. Winston, 1977.

[39] V.N. Vapnik, *An Overview of Statistical Learning Theory.*
IEEE transactions on neural networks, 10(5): 988-999, 1999.