Metodi di ottimizzazione per le reti neurali

L. Grippo (grippo@dis.uniroma1.it)
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Buonarroti 12, 00185 Roma

M. Sciandrone (sciandro@iasi.rm.cnr.it)
Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Viale Manzoni 30, 00185 Roma

Abstract In questo lavoro viene considerato il problema dell'apprendimento supervisionato per alcune classi di reti neurali, e viene illustrata l'applicazione dei metodi di ottimizzazione non lineare alla soluzione dei problemi di addestramento. Vengono descritti gli algoritmi più significativi ed evidenziati alcuni risultati recenti che appaiono di interesse nella soluzione di problemi fortemente non lineari e di problemi a grande dimensione.

Keywords: Reti neurali, ottimizzazione non lineare.

1 Introduzione

In termini molto generali, una *rete neurale artificiale* è un processore distribuito, ispirato ai principi di funzionamento del sistema nervoso degli organismi evoluti, costituito dalla *interconnessione* di unità computazionali elementari (*neuroni*), con una caratteristica fondamentale: la *conoscenza* è acquisita dall'ambiente attraverso un processo adattativo di *apprendimento* ed è immagazzinata nei parametri della rete e, in particolare, nei *pesi* associati alle *connessioni* [51]. I neuroni, che si possono vedere come *nodi* di una rete orientata provvisti di capacità di elaborazione, ricevono in ingresso una combinazione dei segnali provenienti dall'esterno o dalle altre unità e ne effettuano una trasformazione tramite una funzione, tipicamente *non lineare*, detta *funzione di attivazione*. L'uscita di ciascun neurone viene poi inviata agli altri nodi oppure direttamente all'uscita della rete, attraverso connessioni orientate e pesate.

Una rete neurale, che può essere costituita da un sistema fisico dotato di un elevato grado di parallelismo e di un'elevata connettività o, più comunemente, da un modello matematico che ne simuli il comportamento, consente di approssimare, in uno specifico contesto applicativo, la corrispondenza esistente (o postulata) tra un *ingresso* e un'*uscita* di natura opportuna. Nei problemi di *classificazione* l'ingresso è costituito dal vettore delle *caratteristiche* dell'oggetto o del fenomeno da classificare e l'uscita è una variabile a valori discreti che esprime l'appartenenza ad una delle classi prefissate. Nei problemi di *approssimazione di funzioni* (o *regressione*, secondo la terminologia della Statistica) l'ingresso è usualmente un vettore di numeri reali che rappresentano le *variabili indipendenti* e ciascuna uscita rappresenta una *variabile dipendente* di un legame funzionale; più in generale, ingressi ed uscite possono anche essere costituiti da vettori di funzioni a valori reali, che rappresentano, ad esempio, andamenti temporali di qualche grandezza.

Il legame ingresso-uscita realizzato dalla rete dipende essenzialmente:

- dal *tipo di unità elementari*, che possono avere struttura interna più o meno complessa ed avere funzioni di attivazione caratterizzate da differenti tipi di nonlinearità;

- dall'*architettura* della rete, ossia dal numero di nodi, dalla struttura e dall'orientamento delle connessioni;

- dai valori dei *parametri* interni associati alle unità elementari e alle connessioni, che devono essere determinati attraverso tecniche di apprendimento.

Sulla base delle caratteristiche precedenti, è possibile distinguere diversi tipi di reti neurali. Una prima distinzione significativa è quella tra reti di tipo *dinamico*, in cui i neuroni sono unità dotate di una dinamica temporale e sono quindi descritte da equazioni differenziali o alle differenze, e reti di tipo *statico*, in cui il legame tra gli ingressi e l'uscita (usualmente scalare) di ciascuna unità viene supposto istantaneo.
Una seconda distinzione importante, relativa alla topologia delle connessioni, è quella tra *reti feedforward*, in cui la struttura delle connessioni può essere rappresentata attraverso un grafo orientato aciclico, e *reti ricorsive*, tipicamente dinamiche, in cui esistono connessioni in *feedback* tra le uscite e gli ingressi di qualche coppia di unità. Ulteriori distinzioni possono essere introdotte in relazione alla metodologia di apprendimento utilizzata per determinare i parametri interni a partire dai dati.

L'apprendimento si basa, in generale, sulla disponibilità di un insieme di addestramento (*training set*), costituito da un insieme di coppie ingresso-uscita, che si possono interpretare come *esempi* della relazione funzionale che si vuole approssimare. Una rete *addestrata* sulla base dei campioni del training set deve essere poi in grado di *generalizzare*, ossia di dare la risposta corretta in corrispondenza a ingressi non considerati nell'insieme di addestramento e ciò costituisce l'uso applicativo della rete in problemi di classificazione o di regressione.

Si possono distinguere due paradigmi fondamentali di apprendimento:

- *apprendimento non supervisionato*, in cui i campioni di uscita non sono noti (oppure non vengono utilizzati), e i parametri della rete vengono determinati attraverso tecniche di *clustering* applicate ai soli campioni di ingresso;

- *apprendimento supervisionato*, in cui i parametri della rete vengono determinati tenendo conto anche delle uscite relative ai campioni di training.

Un'altra distinzione importante, relativa alle metodologie di apprendimento, riguarda le modalità con cui viene acquisito o utilizzato l'insieme di campioni di training durante l'apprendimento; da questo punto di vista si può distinguere:

- *apprendimento on-line*, in cui gli esempi del training set vengono acquisiti (o utilizzati) in modo incrementale durante il processo di addestramento;

- *apprendimento batch* o *fuori linea*, in cui si suppone che tutto l'insieme di addestramento sia disponibile prima che l'addestramento abbia inizio.

Nel seguito prenderemo in considerazione una delle classi più comuni e più studiate di reti neurali, che è quella delle reti *feedforward* di tipo *statico*. Ci riferiremo, in prevalenza, a due delle strutture più note, ossia alle reti *multistrato* (*multilayer feedforward*) e alle reti di *funzioni di base radiali* (*radial basis function networks*) e faremo riferimento esclusivamente al caso in cui la determinazione dei parametri sia effettuata attraverso tecniche di apprendimento supervisionato. L'attività di studio e di ricerca relativa a questa classe di reti riguarda, in particolare:

- le proprietà di approssimazione;

- la teoria dell'apprendimento;

- gli algoritmi di apprendimento per il calcolo dei parametri.

Da un punto di vista deterministico, lo studio dei modelli neurali si può inquadrare nell'ambito della teoria dell'approssimazione e il problema della costruzione del modello si può formulare come un problema di approssimazione funzionale in cui la classe delle funzioni approssimanti è costituita dalle funzioni realizzabili dalla rete, al variare dei parametri interni. La specificità delle reti neurali, nell'ambito dei metodi di approssimazione di tipo più generale, sta nella *particolare scelta delle funzioni approssimanti*, che dipendono tipicamente in modo *non lineare* dai parametri, e nel fatto che i *dati*, rappresentati dalle coppie del training set, sono *discreti* (ossia la funzione da approssimare si suppone nota solo in corrispondenza a un insieme discreto di punti). Tali caratteristiche, che sono spesso alla base dei notevoli successi applicativi delle reti neurali, pongono difficili problemi di analisi e di calcolo.

Uno dei primi problemi affrontati è stato quello della *densità*, ossia quello di stabilire se le funzioni realizzabili attraverso le reti neurali siano degli *approssimatori universali* in una classe di funzioni prefissata (ad esempio quella delle funzioni continue su sottoinsiemi compatti di uno spazio euclideo con la topologia della convergenza uniforme). Per alcune strutture (reti feedforward *a due strati*, reti radial basis) tale problema si ricollega direttamente a risultati già noti nel campo della teoria dell'approssimazione ed è stato risolto in modo completo [82], [85], [109]; alcuni risultati generali verranno riportati nel seguito. Per altre strutture (ad esempio reti feedforward con *tre o più strati*) si pongono problemi nuovi, solo parzialmente studiati, ma esistono alcuni risultati recenti che appaiono particolarmente significativi [67]. Un altro problema rilevante è quello di caratterizzare il *grado di approssimazione*, ossia l'andamento dell'errore di approssimazione, in funzione della dimensione dell'ingresso e del numero di unità che costituiscono la rete; alcuni risultati interessanti sono stati ottenuti imponendo opportune ipotesi di regolarità sulla classe di funzioni in cui si ricerca l'approssimazione [5], [59], [97].

I risultati prima citati sono in genere risultati di esistenza di tipo asintotico e quindi consentono soltanto di stabilire la capacità delle reti di approssimare con la precisione voluta una funzione assegnata al variare del numero di neuroni o delle funzioni di attivazione. Tuttavia, i parametri dei modelli e, almeno parzialmente anche l'architettura, devono essere determinati esclusivamente a partire da un insieme finito di dati, senza conoscere la funzione da approssimare. Si può dire, anzi, che una delle motivazioni principali dell'uso di una rete neurale è proprio quello di tentare di costruire attraverso di essa un legame funzionale tra grandezze che non sono fra loro correlate da un modello analitico di struttura nota. Il problema di approssimazione, in presenza di dati discreti, è, in genere, un *problema mal posto*, nel senso che esistono infinite funzioni in grado di approssimare i dati e che il processo di approssimazione non dipende in modo continuo dai dati. C'è da aggiungere che i dati di training possono essere anche affetti da errori di misura e che il campionamento ottenuto attraverso il training set può risultare sempre meno significativo al crescere della dimensione dell'ingresso.

Lo studio teorico dell'apprendimento e quindi la caratterizzazione delle capacità di generalizzazione dei modelli, costituisce l'oggetto della *teoria dell'apprendimento*, che si può inquadrare, in linea di principio, nell'ambito della Statistica [54], [109]; la specificità delle reti neurali sta, ancora una volta, nella particolarità delle strutture neurali e nella non linearità della dipendenza dai parametri.

In termini generali, il problema dell'apprendimento è quello di definire in modo "ottimo", in un senso da precisare, la *complessità* di un modello e il procedimento di identificazione dei parametri a partire dai dati, in modo da ottenere la migliore capacità di generalizzazione possibile. È infatti evidente che un modello troppo "semplice" potrebbe non essere in grado di descrivere con sufficiente accuratezza il fenomeno in esame, mentre un modello troppo "complesso" potrebbe portare a interpolare esclusivamente i dati di training, a scapito della capacità di generalizzazione.

Ciò implica che la complessità del modello, che si può porre in relazione con il *numero di parametri liberi*, deve essere definita opportunamente, tenendo conto delle ipotesi a priori sul processo da identificare e della cardinalità del training set.

Da un punto di vista statistico, un fondamento concettuale al problema dell'apprendimento a partire da un insieme finito di dati è stato fornito dalla *Statistical Learning Theory* [103], [104], [105], che ha portato anche a definire specifici algoritmi di apprendimento per scegliere in modo "ottimo" sia la struttura dei modelli che il valore dei parametri. Questi contributi hanno dato luogo a un complesso di risultati e di metodi di apprendimento, noti complessivamente come *Support Vector Machines* (SVM, *macchine con vettori di supporto*) [16], [24]. Le tecniche di apprendimento delle SVM, a cui si accennerà in seguito, si pongono spesso in alternativa rispetto alle tecniche neurali e, tuttavia, possono anche essere utilizzate per giustificare alcune metodologie di addestramento di strutture neurali.

Quale che sia l'impostazione concettuale a cui si faccia riferimento, gli algoritmi di addestramento per il calcolo dei parametri dei modelli neurali si basano invariabilmente sulla formulazione di un *problema di ottimizzazione* consistente nella minimizzazione di una funzione di errore di struttura opportuna rispetto ai parametri della rete. L'efficienza degli algoritmi di ottimizzazione nella soluzione dei problemi di addestramento condiziona quindi, in pratica, l'applicabilità dei modelli neurali.

Scopo di questa rassegna è quello di indicare, in corrispondenza alle classi di strutture neurali considerate, i problemi di calcolo che emergono nel contesto dell'addestramento, di illustrare vantaggi e limitazioni degli algoritmi più significativi e di evidenziare alcuni indirizzi di ricerca motivati dalle applicazioni neurali. In particolare, nella Sezione 2 viene descritto il neurone formale e vengono considerati i primi algoritmi di addestramento proposti per reti costituite da un solo strato di neuroni formali. Nella Sezione 3 vengono considerate le reti multistrato, vengono presentati gli algoritmi di ottimizzazione non vincolata utilizzabili per problemi a grande dimensione, e vengono descritte alcune recenti tecniche di tipo non monotono e di tipo incrementale. Nella Sezione 4 vengono descritte le reti di funzioni di base radiali e vengono presentati algoritmi di ottimizzazione basati su metodi di decomposizione. Infine, nella Sezione 5 viene mostrato che il problema di addestramento delle support vector machines si può formulare, mediante la teoria della dualità, come problema di programmazione quadratica convessa con vincoli lineari, e viene riportata una breve panoramica dei metodi di soluzione.

Per un'introduzione di carattere generale sulle reti neurali si rinvia, in particolare, a [13], [51]; sulle applicazioni delle reti neurali alla soluzione di problemi di ottimizzazione si segnalano le monografie [21], [111]; sui rapporti tra Ricerca Operativa e reti neurali si può far riferimento a vari lavori di rassegna, tra cui [17], [98], [110].

2 Il neurone formale

In questa sezione viene descritta una delle versioni più comuni del *neurone formale*, ispirata ai principi di funzionamento del neurone biologico. Viene mostrato come il neurone formale si possa interpretare come un classificatore lineare, i cui parametri possono essere determinati, sotto opportune ipotesi, risolvendo un sistema di disequazioni lineari; vengono quindi messe in evidenza le principali limitazioni di questo modello, che rendono necessaria la considerazione di strutture più complesse.

2.1 Struttura del neurone formale

In una rete neurale l'unità di calcolo elementare è costituita dal *neurone* o *neurone formale*, che esegue una trasformazione, in genere non lineare, di un vettore di *ingressi* x, fornendo in corrispondenza un'*uscita* scalare $y(x)$. Nella struttura più semplice, basata sul modello proposto nel 1943 da McCulloch e Pitts [72] e rielaborato successivamente da Rosenblatt [92], gli ingressi sono moltiplicati per dei *pesi*, rappresentativi dell'entità delle connessioni sinaptiche, e la somma algebrica pesata degli ingressi viene confrontata con un valore di *soglia*; il neurone fornisce l'uscita 1 se la somma pesata degli ingressi è maggiore del valore di soglia, e l'uscita -1 (oppure 0) altrimenti. Assegnato un vettore di ingresso $x \in R^n$, indicando con $w \in R^n$ il vettore dei *pesi*, con $\theta \in R$ il valore di *soglia*, e con $y \in \{-1, 1\}$ l'uscita del neurone, si può porre

$$y(x) = g\left(\sum_{i=1}^{n} w_i x_i - \theta\right) \equiv g(w^T x - \theta),$$

dove g, detta *funzione di attivazione* del neurone, può essere definita, ad esempio, attraverso la *funzione segno*:

$$g(t) \equiv \mathrm{sgn}(t) = \begin{cases} 1 & t \geq 0 \\ -1 & t < 0. \end{cases}$$

Una diversa funzione di attivazione spesso utilizzata è anche la *funzione di Heaviside* (o funzione a gradino) che fornisce un'uscita $g(t) \in \{0, 1\}$.
Uno schema di neurone formale in cui $g(t) \equiv \mathrm{sgn}(t)$ è riportato nella figura seguente.

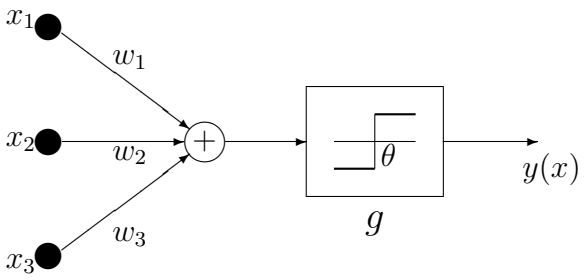


Fig.1 Schema di neurone formale

Si verifica facilmente che con una scelta appropriata dei pesi e della soglia un neurone formale può realizzare le operazioni logiche elementari *not*, *and*, *or*; di conseguenza, ogni funzione logica può essere realizzata mediante una opportuna connessione di neuroni formali.

Un'interpretazione molto più significativa del neurone formale nella versione proposta da Rosenblatt, è tuttavia quella di considerarlo come un *classificatore lineare*, che attribuisce a un generico ingresso x (le cui componenti scalari definiscono le *caratteristiche* dell'oggetto da classificare) il valore $y(x) = 1$ oppure $y(x) = -1$ (e quindi effettua una classificazione binaria), in base al segno della *funzione discriminante* lineare $w^T x - \theta$. L'aspetto più rilevante e innovativo rispetto agli altri modelli di calcolo consiste nel fatto che i valori dei pesi e della soglia possono essere determinati attraverso un processo di *apprendimento* a partire da un *insieme di addestramento*

$$T = \{(x^p, y^p),\ x^p \in R^n,\ y^p \in \{-1, 1\}, \quad p = 1, \ldots, P\}$$

costituito da coppie ingresso-uscita, in cui all'ingresso x^p viene associata la classificazione corretta y^p. Il neurone addestrato usando i campioni dell'insieme T può essere utilizzato successivamente per classificare nuovi ingressi x non appartenenti a T (*generalizzazione*). Si definisce così un strumento per il riconoscimento automatico di configurazioni che è stato denominato *Perceptron*.

I campioni dell'insieme di addestramento saranno classificati in modo corretto se i pesi w e la soglia θ sono determinati in modo tale che risulti

$$\begin{array}{ll} w^T x^p - \theta \geq 0 & \text{se} \quad y^p = \ \ 1, \\ w^T x^p - \theta < 0 & \text{se} \quad y^p = -1. \end{array} \tag{1}$$

Dal punto di vista geometrico, la (1) si può interpretare come la ricerca di un iperpiano di separazione $H = \{x \in R^n :\ w^T x = \theta\}$ che separi gli insiemi

$$\mathcal{A} = \{x^p : (x^p, y^p) \in T, y^p = 1\} \qquad \mathcal{B} = \{x^p : (x^p, y^p) \in T, y^p = -1\}.$$

L'esistenza di w, θ che risolvano il sistema (1) può quindi essere assicurata se e solo se gli insiemi $\mathcal{A}$ e $\mathcal{B}$ sono *linearmente separabili*, ossia se e solo se i rispettivi involucri convessi sono disgiunti. Si verifica anche facilmente che le condizioni (1) ammettono soluzione se e solo se ammette soluzione il sistema

$$\begin{array}{ll} w^T x^p - \theta > 0 & x^p \in \mathcal{A}, \\ w^T x^p - \theta < 0 & x^p \in \mathcal{B}. \end{array} \tag{2}$$

Quindi, aggiungendo un ingresso fittizio $x_0 = -1$ e ridefinendo x e w come vettori a $n + 1$ componenti, ossia ponendo $x = (x_0, x_1, \ldots, x_n)^T$ e $w = (\theta, w_1, \ldots, w_n)^T$, ci si può ricondurre a risolvere rispetto a $w \in R^{n+1}$ il sistema

$$\begin{array}{ll} w^T x^p > 0 & x^p \in \mathcal{A}, \\ w^T x^p < 0 & x^p \in \mathcal{B}, \end{array} \tag{3}$$

in cui, senza perdita di generalità, si può supporre

$$\|x^p\| = 1, \quad p = 1, \ldots P.$$

2.2 L'algoritmo di addestramento del Perceptron

Per la soluzione di (3) è possibile definire un algoritmo di addestramento di tipo incrementale (cioè un algoritmo in cui, ad ogni iterazione, viene utilizzato un solo campione del training set) che consente la determinazione di w a partire da T.

Nello schema seguente, che descrive l'algoritmo di Rosenblatt, l'insieme di addestramento viene preso ripetutamente in considerazione, finché tutti i campioni non sono classificati correttamente. Ad ogni ciclo il vettore w viene aggiornato in corrispondenza a ciascun campione x^p non classificato correttamente, sommando al valore corrente $w(k)$ un termine di correzione.

Algoritmo di addestramento del Perceptron

Dati. Input x^p, con $\|x^p\| = 1$, Target y^p, $p = 1, \ldots, P$.

Inizializzazione. Poni $w(0) = 0$, $k = 0$, $nclass = 0$.

While $nclass < P$ **do**

For $p = 1, \ldots, P$ **do**

If $\text{sgn}(w(k)^T x^p) = y^p$ **then**

poni $nclass = nclass + 1$

else

poni $w(k+1) = w(k) + y^p x^p$

$k = k + 1$

end if

Poni $p = p + 1$

End For

If $nclass < P$ **then** poni $nclass = 0$

end While

Si osservi che, nel corso delle iterazioni, la regola di aggiornamento del vettore dei pesi può rendere non classificati correttamente dei campioni che precedentemente erano ben classificati, e viceversa. Tuttavia, si può dimostrare che se gli insiemi $\mathcal{A}$ e $\mathcal{B}$ sono linearmente separabili, in un numero finito di iterazioni l'algoritmo determina un vettore dei pesi $\bar{w}$ tale che tutti i campioni del training set risultano classificati correttamente, ossia soddisfano

$$y^p = \text{sgn}(\bar{w}^T x^p) \qquad p = 1, \ldots, P. \tag{4}$$

Per dimostrarlo, supponiamo che esista un vettore $\bar{w}$ che soddisfi la (3) e supponiamo, senza perdita di generalità, che sia $\|\bar{w}\| = 1$. Supponiamo che l'algoritmo non termini; ne segue che, per ogni k fissato, esisterà almeno un campione x^p (dipendente da k) non classificato correttamente per cui, in base alle istruzioni dell'algoritmo si avrà:

$$w(k+1) = w(k) + y^p x^p.$$

Effettuando il prodotto scalare di $w(k+1)$ per $\bar{w}$ si ottiene

$$\begin{aligned} \bar{w}^T w(k+1) &= \bar{w}^T w(k) + y^p \bar{w}^T x^p \\ &\geq \bar{w}^T w(k) + \delta, \end{aligned} \tag{5}$$

dove

$$\delta = \min_p \{y^p \bar{w}^T x^p\} = \min_p \{\mathrm{sgn}(\bar{w}^T x^p) \bar{w}^T x^p\} > 0.$$

Poiché la (5) vale per ogni k e $w(0) = 0$, ragionando per induzione e usando la diseguaglianza di Schwarz si può scrivere,

$$\|w(k+1)\| \geq \bar{w}^T w(k+1) \geq (k+1)\delta. \tag{6}$$

D'altra parte, tenendo conto del fatto che $y^p w(k)^T x^p \leq 0$ (poiché x^p non è classificato correttamente) e che $\|y^p x^p\| = 1$, si ha

$$\begin{aligned} \|w(k+1)\|^2 &= \|w(k)\|^2 + 2y^p w(k)^T x^p + \|y^p x^p\|^2 \\ &\leq \|w(k)\|^2 + 1, \end{aligned}$$

il che implica, per induzione

$$\|w(k+1)\|^2 \leq k+1. \tag{7}$$

Dalle (6) e (7) segue

$$\sqrt{(k+1)} \geq (k+1)\delta,$$

ossia $k+1 \leq (1/\delta)^2$, il che contraddice l'ipotesi che l'algoritmo non termini.

L'algoritmo precedente, di cui sono state proposte diverse varianti, è riconducibile a un *metodo di rilassamento* per la soluzione iterativa di sistemi di disequazioni lineari. Si verifica facilmente che il sistema (3) è equivalente a un particolare sistema del tipo

$$Aw \geq b, \tag{8}$$

in cui $b \in R^P$ ha componenti unitarie e la matrice $A(P \times n + 1)$ è definita da

$$A^T = (a_1, \ldots, a_P) = \left(\sigma_1 x^1, \sigma_2 x^2, \ldots, \sigma_P x^P\right),$$

dove si è posto:

$$\sigma_p = 1, \quad \text{se} \quad x^p \in \mathcal{A}, \qquad \sigma_p = -1 \quad \text{se} \quad x^p \in \mathcal{B}.$$

Il metodo di rilassamento per la soluzione di un qualsiasi sistema del tipo (8) è un metodo iterativo introdotto nel 1954 da Agmon [1] e Motzkin - Schoenberg [77], in cui ad ogni iterazione viene selezionata una disequazione violata e viene aggiornata la stima corrente $w(k)$ mediante l'iterazione

$$w(k+1) = w(k) + \lambda \frac{\left(b_k - a_k^T w(k)\right)}{||a_k||^2} a_k, \tag{9}$$

dove λ è un parametro scalare. È facile verificare che la (9) fornisce un criterio di aggiornamento del tutto analogo a quello considerato nell'algoritmo di addestramento del Perceptron. Si dimostra, in particolare, che se $\lambda \in (0, 1)$ e il sistema (8) ha soluzione, l'algoritmo di rilassamento, a partire da un punto $w(0)$ arbitrario termina in un punto ammissibile, oppure genera una successione convergente a una soluzione sulla frontiera dell'insieme ammissibile. È noto anche che la complessità computazionale può essere esponenziale nel caso peggiore (cfr., ad esempio, [93]).

Il problema dell'addestramento del Perceptron può tuttavia essere risolto completamente mediante la programmazione lineare, utilizzando "contemporaneamente" tutti i dati del training set T (si deve perciò supporre di disporre di tutti i campioni di T prima di effettuare l'addestramento). In particolare è possibile impiegare il metodo delle variabili artificiali, oppure trasformare in un problema di programmazione lineare il problema di minimizzare una norma poliedrale dell'errore. Se i coefficienti sono razionali e si adotta un metodo interno si può determinare w in tempo polinomiale, oppure concludere che il sistema (8) non ha soluzione.

Un'impostazione alternativa è quella di sostituire la funzione di attivazione $\mathrm{sgn}(t)$ con una funzione continuamente differenziabile g di tipo *sigmoidale*, ossia una funzione *monotona crescente* tale che:

$$\lim_{t \to -\infty} g(t) = -1, \quad \lim_{t \to \infty} g(t) = 1,$$

come, ad esempio la *funzione tangente iperbolica*

$$\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}},$$

e minimizzare rispetto a w l'errore quadratico

$$E(w) = \frac{1}{2} \sum_{p=1}^{P} \left(y^p - g(w^T x^p)\right)^2.$$

La scelta fatta per g assicura che E sia una funzione continuamente differenziabile e quindi rende possibile l'impiego di metodi efficienti di ottimizzazione non lineare. Gli algoritmi utilizzabili (di tipo batch o di tipo incrementale) verranno discussi nel seguito in relazione a strutture più generali.

2.3 Le limitazioni del Perceptron

Pur essendo possibile risolvere efficientemente il problema dell'addestramento del Perceptron nelle ipotesi considerate, un classificatore basato sul neurone formale ha limitate possibilità di applicazione, in quanto esistono semplici problemi di classificazione in cui gli insiemi di campioni non sono linearmente separabili. Un esempio molto noto è quello dell'OR esclusivo (XOR), in cui gli ingressi sono binari e si richiede che l'uscita sia 1 se e solo se *soltanto uno* degli ingressi è 1. Alcune di tali limitazioni possono essere superate, in linea di principio, effettuando una trasformazione dell'ingresso mediante un insieme di funzioni $\phi_j : R^{n+1} \rightarrow R$ che consentano, se scelte appropriatamente, di ricondursi a insiemi linearmente separabili nello spazio trasformato. In tal caso, la funzione ingresso-uscita del neurone formale diviene:

$$y(x) = g\left(\sum_{j=1}^{n+1} w_j \phi_j(x)\right).$$

Tale possibilità, già prevista nel Perceptron di Rosenblatt, è ancora soggetta a notevoli limitazioni, almeno finchè si pongono limiti sul numero o sulla complessità delle funzioni ϕ_j e gli unici parametri determinabili con criteri adattativi sono i pesi w_j. Ciò è stato dimostrato, in relazione a diverse scelte delle funzioni ϕ_j da Minsky e Papert in un famoso libro [75], che ha avuto l'effetto di far decadere l'interesse iniziale della comunità scientifica verso le reti neurali. Per superare le limitazioni del Perceptron, è necessario utilizzare strutture più complesse, facendo dipendere la struttura delle funzioni ϕ_j dal processo di addestramento.

3 Reti multistrato

Le limitazioni del Perceptron, ossia di reti costituite da un *solo strato* adattativo di neuroni formali hanno motivato lo studio di architetture costituite da più strati di neuroni connessi in cascata, denominate reti *multistrato* (*multilayer feed-forward, multilayer perceptron*). Reti di questo tipo consentono, in linea di principio, sotto ipotesi opportune sulle funzioni di attivazione dei neuroni, di approssimare con la precisione voluta una qualsiasi funzione continua su un insieme compatto e, quindi anche, in particolare, di risolvere problemi di classificazione di insiemi non separabili linearmente. La determinazione dei parametri attraverso il processo di addestramento diviene tuttavia un problema più complesso e richiede l'impiego di tecniche di ottimizzazione non lineare.

In questa sezione, dopo brevi cenni sulla struttura delle reti multistrato e sulle proprietà di approssimazione, viene illustrata la formulazione del problema di addestramento come problema di ottimizzazione, vengono discussi i problemi computazionali e gli algoritmi utilizzabili e, infine, viene descritta una tecnica di stabilizzazione di tipo non monotono che è stata proposta nell'ambito di metodi di ottimizzazione e può essere impiegata per la definizione di algoritmi di addestramento di reti neurali.

3.1 Architettura e notazioni

L'architettura di una rete neurale *multistrato* può essere descritta definendo:

- un insieme di n nodi di ingresso, sprovvisti di capacità di elaborazione, associati agli n *ingressi* della rete: $x_i \in R, \quad i = 1, \ldots, n$;
- un insieme di *neuroni* (formali) organizzati in $L \geq 2$ diversi *strati* di cui:
  - $L - 1$ *strati nascosti* (*hidden layers*), costituiti ciascuno da neuroni le cui uscite contribuiscono agli ingressi dei neuroni dello strato successivo;
  - uno *strato di uscita* costituito da $K \geq 1$ neuroni le cui uscite costituiscono le *uscite* della rete $y_i \in R, \quad i = 1, \ldots, K$;
- un insieme di archi orientati e pesati che rappresentano le connessioni interneurali e le connessioni con i nodi di ingresso; si suppone che non esistano connessioni tra i neuroni di uno stesso strato, né connessioni in feedback tra le uscite dei neuroni di uno strato e gli ingressi dei neuroni degli strati precedenti.

La struttura di una rete multistrato è illustrata schematicamente in fig.2.

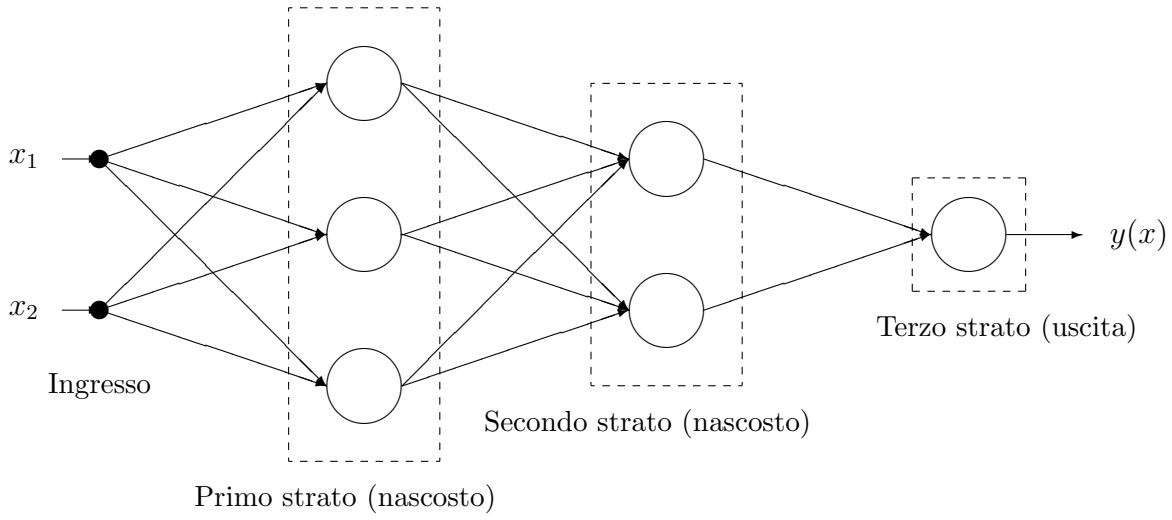
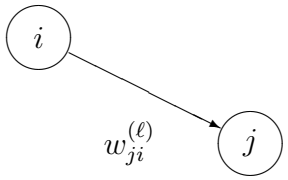


Fig.2 Rete neurale a 3 strati, con 2 strati nascosti, 2 ingressi, 1 uscita

Indicheremo con $\ell = 1, \ldots, L$ gli indici associati ai vari strati. A ciascun arco orientato entrante nel neurone j dello strato ℓ e uscente dal neurone i dello strato $\ell - 1$ oppure dal nodo di ingresso i, è associato un *peso* costituito da un numero reale $w_{ji}^{(\ell)}$ che rappresenta l'entità della *connessione sinaptica*.



Ciascun neurone formale si suppone caratterizzato da una *funzione di attivazione* $g_j^{(\ell)} : R \to R$ che opera su una combinazione pesata degli ingressi e di un valore di soglia $w_{j0}^{(\ell)}$. Indicando con $a_j^{(\ell)}$ la somma pesata degli ingressi e della soglia e con $z_j^{(\ell)}$ l'uscita del neurone, per il neurone j dello strato 1 si può scrivere:

$$a_j^{(1)} = \sum_{i=1}^{n} w_{ji}^{(1)} x_i - w_{j0}^{(1)}, \qquad z_j^{(1)} = g_j^{(1)}(a_j^{(1)}) \tag{10}$$

e per il neurone j di uno strato $\ell > 1$ si ha:

$$a_j^{(\ell)} = \sum_{i=1}^{N^{(\ell-1)}} w_{ji}^{(\ell)} z_i^{(\ell-1)} - w_{j0}^{(\ell)}, \qquad z_j^{(\ell)} = g_j^{(\ell)}(a_j^{(\ell)}) \tag{11}$$

avendo indicato con $N^{(\ell)}$ il numero di neuroni dello strato ℓ.

Nel seguito ci riferiremo spesso al caso di una rete a 2 strati con un solo strato nascosto di N neuroni con funzione di attivazione g e un solo neurone d'uscita di tipo lineare. In tali ipotesi, le notazioni possono essere molto semplificate. In particolare, adotteremo le notazioni seguenti

- w_{ji}: pesi delle connessioni tra i nodi di ingresso e lo strato nascosto;
- θ_j: soglia del neurone nascosto j;
- v_j: pesi delle connessioni tra i neuroni dello strato nascosto e il neurone d'uscita;
- g: funzione di attivazione dei neuroni dello strato nascosto.

Possiamo quindi porre:

$$y = \sum_{j=1}^{N} v_j g\left(\sum_{i=1}^{n} w_{ji} x_i - \theta_j\right) = \sum_{j=1}^{N} v_j g\left(w_j^T x - \theta_j\right),$$

dove

$$w_j = (w_{j1}, \ldots, w_{jn})^T.$$

La funzione di attivazione g si suppone usualmente differenziabile e sigmoidale. Le funzioni piu comuni sono la *funzione logistica*

$$g(t) \equiv \frac{1}{1+e^{-ct}}, \quad c > 0$$

che fornisce un'uscita in $(0,1)$ e la *funzione tangente iperbolica*

$$g(t) \equiv \tanh(t/2) = \frac{1-e^{-t}}{1+e^{-t}},$$

che dà un'uscita in (-1.1).

Lo schema di una rete a due strati, in cui è evidenziata la struttura interna dei neuroni è riportato in Fig.4.

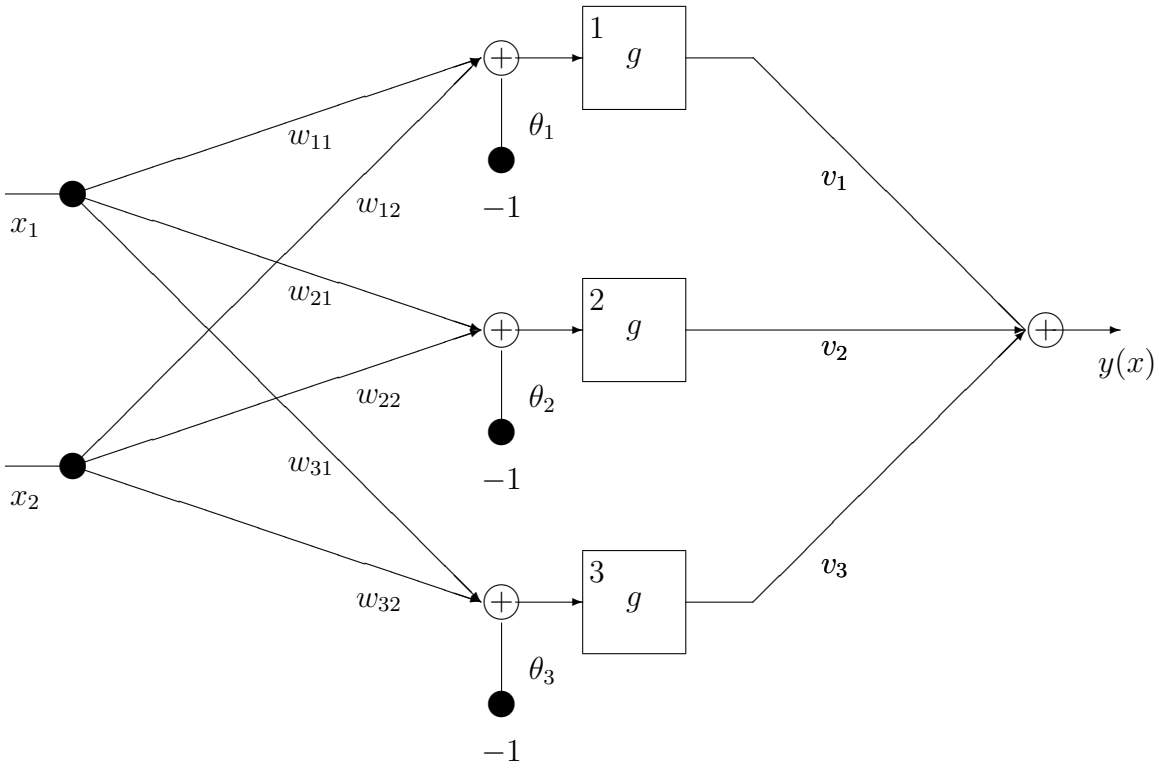


Fig.3 Rete neurale a 2 strati, con 1 strato nascosto, 2 ingressi, 1 uscita

3.2 Proprietà di approssimazione

Le proprietà di approssimazione delle reti multistrato sono state oggetto di numerosi studi. In particolare, è stato dimostrato che le reti a 2 strati con 1 strato nascosto sono *approssimatori universali* per le funzioni continue su insiemi compatti di R^n, per un'ampia classe di funzioni di attivazione. Ponendo:

$$\mathcal{M}(g) = \text{span}\{g(w'x - \theta), \theta \in R, w \in R^n\},$$

(ossia indicando con $\mathcal{M}(g)$ l'insieme di tutte le combinazioni lineari delle funzioni ottenute applicando la funzione di attivazione g a una trasformazione affine di x definita da w e θ), vale il risultato seguente [81].

Teorema 1 (Pinkus, 1996) *Sia $g \in C(R)$. Allora $\mathcal{M}(g)$ è denso in $C(R^n)$, nella topologia della convergenza uniforme sugli insiemi compatti se e solo se g* non è un polinomio.

Ne segue che, dati una qualsiasi funzione $f \in C(R^n)$, un qualsiasi insieme compatto $\Omega \subset R^n$ e un qualsiasi $\varepsilon > 0$, e fissata una funzione di attivazione g qualsiasi (purché continua e non polinomiale) si può costruire una rete a 2 strati (con scelta opportuna del numero dei neuroni, e dei valori dei pesi e delle soglie), tale che la funzione ingresso-uscita $y \in \mathcal{M}(g)$ realizzata dalla rete soddisfi:

$$\max_{x \in \Omega} |f(x) - y(x)| < \varepsilon.$$

Altri risultati recenti consentono di restringere la scelta di w e θ e anche di allargare l'insieme delle funzioni di attivazione ammissibili. Le reti a 2 strati consentono anche, in linea di principio, di interpolare esattamente i dati, nel senso precisato nel teorema successivo [82].

Teorema 2 (Pinkus, 1999) *Sia $g \in C(R)$ e si assuma che g non sia un polinomio. Dati K punti distinti $\{x^i\}_{i=1}^K \subset R^n$ e K numeri $\{\alpha_i\}_{i=1}^K \subset R$, esistono K vettori $\{w_j\}_{j=1}^K \subset R^n$ e $2K$ numeri $\{v_j\}_{j=1}^K, \{\theta_j\}_{j=1}^K \subset R$ tali che*

$$\sum_{j=1}^{K} v_j g(w_j^T x^i - \theta_j) = \alpha_i, \quad i = 1, \ldots, K.$$

Molti studi recenti sono stati dedicati alla stima del *grado di approssimazione*, in funzione di n, della struttura della rete e delle ipotesi sullo spazio di funzioni in cui il problema è formulato (cfr., ad esempio [5], [59], [82], [97]).

Un risultato interessante [67] è che per reti a 3 strati, in cui si assume che la funzione ingresso-uscita sia del tipo

$$y = \sum_{i=1}^{N_2} u_i g \left(\sum_{j=1}^{N_1} v_{ij} g(w_{ij}^T x - \theta_{ij}) - \gamma_i \right),$$

non esistono lower bound sull'errore di approssimazione e vale il risultato seguente.

Teorema 3 (Maiorov-Pinkus, 1999) *Esiste una funzione di attivazione $g \in C^\infty$, strettamente crescente e sigmoidale, tale che per ogni funzione $f \in C([0,1]^n)$ e ogni $\varepsilon > 0$, esistono costanti u_i, v_{ij}, θ_{ij}, γ_i e vettori $w_{ij} \in R^n$, per cui risulta:*

$$\left| f(x) - \sum_{i=1}^{4n+3} u_i g \left(\sum_{j=1}^{2n+1} v_{ij} g(w_{ij}^T x - \theta_{ij}) - \gamma_i \right) \right| < \varepsilon,$$

per ogni $x \in [0,1]^n$.

Il teorema precedente è basato sul famoso *Teorema di Kolmogorov* (1957), che ha risolto (in senso negativo) la congettura di Hilbert sull'esistenza di funzioni continue di tre variabili, non rappresentabili come composizione di funzioni continue di due variabili (*13-mo problema di Hilbert*). Nel Teorema di Kolmogorov si dimostra che le funzioni continue di n varabili si possono *rappresentare* (e non soltanto *approssimare*) come sovrapposizione di funzioni di una variabile. Una delle versioni più semplici è riportata nel teorema seguente [65].

Teorema 4 (Lorentz, 1976) *Esistono n costanti $\lambda_j > 0$ tali che $\sum_{j=1}^n \lambda_j \leq 1$ e $2n+1$ funzioni continue strettamente crescenti $\phi_i : [0,1] \to [0,1]$ tali che ogni funzione $f \in C([0,1]^n)$ si può rappresentare nella forma:*

$$f(x_1 \ldots, x_n) = \sum_{i=1}^{2n+1} g \left(\sum_{j=1}^{n} \lambda_j \phi_i(x_j) \right),$$

per qualche $g \in C[0,1]$ dipendente da f.

Il ruolo (controverso) del Teorema di Kolmogorov nello studio delle reti neurali è stato oggetto di vari lavori. Alcune delle versioni del teorema sembrano suggerire la rappresentazione di una funzione attraverso una struttura di rete multistrato, in cui però la funzione (di attivazione) g è incognita e può avere un andamento molto irregolare. Ciò potrebbe rendere scarsamente significativa l'interpretazione neurale; questa è la tesi sostenuta in [34]. D'altra parte, in vari lavori recenti, come ad esempio [58], [82], [100], il teorema di Kolmogorov viene utilizzato per stabilire risultati di approssimazione attraverso reti multistrato e quindi ha un ruolo significativo.

3.3 Il problema dell'addestramento

La costruzione di una rete multistrato con n ingressi e K uscite consiste nello *scegliere l'architettura* (numero di strati e numero di neuroni di ogni strato), e nell'*addestrare* la rete, ossia nel determinare il vettore $w \in R^m$ le cui componenti corrispondono ai parametri incogniti (pesi e soglie dei neuroni nei vari strati).

La scelta dei parametri, per una architettura fissata, viene in genere effettuata definendo un opportuno sottoinsieme dei dati disponibili

$$T = \{(x^p, y^p),\ x^p \in R^n,\ y^p \in R^K,\ p = 1, \ldots, P\},$$

che costituisce *il training set* e risolvendo successivamente un problema di ottimizzazione del tipo:

$$\min_{w \in R^m} E(w) = \sum_{p=1}^{P} E_p(w),$$

in cui E_p è il termine di errore relativo al $p-$mo campione e misura la distanza tra l'uscita desiderata y^p e l'uscita $y(x^p; w)$ fornita dalla rete. La misura più usata è l'errore quadratico

$$E_p(w) = \frac{1}{2}\|y(x^p; w) - y^p\|^2,$$

ma è possibile considerare anche funzioni di errore di struttura diversa [99]. Nel seguito ci limiteremo a supporre che E sia una funzione *continuamente differenziabile*.

Come si è detto in precedenza, scopo dell'addestramento non è quello di interpolare i dati di training, quanto piuttosto quello di modellare il processo che ha generato i dati. Ciò implica che la scelta dell'architettura, la selezione dei dati da includere in T, la definizione di E e la strategia di addestramento devono tener conto dell'esigenza di assicurare buone capacità di generalizzazione. Dal punto di vista teorico, uno dei problemi più importanti è quello di definire opportunamente la complessità del modello, e quindi il numero di parametri liberi, in relazione ai dati disponibili. Per le reti multistrato, a partire dai risultati sulla teoria statistica dell'apprendimento [103], [104], sono state stabilite delle stime del numero minimo dei campioni di training occorrenti per addestrare una rete in modo tale che si abbia una "buona" capacità di generalizzazione [4], [8].

In pratica, tuttavia le stime teoriche possono essere inadeguate ed occorre basarsi su opportune euristiche per la scelta della struttura e la definizione del training set. In linea di massima, nel caso delle reti multistrato, vengono seguite due strategie fondamentali. La prima, chiamata *stabilizzazione strutturale* [13] consiste nello scegliere il numero di unità, attraverso l'addestramento di una sequenza di reti in cui viene fatto crescere (o diminuire) il numero di neuroni. Per ciascuna di tali reti i parametri vengono determinati minimizzando l'errore sul training set e le prestazioni delle varie reti sono confrontate attraverso una tecnica di *cross-validation*, valutando l'errore che ogni rete commette su un altro insieme di dati (*validation set*) non inclusi in T. La rete selezionata è quella che fornisce l'errore minimo sul validation set.

Le prestazioni di una rete addestrata vengono valutate utilizzando un terzo insieme di dati denominato *test set*, che non deve essere stato utilizzato né per la scelta dell'architettura, né per la determinazione dei parametri.

La seconda strategia si basa su una tecnica di *regolarizzazione* e consiste nell'aggiungere alla funzione di errore un termine di penalità sulla norma di w che ha l'effetto di restringere l'insieme entro cui vengono scelti i parametri. Ciò equivale, essenzialmente, ad imporre condizioni di regolarità sulla classe di funzioni realizzata dalla rete [36]. L' addestramento della rete viene effettuato definendo la nuova funzione obiettivo

$$E(w) = \sum_{p=1}^{P} E_p(w) + \gamma \|w\|^2,$$

con $\gamma > 0$. Il valore "ottimale" di γ può essere determinato utilizzando, anche in questo caso, una tecnica di cross-validation. In particolare, in corrispondenza a differenti valori di γ, si addestrano varie reti minimizzando la funzione d'errore E e viene successivamente prescelto il valore di γ a cui corrisponde il minimo errore sul validation set.

In alternativa alla tecnica di regolarizzazione, una strategia euristica talvolta utilizzata è quella di interrompere prematuramente la minimizzazione (*early stopping*) della funzione d'errore. In particolare, la tecnica di early stopping si basa sull'idea di valutare periodicamente, durante il processo di minimizzazione, l'errore che la rete commette su un validation set ausiliario. In generale, nelle prime iterazioni l'errore sul validation set diminuisce con la funzione obiettivo, mentre può aumentare se l'errore sul training set diviene "sufficientemente piccolo". Il processo di addestramento termina quindi quando l'errore sul validation set inizia ad aumentare, perchè ciò potrebbe evidenziare l'inizio della fase di *overfitting* della rete, cioè della fase in cui la rete tende a interpolare i dati di training a scapito della capacità di generalizzazione.

Quale che sia la strategia di addestramento seguita, pur non essendo possibile ridurre banalmente i problemi di addestramento alla soluzione di un problema di ottimizzazione, è necessario, in pratica, ripetere la minimizzazione in corrispondenza a varie architetture o a varie funzioni di errore. La disponibilità di algoritmi efficienti di ottimizzazione costituisce, quindi, uno strumento essenziale per la costruzione delle reti neurali. La minimizzazione dell'errore di training E è, in generale, un difficile problema di ottimizzazione non lineare, in cui le difficoltà computazionali sono tipicamente dovute a

- forti non linearità di E, che creano "valli ripide" e zone "piatte" nella superficie della funzione di errore;
- possibile mal condizionamento dell'Hessiana;
- elevata dimensionalità di w ed elevato numero P di campioni;
- presenza di minimi locali non globali.

Una ulteriore difficoltà è costituita dal fatto che gli insiemi di livello della funzione d'errore, ossia gli insiemi

$$\mathcal{L}(\alpha) = \{w \in R^m : E(w) \leq \alpha\}$$

possono non essere compatti, per cui non è possibile assicurare la convergenza globale di molti algoritmi a partire da punti iniziali arbitrari. È da notare, tuttavia, che in presenza di un termine di regolarizzazione *tutti* gli insiemi di livello sono compatti.

3.4 Generalità sugli algoritmi di addestramento

Uno dei primi algoritmi proposti per il calcolo dei pesi in una rete neurale è il metodo iterativo noto come *metodo di backpropagation* [94], [95] che è interpretabile come una versione euristica del *metodo del gradiente*. La scoperta (o meglio la riscoperta) di questo metodo verso la metà degli anni '80 ha reso possibile definire algoritmi di addestramento per reti multistrato e quindi è stata alla base del successivo considerevole sviluppo degli studi sulle reti neurali negli ultimi due decenni.

Sono state introdotte, in particolare, due classi principali di metodi iterativi per il calcolo dei pesi:

- metodi *batch* in cui ad ogni passo i pesi vengono aggiornati utilizzando informazioni relative a tutti i campioni dell'insieme di addestramento T;

- metodi *on-line* in cui ad ogni passo i pesi vengono aggiornati tenendo conto soltanto di un singolo campione di T.

I metodi batch possono essere utilizzati esclusivamente per l'addestramento *fuori linea*, supponendo di disporre di tutto l'insieme T prima di avviare il processo di minimizzazione. I metodi *on-line* possono essere impiegati sia per l'addestramento fuori linea, sia per l'addestramento in tempo reale, cioè quando gli elementi di T vengono acquisiti durante il processo di addestramento.

I metodi batch sono ovviamente riconducibili a metodi di ottimizzazione non vincolata per la minimizzazione di E. Infatti, nella letteratura neurale più recente sono sempre più frequentemente utilizzati, in luogo dei primi metodi euristici, metodi efficienti già sviluppati da tempo nel campo dell'ottimizzazione, che garantiscono la convergenza a *punti stazionari* della funzione di errore e usualmente assicurano una buona riduzione dell'obiettivo rispetto alla stima iniziale. I metodi più appropriati per costruire algoritmi di addestramento sono quelli che richiedono esclusivamente la conoscenza delle derivate prime e che possono essere impiegati in problemi a "grande dimensione", ossia quando il numero di variabili è dell'ordine di qualche migliaio. I metodi più "classici" con questi requisiti comprendono: i *metodi del gradiente*, i *metodi delle direzioni coniugate*, i *metodi Quasi-Newton a memoria limitata* e i *metodi tipo Gauss-Newton*. Per problemi di dimensione non molto elevata sono stati anche utilizzati *metodi tipo-Newton* troncati, basati sull'uso delle derivate seconde di E [99].

Il limite principale dei metodi classici nei problemi di addestramento "difficili" risiede nel requisito di *monotonicità* sulla riduzione di E che deve esere "sufficiente" e deve essere assicurata attraverso costose ricerche unidimensionali. Ciò può comportare costi computazionali inaccettabili, soprattutto in presenza di "valli ripide" a forte curvatura (difficoltà tipica dei problemi non lineari con matrice Hessiana mal condizionata). Un'alternativa molto promettente è costituita da metodi di tipo *non monotono* che consentono anche una crescita occasionale della funzione obiettivo, pur continuando ad assicurare le stesse proprietà di convergenza globale dei metodi monotoni usuali e una riduzione dell'obiettivo rispetto al valore iniziale. La crescente diffusione di tali tecniche, che si basano in prevalenza sull'uso della ricerca unidimensionale non monotona proposta in [41], eventualmente associata a tecniche di tipo *watchdog* [20], ha consentito, nell'ambito dell'ottimizzazione non vincolata, di realizzare versioni non monotone efficienti dei metodi tipo-Newton [41] [42] e Gauss-Newton [29] [60] e, più recentemente, di globalizzare il *metodo del gradiente di Barzilai-Borwein* [91]. Questa versione non monotona del metodo del gradiente, sperimentata con successo in problemi di addestramento di reti neurali [83], ha anche il vantaggio di consentire la possibilità di sfuggire all'attrazione di minimi locali irrilevanti. Una nuova strategia di globalizzazione [48], che combina nuove tecniche di ricerca unidimensionale non monotone con tecniche di tipo watchdog non monotone, consente di migliorare ulteriormente il comportamento del metodo di Barzilai Borwein e appare di particolare interesse nella realizzazione di nuovi algoritmi di addestramento.

In associazione ai metodi di minimizzazione considerati, che garantiscono la convergenza a punti stazionari, sono state anche utilizzate teniche di *ottimizzazione globale* sia di tipo probabilistico, come ad esempio tecniche *multistart* o tecniche di *simulated annealing* [57], sia di tipo deterministico come metodi di *tunnelling* [19] o metodi di *omotopia* [23], [49].

I metodi on-line possono essere interpretati, in un contesto deterministico, come algoritmi di ottimizzazione *incrementali*, in cui ad ogni iterazione si utilizzano informazioni parziali sulla funzione obiettivo e sulle sue derivate. Gli studi più recenti su questa classe di metodi nel campo dell'ottimizzazione è stata in gran parte motivata dalle applicazioni neurali [9], [12].

Nei paragrafi successivi saranno illustrati in maggior dettaglio alcuni degli algoritmi citati e saranno indicati alcuni indirizzi di ricerca che appaiono promettenti.

3.5 Backpropagation e momentum

Il metodo di *backpropagation* (BP) è tuttora uno dei metodi di addestramento più diffusi. Il termine "backpropagation" (retropropagazione) è legato essenzialmente alla tecnica utilizzata per il calcolo delle derivate della funzione di errore, basata sulle regole di derivazione delle funzioni composte. Una tecnica analoga era stata utilizzata da tempo nei problemi di controllo ottimo per esprimere il gradiente rispetto al controllo, senza ricavare la dipendenza esplicita dello stato dal controllo [15].

La procedura di calcolo del gradiente mediante backpropagation, che si può ricondurre a una particolare tecnica di *differenziazione automatica* [40] verrà illustrata nel paragrafo successivo.

Il metodo di BP è stato utilizzato in due versioni, note rispettivamente come:

- BP *batch*, in cui i pesi vengono aggiornati dopo la presentazione di tutti i campioni del training set T;

- BP *on-line*, in cui i pesi vengono aggiornati in corrispondenza a ciascun campione di T.

La BP batch è definita dall'iterazione

$$w^{k+1} = w^k - \eta \nabla E(w^k),$$

dove $\nabla E(w^k)$ è il *gradiente* di E nel vettore corrente w^k e lo scalare $\eta > 0$ (detto *learning rate*) definisce il passo lungo l'antigradiente.

La BP on-line consiste invece nel selezionare ad ogni passo un campione $(x^{p(k)}, y^{p(k)})$ dell'insieme di addestramento e nell'aggiornare i pesi utilizzando soltanto il termine $\nabla E_{p(k)}$ del gradiente di E, ossia nel porre:

$$w^{k+1} = w^k - \eta \nabla E_{p(k)}(w^k).$$

Il metodo di BP è in genere inefficiente e può non convergere se il passo η non è scelto in modo appropriato; nella letteratura neurale sono state quindi considerate varie tecniche euristiche per effettuare una scelta adattativa del passo e per modificare la direzione di ricerca [21], [51], [99].

Una modifica spesso efficace è la cosiddetta *momentum updating rule*, la cui versione batch è definita dall'iterazione

$$w^{k+1} = w^k - \eta \nabla E(w^k) + \beta(w^k - w^{k-1}),$$

dove $\beta > 0$ è uno scalare fissato. Analogamente, la versione on-line può essere posta nella forma

$$w^{k+1} = w^k - \eta \nabla E_{p(k)}(w^k) + \beta(w^k - w^{k-1}).$$

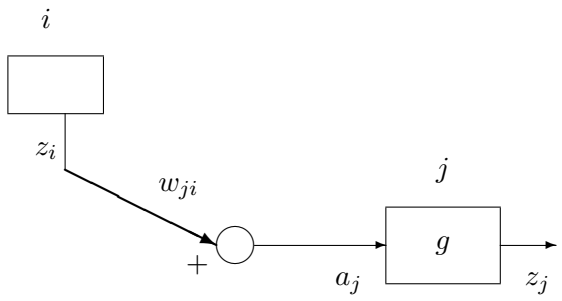
Come già si è detto, la BP batch si può porre in relazione con il metodo del gradiente e la versione batch del metodo *momentum* coincide con l'algoritmo noto nel campo dell'ottimizzazione come *heavy ball method* [87]. La backpropagation on-line si può identificare con l'algoritmo di regressione non lineare noto nel campo della Statistica come *metodo di approssimazione stocastica* [108] e la convergenza può essere stabilita, anche in un contesto deterministico, solo imponendo condizioni appropriate sul passo che deve essere necessariamente forzato a zero.

Calcolo del gradiente mediante backpropagation

Consideriamo una rete multistrato di tipo generale costituita da L strati, in cui $x \in R^n$ è il vettore di ingresso, $y \in R^K$ è il vettore di uscita e poniamo

$$z_i^{(0)} = x_i, \quad i = 1, \ldots, n \qquad z_i^{(L)} = y_i, \quad i = 1, \ldots, K.$$

Per semplificare le notazioni, omettiamo gli indici relativi ai diversi strati. Sia w_{ji} il peso di un arco entrante nel neurone j di uno strato qualsiasi. Calcoliamo il termine $\partial E_p / \partial w_{ji}$, tenendo conto del fatto che E_p dipende da w_{ji}, solo attraverso la dipendenza dall'ingresso a_j al neurone j.



Utilizzando le regole di derivazione, e tenendo conto della struttura, si può porre

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}.$$

Definendo

$$\delta_j = \frac{\partial E_p}{\partial a_j},$$

e ricordando che $a_j = \sum_h w_{jh} z_h$, dove la sommatoria è estesa ai neuroni o agli ingressi che inviano archi a j (inclusa la soglia w_{j0} con $z_0 = -1$), si ha

$$\frac{\partial E_p}{\partial w_{ji}} = \delta_j z_i.$$

Basta quindi calcolare z_i, δ_j per ottenere la derivata rispetto a w_{ji}.
La quantità z_i è l'uscita del neurone i, oppure coincide con l'ingresso $i-$mo, e quindi si può determinare, a partire dall'ingresso, applicando successivamente le trasformazioni definite dalla rete (*propagazione in avanti* degli ingressi). Rimane da calcolare la quantità δ_j, che è associata a ogni neurone ed è denominata *errore*.

Per calcolare gli *errori* associati ai neuroni, distinguiamo i due casi possibili:

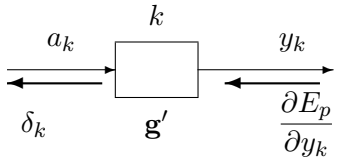
(a) il neurone appartiene allo strato d'uscita;

(b) il neurone è un neurone nascosto.

Nel caso (a), indicando con k un neurone d'uscita, si ha $y_k = g(a_k)$ e quindi si può scrivere

$$\delta_k \equiv \frac{\partial E_p}{\partial a_k} = g(a_k)' \frac{\partial E_p}{\partial y_k},$$

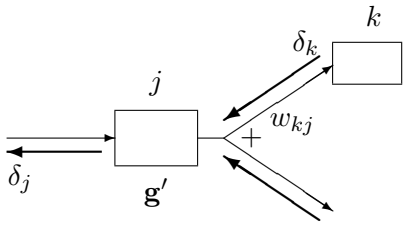
essendo $g'(a_k)$ la derivata della funzione di attivazione e $\partial E_p / \partial y_k$ calcolabile analiticamente.



Nel caso (b), indicando con j un neurone nascosto, si può porre:

$$\delta_j \equiv \frac{\partial E_p}{\partial a_j} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j},$$

dove la somma è estesa a tutti i neuroni (nascosti o d'uscita) che ricevono segnali dal neurone j. (E_p dipende da a_j solo attraverso la dipendenza da a_k).



Essendo $a_k = \ldots + w_{kj} z_j + \ldots \quad = \quad \ldots + w_{kj} g(a_j) + \ldots$ si ha:

$$\frac{\partial a_k}{\partial a_j} = g'(a_j) w_{kj},$$

e quindi si può scrivere

$$\delta_j = g'(a_j) \sum_k \delta_k w_{kj}.$$

Ne segue che gli errori δ_j relativi ai neuroni di uno strato, si ottengono propagando all'indietro lungo la rete gli errori relativi ai neuroni dello strato successivo, a partire dallo strato d'uscita (*retro-propagazione* degli errori). La tecnica di backpropagation consente di calcolare $\nabla E = \sum_p \nabla E_p$ con un costo $O(P \times W)$, dove W è il numero di parametri, P il numero di pattern. È possibile fornire formule di backpropagation anche per il calcolo dell'Hessiana e per il calcolo del prodotto Hessiana×direzione [13].
Con riferimento ad una rete con L strati, n ingressi, K uscite, e ad una funzione d'errore

$$E_p(w) = \frac{1}{2} \| y(x^p; w) - y^p \|^2,$$

la procedura di calcolo del gradiente ∇E_p, è riportata nello schema seguente, dove $N^0 = n$, e $N^{(\ell)}$, per $\ell = 1, \ldots, L$ è il numero di neuroni dello strato ℓ, con $N^{(L)} = K$.

Procedura di Backpropagation per il calcolo di $\nabla E_p(w)$

Dati. Input $x^p \in R^n$, Target $y^p \in R^K$.

Poni $z_i^{(0)} = x_i^p$, $i = 1, \ldots, n$, $z_{n+1}^{(0)} = -1$

Propagazione in avanti dell'ingresso

For $\ell = 1, \ldots, L$

For $j = 1, \ldots, N^{(\ell)}$

calcola $\quad a_j^{(\ell)} = \sum_{i=1}^{N^{(\ell-1)}+1} w_{ji}^{(\ell)} z_i^{(\ell-1)}, \qquad z_j^{(\ell)} = g_j(a_j^{(\ell)})$

End For

Poni $z_{N^{(\ell)}+1}^{(\ell)} = -1$

End For

For $i = 1, \ldots, K$

poni $e_i = z_i^{(L)} - y_i^p$

End For

Retro-propagazione dell'errore

For $j = 1, \ldots, K$

calcola $\quad \delta_j^{(L)} = e_j g_j'(a_j^{(L)})$

poni $\quad \dfrac{\partial E_p}{\partial w_{ji}^{(L)}} = \delta_j^{(L)} z_i^{(L-1)}$

End For

For $\ell = L-1, \ldots, 1$

For $j = 1, \ldots, N^{(\ell)}$

calcola $\quad \delta_j^{\ell} = g_j'(a_j^{(\ell)}) \sum_{k=1}^{N^{(\ell+1)}} \delta_k^{(\ell+1)} w_{kj}^{(\ell+1)}$

For $i = 1, \ldots, N^{(\ell-1)} + 1$

poni $\quad \dfrac{\partial E_p}{\partial w_{ji}^{(\ell)}} = \delta_j^{(\ell)} z_i^{(\ell-1)}$

End For

End For

End For

3.6 Metodi del gradiente

Come si è detto, il metodo di backpropagation si può porre in relazione con il metodo del gradiente, che è uno dei primi metodi iterativi di ottimizzazione, e può essere definito dall'iterazione

$$w^{k+1} = w^k - \eta^k \nabla E(w^k), \tag{12}$$

dove il passo η^k viene scelto in modo da soddisfare opportune condizioni. Se il passo si suppone costante a un valore fissato $\eta > 0$, la convergenza può essere garantita, imponendo limitazioni sul valore di η, nell'ipotesi che il gradiente di E soddisfi una condizione di Lipschitz, ossia che esista uno scalare $L > 0$ tale che, per ogni $w, u \in R^m$ si abbia:

$$\|\nabla E(w) - \nabla E(u)\| \leq L\|w - u\|.$$

Fissato un punto iniziale $w^0 \in R^m$ e considerato l'insieme di livello

$$\mathcal{L}_0 = \{w : E(w) \leq E(w^0)\},$$

vale il risultato seguente [12].

Teorema 5 *Supponiamo che valga una condizione di Lipschitz su R^m per ∇E e che si assuma*

$$\varepsilon \leq \eta \leq \frac{2 - \varepsilon}{L},$$

con $\varepsilon > 0$. Allora ogni punto di accumulazione della successione $\{w^k\}$ definita dallo schema iterativo

$$w^{k+1} = w^k - \eta \nabla E(w^k)$$

è un punto stazionario di E; inoltre, se $\mathcal{L}_0$ è compatto esistono punti di accumulazione di $\{w^k\}$ che sono punti stazionari di E.

In pratica, può essere difficile stimare L e quindi scegliere appropriatamente un passo costante. In alternativa, sono disponibili varie *tecniche di ricerca unidimensionale* [12] per la determinazione adattativa di η^k, ben note nel campo dei metodi di ottimizzazione, che garantiscono la convergenza e fanno migliorare notevolmente il comportamento del metodo. Tali tecniche si basano essenzialmente sulla definizione di un intervallo di valori accettabili per η^k in modo tale che siano garantiti:

- un sufficiente spostamento;

- una sufficiente riduzione della funzione obiettivo.

In particolare, uno degli algoritmi più semplici è il cosiddetto *metodo di Armijo*, in cui si assume che la direzione di ricerca d^k sia di discesa, cioè tale che $\nabla E(w^k)^T d^k < 0$.

Metodo di Armijo

Dati. $a^k > 0$, $\delta \in (0,1)$, $\gamma \in (0, 1/2)$.

Passo 1. Poni $\eta = a^k$ e $j = 0$.

Passo 2. Se è soddisfatta la condizione

$$E(w^k + \eta d^k) \leq E(w^k) + \gamma\eta\nabla E(w^k)^T d^k$$

poni $\eta^k = \eta$ e stop.

Passo 3. Poni $\eta = \delta\eta$, $j = j + 1$ e vai al Passo 2.

Il Passo 3 può essere sostituito da un procedimento di interpolazione basato su un'approssimazione quadratica della funzione.

Un potenziale svantaggio del metodo di Armijo è costituito dal fatto che la ricerca di η^k deve necessariamente iniziare a partire da una stima iniziale a^k tale che

$$a^k \geq \frac{1}{\|d^k\|}\sigma\left(\frac{\nabla E(w^k)^T d^k}{\|d^k\|}\right),$$

dove $\sigma : R^+ \to R^+$ è una funzione di forzamento. Ciò potrebbe comportare la necessità di effettuare un numero elevato di valutazioni della funzione durante la ricerca unidimensionale. Può essere quindi preferibile utilizzare criteri di accettabilità che risultino applicabili anche a partire da stime iniziali arbitrarie. Un criterio di tale tipo è costituito, ad esempio, dalle cosiddette *condizioni di Goldstein*, in base alle quali, se $\nabla E(w^k)^T d^k < 0$, si può scegliere η^k come un qualsiasi numero positivo che soddisfi le condizioni:

$$E(w^k + \eta^k d^k) \leq E(w^k) + \gamma_1\eta^k\nabla E(w^k)^T d^k$$

$$E(w^k + \eta^k d^k) \geq E(w^k) + \gamma_2\eta^k\nabla E(w^k)^T d^k,$$

in cui $0 < \gamma_1 < \gamma_2 < 1/2$.

Posto $d^k = -\nabla E(w^k)$ e utilizzando il metodo di Armijo (oppure metodi di tipo Goldstein) per il calcolo di η^k, si può dimostrare che ogni punto limite della sequenza generata dallo schema (12) è un punto stazionario. L'impiego di una tecnica di ricerca unidimensionale nel metodo del gradiente risulta notevolmente vantaggioso rispetto alla scelta di un passo costante o all'utilizzo di una tecnica euristica. Tuttavia, tutte le implementazioni tradizionali del metodo del gradiente sono in generale inefficienti e possono richiedere un numero molto elevato di iterazioni e di valutazioni della funzione e del gradiente anche per problemi di piccola dimensione.

La rapidità di convergenza del metodo del gradiente può essere migliorata modificando la direzione di ricerca con l'aggiunta di un termine dipendente dalla direzione precedente, come nel metodo *momentum*, ossia ponendo:

$$w^{k+1} = w^k - \eta \nabla E(w^k) + \beta(w^k - w^{k-1}),$$

dove $\eta > 0$ e $\beta > 0$ sono valori opportuni. Come già si è detto, tale modifica è nota nel campo dei metodi di ottimizzazione come *heavy ball method* (HB), a causa dell'analogia con il moto di un corpo pesante in un campo di forze, in presenza di attrito. Si può enunciare il seguente risultato [87] di convergenza locale.

Teorema 6 (Poljak, 1964) *Sia w^* un punto di minimo di E, tale che la matrice Hessiana sia definita positiva e abbia tutti gli autovalori in $[\ell, L]$ con $\ell > 0$. Supponiamo che*

$$0 \leq \beta < 1, \quad 0 < \eta < 2(1+\beta)/L.$$

Allora esiste $\varepsilon > 0$, tale che se w^0, w^1 stanno nella sfera $\{u : ||u - w^*|| \leq \varepsilon\}$, il metodo (HB) *converge a w^* con rapidità lineare ed esistono q, δ tali che:*

$$||w^k - w^*|| \leq C(\delta)(q+\delta)^k,$$

dove $0 \leq q < 1, \quad 0 < \delta < 1 - q$, e $C(\delta) > 0$ è una costante dipendente da δ.

Nel caso quadratico si dimostra che, con una scelta opportuna dei parametri, la rapidità di convergenza è superiore a quella del metodo del gradiente. La scelta ottima dei parametri, dal punto di vista della rapidità di convergenza è

$$\eta = \frac{4}{(\sqrt{L} + \sqrt{\ell})^2} \qquad \beta = \left(\frac{\sqrt{L} - \sqrt{\ell}}{\sqrt{L} + \sqrt{\ell}}\right)^2$$

che garantisce una limitazione superiore del rapporto fra gli errori consecutivi approssimativamente eguale a

$$q = \frac{\sqrt{L} - \sqrt{\ell}}{\sqrt{L} + \sqrt{\ell}},$$

mentre nel metodo del gradiente con ricerche esatte si ha:

$$q = \frac{L - \ell}{L + \ell}.$$

Nel caso generale, tuttavia, la convergenza globale non può essere garantita e la scelta dei parametri può essere difficile. Il metodo HB è stato utilizzato con criteri euristici per la scelta dei parametri nelle applicazioni neurali e risulta notevolmente più efficiente, in pratica, rispetto al metodo del gradiente anche per valori costanti di η e β.

3.7 Metodi delle direzioni coniugate

La motivazione alla base dell'uso dei metodi delle direzioni coniugate [52] sta nel fatto che tali metodi consentono di ottenere una rapidità di convergenza superiore rispetto a quella del metodo del gradiente, senza tuttavia richiedere *esplicitamente* la conoscenza della matrice Hessiana e senza far uso di operazioni matriciali.

I metodi delle direzioni coniugate sono stati originariamente introdotti [53] per risolvere sistemi lineari con matrice dei coefficienti simmetrica definita positiva, o equivalentemente, per risolvere problemi non vincolati di ottimizzazione quadratica convessa.

Data una matrice $Q(m \times m)$ simmetrica e definita positiva, k vettori non nulli $d_1, \ldots, d_k \in R^m$ si dicono *coniugati* rispetto a Q se risulta

$$d_i^T Q d_j = 0 \qquad i, j = 1, \ldots, k \quad i \neq j.$$

Nel caso di funzione obiettivo quadratica strettamente convessa, cioè

$$E(w) = \frac{1}{2} w^T Q w + c^T w,$$

dove la matrice Hessiana Q è simmetrica e definita positiva, il metodo del gradiente coniugato può essere descritto da un'iterazione del tipo

$$w^{k+1} = w^k + \eta^k d^k \tag{13}$$

con

$$d_k = \begin{cases} -\nabla E(w^k) & \text{per } k = 0 \\ -\nabla E(w^k) + \beta^k d^{k-1} & \text{per } k \geq 1. \end{cases} \tag{14}$$

Il passo η^k è ottenuto calcolando il minimo di $E(w^k + \eta d^k)$ rispetto ad η, per cui si ha

$$\eta^k = -\frac{\nabla E(w^k)^T d^k}{(d^k)^T Q d^k},$$

e lo scalare β^k è definito mediante la formula

$$\beta^k = \frac{\nabla E(w^k)^T Q d^{k-1}}{(d^{k-1})^T Q d^{k-1}}. \tag{15}$$

Il metodo genera direzioni mutuamente coniugate e converge alla soluzione ottima in un numero di iterazioni minore o uguale alla dimensione del problema.

Nel caso generale il metodo del gradiente coniugato è ancora definito dalle (13), (14), dove tuttavia il passo η^k viene determinato per mezzo di ricerche unidimensionali (non necessariamente finalizzate alla determinazione del minimo lungo la direzione).

Per quanto riguarda il calcolo di β^k, si utilizzano formule che non contengono esplicitamente la matrice Hessiana, e coincidono nel caso quadratico con la (15). Le formule più note sono quella di *Fletcher e Reeves* (FR)

$$\beta^k_{\mathrm{FR}} = \frac{\|\nabla E(w^k)\|^2}{\|\nabla E(w^{k-1})\|^2}$$

e quella di *Polak-Ribière* (PR)

$$\beta^k_{\mathrm{PR}} = \frac{\nabla E(w^k)^T(\nabla E(w^k) - \nabla E(w^{k-1}))}{\|\nabla E(w^{k-1})\|^2}.$$

Lo studio di proprietà di convergenza globale nel caso non quadratico di metodi di tipo gradiente coniugato costituisce un importante argomento di ricerca nell'ambito dei metodi di ottimizzazione. Una strategia semplice di stabilizzazione è quella di applicare periodicamente come direzione di ricerca la direzione dell'antigradiente (*restart*). In tal caso la convergenza globale può essere assicurata (almeno per una sottosequenza). L'esperienza di calcolo indica tuttavia che è preferibile evitare il restart lungo l'antigradiente. È stato inoltre dimostrato [2] che il metodo di Fletcher e Reeves, applicato utilizzando una tecnica di ricerca unidimensionale che consente di soddisfare le condizioni seguenti (note come condizioni di Wolfe)

$$E(w^k + \eta^k d^k) \leq E(w^k) + \gamma \eta^k \nabla E(w^k) d^k$$

$$|\nabla E(w^k + \eta^k d^k)^T d^k| < \beta |\nabla E(w^k)^T d^k|$$

con $\gamma < \beta < 1/2$, assicura la proprietà

$$\lim_{k\to\infty} \inf \|\nabla E(w^k)\| = 0.$$

In pratica, il metodo di Polak-Ribière è risultato più efficiente rispetto a quello di Fletcher-Reeves. La convergenza globale del metodo di Polak-Ribière con ricerche di linea esatte è stata dimostrata [86] nell'ipotesi di funzione obiettivo *fortemente convessa*. Nel caso generale non convesso, è stato dimostrato [38] che il metodo di Polak-Ribière *modificato* secondo la formula (introdotta originariamente da Powell)

$$\beta^k = \max\{\beta^k_{\mathrm{PR}}, 0\}$$

ha proprietà di convergenza globale. Una versione del metodo di Polak-Ribière che modifiche della formula di calcolo di β^k e presenta proprietà di convergenza globale è stata proposta in [44] ed è basata su tecniche di ricerca unidimensionale che consentono di soddisfare la condizione

$$\lim_{k\to\infty} \|w^{k+1} - w^k\| = 0.$$

È anche possibile definire delle opportune condizioni di accettabilità che consentono di stabilire proprietà di convergenza del metodo di Polak-Ribière più forti rispetto a quelle del metodo di Fletcher e Reeves, nel senso che risulta

$$\lim_{k\to\infty} \|\nabla E(w^k)\| = 0,$$

per cui ogni punto limite è un punto stazionario.

Nell'ambito più specifico dei metodi per l'addestramento di reti neurali, è stato proposto in [76] un metodo di tipo *gradiente coniugato scalato*. L'idea alla base del metodo è di modificare la matrice Hessiana $\nabla^2 E(w^k)$ aggiungendo un termine del tipo

$$\lambda^k I,$$

in cui $\lambda^k \geq 0$ è il parametro di *scaling*, e di calcolare il passo

$$\eta^k = -\frac{\nabla E(w^k)^T d^k}{(d^k)^T \nabla^2 E(w^k) d^k + \lambda^k \|d^k\|^2},$$

che rappresenta il passo ottimo nel caso di funzione quadratica strettamante convessa. Il termine $\nabla^2 E(w^k)d^k$ viene approssimato secondo la formula

$$\nabla^2 E(w^k)d^k \approx \frac{\nabla E(w^k + \sigma^k d^k) - \nabla E(w^k)}{\sigma^k},$$

dove $\sigma^k = \omega/\|d^k\|^2$, e ω è uno scalare positivo "sufficientemente piccolo". Le regole per la determinazione e per l'aggiornamento di λ^k, e il criterio per accettare il punto

$$w^k + \eta^k d^k$$

sono derivate dai metodi di tipo *trust region* [27]. Questo metodo è risultato in alcuni problemi di addestramento più efficiente rispetto a metodi tradizionali di tipo gradiente coniugato.

3.8 Metodi Quasi-Newton a memoria limitata

I *metodi Quasi-Newton* [27] costituiscono una classe di metodi per la minimizzazione non vincolata che richiedono soltanto la conoscenza delle derivate prime. Alcuni metodi di questa classe consentono di determinare il minimo di una funzione quadratica definita positiva in un numero finito di iterazioni, in quanto generano direzioni coniugate. La caratteristica più importante dei metodi Quasi-Newton appare risiedere nel fatto che essi forniscono una "approssimazione" del metodo di Newton che conserva (sotto appropriate ipotesi) una rapidità di convergenza superlineare, pur non richiedendo che venga prodotta un'approssimazione *consistente* della matrice Hessiana.

Il *metodo* BFGS (Broyden, Fletcher, Golfarb, Shanno) è uno dei più noti metodi Quasi-Newton, ed è comunemente ritenuto uno tra i metodi più efficienti di ottimizzazione non vincolata per problemi di dimensione non elevata. Il metodo BFGS può essere descritto da una iterazione della forma

$$w^{k+1} = w^k - \eta^k H^k \nabla E(w^k),$$

dove η^k è il passo determinato con una tecnica di ricerca unidimensionale in modo da soddisfare condizioni di Wolfe e la matrice H^k soddisfa l'equazione

$$H^k y^k = s^k,$$

con

$$s^k = x^{k+1} - x^k$$
$$y^k = \nabla E(w^{k+1}) - \nabla E(w^k).$$

La matrice H^k costituisce un'approssimazione dell'inversa della matrice Hessiana $\nabla^2 E(w^k)$ ed è aggiornata utilizzando la coppia $\{s^k, y^k\}$ secondo la formula

$$H^{k+1} = (V^k)^T H^k V^k + \rho^k s^k (s^k)^T, \tag{16}$$

dove

$$\rho^k = \frac{1}{(y^k)^T s^k}, \qquad V^k = I - \rho^k y^k (s^k)^T. \tag{17}$$

La matrice H^k è in generale densa, per cui la sua memorizzazione ed il costo computazionale di operazioni ad essa associate possono rendere proibitivo il suo impiego quando il numero di variabili è sufficientemente elevato.

Per ovviare a questo limite sono stati proposti i *metodi Quasi-Newton a memoria limitata* [37], [64], [78] in cui una versione *modificata* di H^k è memorizzata implicitamente utilizzando un numero fissato m di coppie $\{s^i, y^i\}$ nelle formule (16), (17). In particolare, il prodotto $H^k \nabla E(w^k)$ può essere ottenuto eseguendo una sequenza di prodotti scalari e di somme vettoriali in cui compaiono $\nabla E(w^k)$ e le coppie $\{s^i, y^i\}$. Una volta aggiornato il vettore corrente w^k, la coppia "più vecchia" $\{s^i, y^i\}$ è rimossa e sostituita con la nuova coppia $\{s^k, y^k\}$. In tal modo, l'insieme di coppie contiene informazioni relative alle ultime m iterazioni eseguite. L'esperienza pratica ha mostrato che si ottengono risultati soddisfacenti con valori relativamente piccoli (compresi tra 2 e 20) della *memoria* m.

Il calcolo del prodotto $H^k \nabla E(w^k)$ può essere effettuato in maniera efficiente mediante una procedura ricorsiva. Prima di descrivere formalmente la procedura, riportiamo di seguito una breve giustificazione.

Sia w^k il punto corrente, sia $\{s^i, y^i\}$, $i = k-m, \ldots, k-1$, l'insieme delle coppie di vettori memorizzate, e sia H_0^k la matrice iniziale selezionata. Applicando ripetutamente la (16) a partire dalla matrice H_0^k scelta si ottiene

$$\begin{aligned} H^k = \quad & [(V^{k-1})^T \ldots (V^{k-m})^T] H_0^k [(V^{k-m}) \ldots (V^{k-1})] \\ & + \rho^{k-m} [(V^{k-1})^T \ldots (V^{k-m+1})^T] s^{k-m} (s^{k-m})^T [(V^{k-m+1}) \ldots (V^{k-1})] \\ & + \rho^{k-m+1} [(V^{k-1})^T \ldots (V^{k-m+2})^T] s^{k-m+1} (s^{k-m+1})^T [(V^{k-m+2}) \ldots (V^{k-1})] \\ & + \ldots \\ & + \rho^{k-1} s^{k-1} (s^{k-1})^T. \end{aligned}$$

Si ponga $q^k = \nabla E(w^k)$ e si definiscano, per $i = k-1, \ldots, k-m$, i vettori

$$q^i = V^i \ldots V^{k-1} \nabla E(w^k).$$

Risulta perciò

$$q^i = V^i q^{i+1} = q^{i+1} - \rho^i y^i (s^i)^T q^{i+1},$$

da cui, ponendo $\alpha^i = \rho^i (s^i)^T q^{i+1}$, si ottiene

$$q^i = q^{i+1} - \alpha^i y^i.$$

Utilizzando i vettori q^i e gli scalari α^i si può scrivere

$$\begin{aligned} H^k \nabla E(w^k) = \quad & [(V^{k-1})^T \quad \ldots \quad (V^{k-m+2})^T \ (V^{k-m+1})^T \ (V^{k-m})^T] H_0^k q^{k-m} \\ & + [(V^{k-1})^T \quad \ldots \quad (V^{k-m+2})^T \ (V^{k-m+1})^T] \alpha^{k-m} s^{k-m} \\ & + [(V^{k-1})^T \quad \ldots \quad (V^{k-m+2})^T] \alpha^{k-m+1} s^{k-m+1} \\ & + \ldots \\ & + \alpha^{k-1} s^{k-1}. \end{aligned}$$

Si ponga ora $r^{k-m-1} = H_0^k q^{k-m}$ e si definiscano i seguenti vettori r^i per $i = k-m, \ldots, k-1$

$$r^i = (V^i)^T r^{i-1} + \alpha^i s^i.$$

Risulta perciò

$$r^i = r^{i-1} + \rho^i (y^i)^T r^{i-1} s^i + \alpha^i s^i,$$

da cui ponendo $\beta^i = \rho^i (y^i)^T r^{i-1}$ si ottiene

$$r^i = r^{i-1} + (\alpha^i - \beta^i) s^i.$$

Dall'espressione di $H^k \nabla E(w^k)$ e dalla definizione dei vettori r^i si ha $H^k \nabla E(w^k) = r^{k-1}$.

Possiamo quindi riportare la procedura di calcolo di $H^k \nabla E(w^k)$.

Procedura (HG)

Poni $q^k = \nabla E(w^k)$;

For $i = k-1, k-2, \ldots, k-m$

poni $\alpha^i = \rho^i (s^i)^T q^{i+1}$

poni $q^i = q^{i+1} - \alpha^i y^i$

End For

Poni $r^{k-m-1} = H_0^k q^{k-m}$

For $i = k-m, k-m+1, \ldots, k-1$

poni $\beta^i = \rho^i (y^i)^T r^{i-1}$

poni $r^i = r^{i-1} + s^i(\alpha^i - \beta^i)$

End For

Poni $H^k \nabla E(w^k) = r^{k-1}$ e stop.

L'algoritmo BFGS a memoria limitata può essere riassunto nello schema seguente, dove si assumono assegnati il punto iniziale w^0 e l'intero m.

Algoritmo L-BFGS

Per $k = 0, \ldots$

scegli H_0^k;

calcola $d^k = -H^k \nabla E(w^k)$ utilizzando la Procedura HG;

poni $w^{k+1} = w^k + \eta^k d^k$ dove η^k è calcolato in modo tale che le condizioni di Wolfe risultino soddisfatte;

Se $k > m$ elimina la coppia $\{s^{k-m}, y^{k-m}\}$ dalla memoria;

Calcola e memorizza s^k e y^k;

Senza considerare la moltiplicazione $H_0^k q^{k-m}$, la procedura HG richiede $4mn$ moltiplicazioni. Si osservi inoltre che H_0^k può essere scelta senza particolari vincoli e può variare da una iterazione all'altra. Una scelta che si è rivelata efficiente in pratica è quella di porre $H_0^k = \gamma^k I$, con

$$\gamma^k = \frac{(s^{k-1})^T y^{k-1}}{(y^{k-1})^T y^{k-1}}.$$

L'algoritmo L-BFGS può essere vantaggiosamente impiegato nei problemi a grande dimensione in cui la matrice Hessiana non è sparsa e può essere considerato uno dei metodi più efficienti per problemi di addestramento di reti multistrato. In tale contesto, uno studio sperimentale dell'algoritmo L-BFGS è riportato in [7], in cui il parametro m è stato posto uguale a 1.

3.9 Metodi di tipo Gauss-Newton

Nel problema di addestramento la misura di errore usualmente adottata è l'errore quadratico, per cui la funzione obiettivo assume la forma

$$E(w) = \sum_{p=1}^{P} E_p = \frac{1}{2}\sum_{p=1}^{P} e_p^2 = \frac{1}{2}\sum_{p=1}^{P} \|y(x^p; w) - y^p\|^2,$$

da cui segue che il problema è un *problema ai minimi quadrati*. Definiamo il *vettore residuo* $e : R^m \to R^P$ ponendo

$$e(w) = \begin{pmatrix} e_1(w) \\ e_2(w) \\ \vdots \\ e_P(w) \end{pmatrix}.$$

Usando tale notazione, il problema assume la forma

$$\min_{w \in R^m} E(w) = \frac{1}{2}\|e(w)\|^2. \tag{18}$$

Le derivate di E possono essere espresse in termini della matrice Jacobiana J $(P \times m)$

$$J(w) = \begin{pmatrix} \nabla e_1(w)^T \\ \nabla e_2(w)^T \\ \vdots \\ \nabla e_P(w)^T \end{pmatrix}.$$

In particolare, il gradiente risulta definito secondo la formula

$$\nabla E(w) = \sum_{p=1}^{P} e_p(w) \nabla e_p(w) = J(w)^T e(w),$$

la matrice Hessiana assume la forma

$$\begin{aligned}\nabla^2 E(w) &= \sum_{p=1}^{P} \nabla e_p(w) \nabla e_p(w)^T + \sum_{p=1}^{P} e_p(w) \nabla^2 e_p(w) \\ &= J(w)^T J(w) + \sum_{p=1}^{P} e_p(w) \nabla^2 e_p(w) = J(w)^T J(w) + Q(w).\end{aligned}$$

La matrice Hessiana è data quindi dalla somma di due termini, $J(w)^T J(w)$ e $\sum_{p=1}^{P} e_p(w) \nabla^2 e_p(w)$. I metodi ai minimi quadrati sono basati, in generale, sull'assunzione che il primo di tali termini, cioè $J(w)^T J(w)$, è *dominante* rispetto all'altro. In particolare, nei problemi in cui i residui e_p sono sufficientemente "piccoli" in un intorno di una soluzione, tale assunzione appare giustificata.

Il *metodo di Gauss-Newton* rappresenta il metodo più semplice per problemi ai minimi quadrati, e si basa sulla regola di aggiornamento

$$w^{k+1} = w^k + d_{gn}^k,$$

dove d_{gn}^k è soluzione del sistema

$$(J^k)^T J^k d = -(J^k)^T e^k. \tag{19}$$

Questo metodo può essere posto in relazione con il metodo di Newton. La direzione di ricerca d^k che caratterizza il metodo di Newton è ottenuta risolvendo il sistema

$$\nabla^2 E(w^k) d = -\nabla E(w^k). \tag{20}$$

Osservando che $\nabla^2 E(w^k) = (J^k)^T J^k + Q^k$, si deduce immediatamente che il sistema (19) deriva dal sistema (20) escludendo il secondo dei termini che forniscono la matrice Hessiana $\nabla^2 E(w^k)$. I potenziali vantaggi del metodo sono i seguenti:

- l'approssimazione $\nabla^2 E^k \approx (J^k)^T J^k$ consente di evitare il calcolo delle singole matrici Hessiane $\nabla^2 e_p$ per $p = 1, \ldots, P$;

- in molte situazioni il termine $J^T J$ è effettivamente dominante, per cui l'efficienza del metodo è paragonabile a quella del metodo di Newton anche se il secondo termine $\sum_{p=1}^{P} e_p \nabla^2 e_p$ non è considerato nell'approssimazione della matrice Hessiana;

- la direzione d_{gn}^k è la soluzione del *problema ai minimi quadrati lineari*

  $$\min_{d \in R^n} \|J^k d + e^k\|^2,$$

  per cui possono essere utilizzati vari algoritmi efficienti per il calcolo di d_{gn}^k; in particolare, nel caso di problemi a grande dimensione appare naturale l'impiego di metodi iterativi (gradiente coniugato, Barzilai-Borwein) con opportuni *criteri di troncamento*.

Per quanto riguarda le proprietà di convergenza locali del metodo di Gauss-Newton, si hanno i seguenti risultati [27].

Teorema 7 *Sia $E(w)$ due volte continuamente differenziabile su un insieme aperto convesso $D \subseteq R^m$. Si supponga che valgano le seguenti condizioni:*

(i) *esiste un $w^* \in D$ tale che $J(w^*)^T e(w^*) = 0$;*

(ii) *la matrice Jacobiana $J(w)$ è Lipschitz continua su D, cioè esiste una costante $L > 0$ tale che*

$$\|J(w) - J(z)\| \leq L\|w - z\| \qquad \text{per ogni } w, z \in D;$$

(iii) *esiste un $\sigma \geq 0$ tale che*

$$\| \left(J(w) - J(w^*)\right)^T e(w^*)\| \leq \sigma\|w - w^*\| \qquad \text{per ogni } w \in D.$$

Sia $\lambda \geq 0$ il più piccolo autovalore di $J(w^*)^T J(w^*)$. Se $\lambda > \sigma$, allora per ogni $c \in (1, \lambda/\sigma)$ esiste una sfera aperta $B(w^*; \epsilon)$ tale che per ogni $w^0 \in B(w^*; \epsilon)$ la sequenza generata con il metodo di Gauss-Newton

$$w^{k+1} = w^k - \left(J(w^k)^T J(w^k)\right)^{-1} J(w^k)^T e(w^k)$$

è ben definita, rimane in $B(w^*; \epsilon)$, converge a w^*, e si ha

$$\|w^{k+1} - w^*\| \leq \frac{c\sigma}{\lambda}\|w^k - w^*\| + \frac{c\alpha L}{2\lambda}\|w^k - w^*\|^2,$$

dove $\alpha > 0$ è tale che $\|J(w)\| \leq \alpha$ per ogni $w \in D$, e

$$\|w^{k+1} - w^*\| \leq \frac{c\sigma + \lambda}{2\lambda}\|w^k - w^*\| < \|w^k - w^\star\|.$$

Nell'ipotesi in cui il punto di minimo $w^\star$ abbia residui nulli cioè tali che $e(w^\star) = 0$, dal Teorema 7 si ottiene il seguente risultato di *convergenza quadratica* del metodo di Gauss-Newton.

Teorema 8 *Si supponga che le ipotesi del Teorema 7 siano verificate e che $e(w^*) = 0$. Allora esiste una sfera aperta $B(w^*; \epsilon)$ tale che se $w^0 \in B(w^*; \epsilon)$ la sequenza generata dal metodo di Gauss-Newton è ben definita, rimane in $B(w^*; \epsilon)$ e converge a w^* con rapidità di convergenza quadratica.*

Per assicurare proprietà di convergenza globale il metodo di Gauss-Newton è spesso implementato nella forma modificata

$$w^{k*1} = w^k - \eta^k \left(J(w^k)^T J(w^k) + D^k \right)^{-1} J(w^k)^T e(w^k),$$

dove η^k è il passo scelto con una opportuna ricerca unidimensionale (di tipo Armijo ad esempio) e D^k è una matrice diagonale tale che la matrice

$$J(w^k)^T J(w^k) + D^k$$

risulti *uniformemente definita positiva*. Ad esempio, D^k può essere determinata con una procedura di fattorizzazione di *Cholesky modificata*.

Un'alternativa basata su tecniche di tipo *trust region* [27] è quella di scegliere come matrice D^k un multiplo positivo della matrice identità. In questo secondo caso, il metodo è noto come il *metodo di Levenberg-Marquardt*, la cui k-esima iterazione è quindi definita da

$$w^{k+1} = w^k - \left(J(w^k)^T J(w^k) + \mu^k I \right)^{-1} J(w^k)^T e(w^k),$$

dove $\mu^k \geq 0$ è uno scalare opportunamente scelto. Le proprietà di convergenza locali del metodo di Levenberg-Marquardt sono simili a quelle del metodo di Gauss-Newton. In letteratura sono state proposte molte versioni del metodo di Levenberg- Marquardt, e differiscono nella regola di aggiornamento dello scalare μ^k. Una versione *inesatta* del metodo basata su una opportuna regola di troncamento nella procedura di calcolo della direzione ed adatta per la soluzione di problemi di elevata dimensione è stata proposta in [26].

Osserviamo che, analogamente al caso del metodo di Newton, anche per il metodo di Gauss-Newton appare vantaggioso l'impiego di tecniche di stabilizzazione non monotone per la soluzione di problemi "fortemente non lineari". Nella letteratura di ottimizzazione metodi tipo Gauss-Newton non monotoni sono stati definiti in [29] e [60], mentre una versione non monotona del metodo di Levenberg-Marquardt è stata proposta in [112].

In [50] il metodo di Levenberg-Marquardt è stato confrontato con altri metodi (backpropagation, gradiente coniugato) su problemi di addestramento di reti multistrato. I risultati ottenuti mostrano che il metodo può essere vantaggiosamente applicato alla soluzione di problemi di dimensione relativamente elevata (cioè, in problemi in cui il numero di pesi da determinare è dell'ordine del centinaio). Il calcolo della matrice Jacobiana può essere effettuato mediante tecniche di backpropagation [13]. Recentemente in [3] sono stati proposte varianti del metodo di Levenberg-Marquardt che includono un termine aggiuntivo di tipo momentum derivante da una formulazione vincolata del problema di addestramento.

3.10 Il metodo di Barzilai-Borwein

Una versione [6] del metodo del gradiente, nota come *metodo del gradiente di Barzilai-Borwein* (BB), appare particolarmente promettente nei problemi di addestramento "difficili". Il metodo BB è un metodo di tipo gradiente descritto dallo schema iterativo

$$w^{k+1} = w^k - \frac{1}{\alpha^k}\nabla E(w^k),$$

dove lo scalare α è definito mediante una delle due formule

$$\alpha_1 = \frac{s^T y}{s^T s} \qquad \alpha_2 = \frac{y^T y}{s^T y} \tag{21}$$

con

$$s = w^k - w^{k-1}, \quad y = \nabla E(w^k) - \nabla E(w^{k-1}).$$

Le scelte di α sono connesse all' *equazione Quasi-Newton*

$$Bs = y$$

dove B è una matrice $m \times m$ simmetrica definita positiva che approssima la matrice Hessiana $\nabla^2 E(w^k)$. Infatti, gli scalari α_1 e α_2 minimizzano, rispettivamente, gli errori $\|\alpha I s - y\|$, e $\|s - (1/\alpha) I y\|$, sull'equazione Quasi-Newton qualora si assuma $B = \alpha I$.

Nel caso quadratico strettamente convesso il metodo BB può essere interpretato come un metodo del gradiente in cui il passo è dato dall'inverso di un'approssimazione di un autovalore della matrice Hessiana, e per tale motivo è anche denominato in letteratura come *metodo spettrale del gradiente*. Nel caso ideale, cioè utilizzando in sequenza gli autovalori esatti per definire i corrispondenti passi lungo l'antigradiente, si determinerebbe in un numero finito di iterazioni il punto di minimo di una funzione quadratica strettamente convessa. Si comprende perciò come il metodo BB presenti potenzialmente buone proprietà locali. La convergenza del metodo BB è stata stabilita nel caso di obiettivo quadratico strettamente convesso [90]. Tuttavia, nel caso generale, il metodo BB può non convergere. Un metodo di stabilizzazione globale (*GBB method*) [91] è definito da

$$w^{k+1} = w^k - \eta^k \frac{1}{\alpha^k}\nabla E(w^k)$$

dove $0 < \eta^k \leq 1$ è calcolato per mezzo di una line search *non monotona* di tipo Armijo (introdotta originariamente per stabilizzare il metodo di Newton) [41] e basata sulla condizione di accettabilità

$$E(w^k + \eta d^k) \leq \max_{0 \leq j \leq \min(k,M)} \left\{ E(w^{k-j}) \right\} + \gamma \eta \nabla E(w^k)^T d^k, \tag{22}$$

dove $d^k = -(1/\alpha^k)\nabla E(w^k)$ e α^k è il passo BB (eventualmente modificato).

Il metodo GBB consente di migliorare notevolmente il comportamento del metodo del gradiente ed è competitivo con il metodo del gradiente coniugato anche nei problemi convessi a grande dimensione. Tuttavia esso può richiedere un costo computazionale elevato in problemi difficili, in cui può essere necessario un ulteriore rilassamento della non monotonicità [31].

In [83] è stata considerata una classe di metodi non monotoni di tipo gradiente per l'addestramento di reti multistrato che comprende anche il metodo BB. La strategia di stabilizzazione di questi metodi è simile a quella impiegata nel metodo GBB, e si basa quindi sulla condizione di accettabilità (22). I risultati sperimentali riportati mostrano i vantaggi che scaturiscono dall'utilizzo di tecniche non monotone per l'addestramento di reti neurali.

3.11 Algoritmo non monotono di stabilizzazione

Una nuova tecnica di stabilizzazione non monotona del metodo BB, che appare particolarmente promettente per l'addestramento di reti multistrato, è stata definita recentemente [48] ed è basata sulla combinazione di una tecnica di tipo *watchdog* non monotona con una tecnica di ricerca unidimensionale non monotona che consente anche incrementi del passo. Lo schema di globalizzazione proposto (NMS) è applicabile, in linea di principio, ad un metodo qualsiasi, allo scopo di assicurare la convergenza globale verso punti stazionari di un algoritmo *locale*, perturbando il meno possibile i punti di tentativo generati localmente. Tale schema si può descrivere definendo:

una sequenza di iterazioni principali $k = 0, 1, \ldots$, che producono i punti w^k;

per ogni k, una sequenza di iterazioni interne che generano, a partire da w^k, attraverso un algoritmo *locale*, una sequenza finita di N punti: $z_1^k, z_2^k, \ldots, z_N^k$;

un criterio di tipo *watchdog* per accettare o meno il punto z_N^k;

un algoritmo non monotono di ricerca unidimensionale che determina uno spostamento η^k lungo una direzione di discesa opportuna nei casi in cui il punto z_N^k non sia accettato.

Si suppone che per ogni k sia generata una direzione di discesa d^k tale che

$$\|d^k\| \leq c_1 \|\nabla E(w^k)\| \tag{23}$$

$$\nabla E(w^k)^T d^k \leq -c_2 \|\nabla E(w^k)\|^2, \tag{24}$$

per $c_1, c_2 > 0$. La condizione precedente è soddisfatta assumendo, ad esempio,

$$d^k = -(1/\alpha^k)\nabla E(w^k),$$

con

$$0 < \varepsilon \leq \alpha^k \leq 1/\varepsilon.$$

L'algoritmo di stabilizzazione è descritto dallo schema seguente.

Algoritmo NMS

Dati. $w^0 \in R^m$, interi $N \geq 1$, $M \geq 0$, $k = 0$, $\sigma : R^+ \to R^+$ funzione di forzamento.

While $\nabla E(w^k) \neq 0$ **do**

Passo 1. Calcola una direzione d^k che soddisfi le condizioni (23), (24) e poni

$$z_1^k = w^k + d^k$$

Passo 2. Per $i = 2, N$ determina una direzione p_i^k e poni

$$z_i^k = z_{i-1}^k + p_i^k$$

Passo 3. **If** $E(z_N^k) \leq \max\limits_{0 \leq j \leq \min(k,M)} \{E(w^{k-j})\} - \max\limits_{1 \leq i \leq N} \left\{\sigma\left(\|z_i^k - w^k\|\right)\right\}$ **then**

poni $w^{k+1} = z_N^k$

else

determina η^k con un algoritmo di ricerca unidimensionale e poni

$$w^{k+1} = w^k + \eta^k d^k$$

end if

Passo 4. Poni $k = k + 1$.

end while

La ricerca unidimensionale deve essere tale da soddisfare la seguente condizione.

Condizione sulla ricerca unidimensionale (LS)

Sia K un sottoinsieme infinito di indici tale che $w^{k+1} = w^k + \eta^k d^k,\ k \in K$. Allora:

(c1) *Per ogni $k \in K$ risulta:*

$$E(w^k + \eta^k d^k) \leq \max_{0 \leq j \leq \min(k,M)} \{E(w^{k-j})\} - \sigma(\eta^k \|d^k\|),$$

dove $M \geq 0$ è un intero prefissato, e $\sigma : R^+ \to R^+$ è una funzione di forzamento;

(c2) *Se $\{E(w^k)\}$ converge e $\{w^k\}_K$ è limitata, si ha*

$$\lim_{k \to \infty, k \in K} \frac{\nabla E(w^k)^T d^k}{\|d^k\|} = 0.$$

La condizione (LS) può essere soddisfatta attraverso una tecnica di ricerca unidimensionale non monotona di tipo Armijo [41]. In [48] è stato definito un nuovo algoritmo di ricerca unidimensionale non monotona che ammette anche eventuali incrementi del passo di primo tentativo $\eta = 1$ e permette di assicurare che la condizione (LS) risulti soddisfatta.

Per quanto riguarda le proprietà di convergenza dell'Algoritmo NMS, si può dimostrare il seguente risultato.

Teorema 9 *Supponiamo che l'insieme di livello $\mathcal{L}_0$ sia compatto, che $\{w^k\}$ sia una successione prodotta dall'algoritmo* **NMS** *e che la condizione* **LS** *sia soddisfatta. Allora:*

(i) $\{w^k\} \subset \mathcal{L}_0$ *ed ha punti limite;*

(ii) *ogni punto limite della successione $\{w^k\}$ è un punto stazionario di E in $\mathcal{L}_0$, e non è un punto di massimo locale.*

Come detto in precedenza, l'Algoritmo NMS può essere visto come uno schema non monotono di globalizzazione di metodi di ottimizzazione con proprietà locali. La caratteristica principale dell'algoritmo è di non modificare il metodo locale durante una sequenza finita di passi, al fine di trarre vantaggio dalle "buone" proprietà locali del metodo.

In [48] è stata definita una versione del metodo BB basata sull'Algoritmo NMS, in cui le direzioni p_i^k, utilizzate in ciascun ciclo principale, sono determinate secondo lo schema

$$p_i^k = -\frac{1}{\alpha_i^k} \nabla f(z_i^k).$$

Gli scalari α_i^k sono definiti alternando, quando possibile, i valori α_1 e α_2 forniti dalle formule riportate nella (21). Il metodo è stato sperimentato su un ampio insieme di problemi test di ottimizzazione, con dimensioni variabili da 100 a 10000. I risultati ottenuti mostrano che la versione proposta del metodo BB è competitiva con metodi più sofisticati di tipo Quasi-Newton.

Per quanto riguarda le reti neurali, si ritiene che la tecnica di stabilizzazione descritta possa costituire un valido strumento per la definizione di algoritmi di addestramento (sia di tipo batch che on-line) efficienti e con proprietà di convergenza globale. In particolare, nell'ambito dei metodi batch, può essere di interesse lo studio di metodi di tipo BB con momentum, stabilizzati mediante l'Algoritmo NMS.

3.12 Metodi *on-line*

I metodi di tipo *on-line* [96] consistono nell'aggiornare iterativamente il vettore w in corrispondenza a ciascun singolo termine E_p della funzione di errore, senza prima formare la funzione complessiva E, come invece è richiesto nei metodi batch. Nei problemi di apprendimento in tempo reale (*on-line learning*) i metodi di addestramento *on-line* sono gli unici metodi utilizzabili; in tal caso le proprietà di convergenza possono essere caratterizzate solo in termini probabilistici. Tuttavia, anche nell'addestramento fuori linea, i metodi *on-line* sono spesso preferiti ai metodi batch e le motivazioni principali possono essere così riassunte:

- a (grande) distanza da una soluzione può non essere conveniente impiegare molto tempo a calcolare tutta la funzione e le sue derivate, ma è preferibile ridurre rapidamente un qualsiasi termine di errore;

- se l'insieme di addestramento T ha una cardinalità molto elevata il calcolo di E e ∇E può essere impraticabile;

- se T è grande e ridondante può essere inutilmente costoso tener conto ad ogni passo di tutti i termini della funzione di errore;

- la tecnica on-line introduce un certo grado di 'randomicità' che può essere utile per evitare la convergenza verso minimi locali indesiderati.

Da un punto di vista deterministico, i metodi *on-line* possono esssere visti come metodi *incrementali* in cui si effettua una decomposizione della funzione obiettivo E o del gradiente ∇E e la valutazione delle diverse componenti è distribuita in una sequenza di P iterazioni (denominata *epoca*). Per poter stabilire risultati di convergenza è tuttavia necessario generare una successione (in genere infinita) di *epoche*, ossia prendere ripetutamente in considerazione, come nei metodi batch, tutti i termini della funzione di errore e delle sue derivate.

Uno dei primi metodi incrementali proposti è l'algoritmo di addestramento del perceptron illustrato in precedenza, o, più in generale, il metodo di rilassamento per la soluzione dei sistemi di disequazioni lineari. Per problemi di minimi quadrati lineari, il metodo incrementale più significativo è il cosiddetto *filtro di Kalman*, che ha importanti applicazioni nell'ambito dei sistemi di controllo e di comunicazione, e consente di aggiornare in modo esatto la soluzione di un problema di minimi quadrati quando si aggiunga un nuovo termine nella funzione obiettivo [10]. Per l'addestramento di reti multistrato il metodo più noto è la versione on-line del metodo di backpropagation (nota anche come *metodo del gradiente incrementale*), che si basa, come si è già detto, su un'iterazione del tipo

$$w^{k+1} = w^k - \eta^k \nabla E_{p(k)}(w^k).$$

Dal punto di vista teorico, se si suppone che tutti i termini E_p siano considerati ciclicamente per $p = 1, \ldots, P$, ossia se

$$p(k) = k(\mathrm{mod}P) + 1,$$

è possibile stabilire sotto opportune condizioni la convergenza della backpropagation on-line, che è stata studiata sia in termini probabilistici che in termini deterministici ([33], [66], [70], [102],[108]). Si può enunciare, in particolare, il risultato seguente [9].

Teorema 10 (Convergenza del metodo del gradiente incrementale)
Sia $\{w^k\}$ una successione prodotta dal metodo del gradiente incrementale. Supponiamo che esistano costanti positive L, C e D tali che:

(i) $\|\nabla E_p(u) - \nabla E_p(v)\| \leq L\|u - v\|$ *per ogni* $u, v \in R^m$ *e* $p = 1, \ldots P$;

(ii) $\|\nabla E_p(w)\| \leq C + D\|\nabla E(w)\|$ *per ogni* $w \in R^m$ *e* $p = 1, \ldots P$.

Supponiamo inoltre che il passo η^k sia scelto in modo tale da soddisfare le condizioni

$$\sum_{k=0}^{\infty} \eta^k = \infty, \qquad \sum_{k=0}^{\infty} (\eta^k)^2 < \infty. \tag{25}$$

Allora ogni punto limite di $\{w^k\}$ è un punto stazionario di E.

La condizione del Teorema 10 richiede, in essenza, che il passo η^k sia forzato a zero non troppo rapidamente. In particolare, se si assume (come nel metodo di approssimazione stocastica) che sia

$$\eta^k = c/k,$$

con $c > 0$, le condizioni (25) sono soddisfatte. Risultati di convergenza analoghi si possono stabilire anche per la versione on-line del metodo *momentum*. [70].

La principale limitazione del metodo BP on-line consiste proprio nel fatto che, per assicurare la convergenza senza valutare la funzione obiettivo complessiva, occorre imporre a priori che il passo tenda a zero con una legge prefissata e ciò porta, in pratica, ad una rapidità di convergenza (tipicamente sublineare) molto inferiore rispetto a quella delle versioni batch del metodo del gradiente. Una difficoltà ulteriore è costituita dal requisito che la successione generata si mantenga in un insieme limitato. Ciò ha motivato lo studio di varie tecniche euristiche per la scelta del passo η^k [21], [51], [99]. Nell'addestramento fuori linea un possibile compromesso è l'impiego di tecniche ibride on-line-batch chiamate *bold-driver methods*, in cui la backpropagation on-line BP è utilizzata per una o più epoche ed il passo è ricalcolato periodicamente valutando la funzione di errore complessiva E [106]. Per alcune di tali versioni sono stati stabiliti risultati di convergenza globale [43], [102].

Queste tecniche possono far migliorare, in pratica il comportamento dei metodi *on-line*, ma tuttavia presentano lo svantaggio di richiedere il calcolo di tutta la funzione di errore, il che può essere inaccettabile se P è molto grande o se l'addestramento deve essere effettuato in tempo reale. Un altro tipo di compromesso, con analoghe limitazioni, è quello di passare gradualmente dalla BP on-line, che è particolarmente vantaggiosa nelle iterazioni iniziali, alla BP batch, che assicura una rapidità di convergenza più elevata [11].

Una diversa definizione di tecniche on-line può essere basata sulla trasformazione del problema di addestramento in un problema equivalente vincolato, che può essere riformulato a sua volta, attraverso l'impiego di funzioni Lagrangiane aumentate, come un problema di ottimizzazione non vincolata e risolto con tecniche di decomposizione. Osserviamo che il problema originario si può porre nella forma:

$$\begin{aligned} \min \quad & \sum_{p=1}^{P} E_p(v_p) \\ \text{con i vincoli} \quad & \\ & v_p - u = 0, \quad p = 1, \ldots, P \\ & u \in R^n, v_p \in R^m, \ p = 1, \ldots, P, \end{aligned}$$

dove u and v_p, per $p = 1, \ldots, P$ sono $P + 1$ copie del vettore w.

È possibile ora definire un nuovo problema non vincolato, tale che, quando u è fissato, ci si può ricondurre a P sottoproblemi indipendenti. A tale scopo, introduciamo un vettore di *moltiplicatori* λ con componenti $\lambda_p \in R^m$ e definiamo la funzione Lagrangiana

$$L(v, \lambda, u) := \sum_{p=1}^{P} E_p(v_p) + \sum_{p=1}^{P} \lambda_p^T (v_p - u).$$

Possiamo quindi definire una funzione *Lagrangiana aumentata esatta* [28] ponendo:

$$\Phi(v, \lambda, u; c) := \sum_{p=1}^{P} \left[E_p(v_p) + \pi_p(v_p, \lambda_p, u; c_p) \right],$$

dove

$$\pi_p := \quad \lambda_p^T (v_p - u) + \left[c_p + \tau \|\lambda_p\|^2 \right] \|v_p - u\|^2 + \eta \|\nabla E_p(v_p) + \lambda_p\|^2,$$

e c_p, τ, η sono parametri positivi. Ponendo:

$$\phi_p(v_p, \lambda_p, u; c_p) := E_p(v_p) + \pi_p(v_p, \lambda_p, u; c_p),$$

si ottiene il problema non vincolato

$$\min_{(v, \lambda, u)} \quad \Phi(v, \lambda, u; c) = \sum_{p=1}^{P} \phi_p(v_p, \lambda_p, u; c_p).$$

Se u è fissato, la minimizzazione rispetto a (v, λ) si può decomporre in P sottoproblemi indipendenti, ciascuno corrispondente alla funzione obiettivo

$$\phi_p(v_p, \lambda_p, u; c_p)$$

che richiede solo la conoscenza di E_p. Assegnati i vettori (v_p, λ_p), la minimizzazione rispetto a u richiede solo la considerazione della funzione

$$\Pi := \sum_{p=1}^{P} \lambda_p^T (v_p - u) + (c_p + \tau \|\lambda_p\|^2) \|v_p - u\|^2,$$

che non dipende dai dati del problema di training e si può interpretare come una misura del "disagreement" tra i diversi addestramenti 'locali' relativi alle P diverse "copie" della rete. Le proprietà di Φ e la convergenza di schemi di decomposizione sono state studiate in un lavoro recente [47]. In particolare, si dimostra che esistono valori finiti dei coefficienti di penalità c_p, che assicurano una corrispondenza uno-a-uno, in un insieme di livello compatto, tra punti stazionari e punti di minimo globale di Φ con punti stazionari e punti di minimo globale di E. Si possono definire algoritmi diversi, in relazione al criterio usato per sequenziare e terminare i processi di addestramento locale e agli algoritmi usati per le minimizzazioni locali e per l'aggiornamento di u e dei coefficienti di penalità. La costruzione di algoritmi convergenti si può basare sui risultati relativi alla convergenza dei metodi di ottimizzazione non vincolata basati su tecniche di decomposizione a blocchi rispetto alle variabili [45].

3.13 Impiego di metodi di ottimizzazione globale

La presenza di minimi locali della funzione d'errore costituisce una difficoltà di cui tenere conto nel progetto di un algoritmo di ottimizzazione per l'addestramento di reti multistrato. Lo studio di proprietà teoriche della funzione d'errore in relazione ai minimi locali è stato oggetto di vari lavori. Ad esempio, è stata provata la presenza di minimi locali in alcuni specifici problemi test [63]. Viceversa, in [39] è stato dimostrato che in problemi di classificazione in cui gli insiemi sono linearmente separabili non possono esistere minimi locali non globali. Tuttavia, non sono noti al momento risultati di carattere generale che possano avere un ruolo significativo nella soluzione di problemi reali.

L'esperienza di calcolo mostra che in molte applicazioni le difficoltà maggiori nel processo di addestramento sono dovute alla presenza di "plateau" nella funzione d'errore piuttosto che a quella di minimi locali. Inoltre, come discusso nei paragrafi precedenti, l'addestramento di una rete non è necessariamente finalizzato all'individuazione di un punto di minimo globale del problema di ottimizzazione associato, ma ha come scopo quello di determinare un vettore di parametri cui corrisponda un valore "sufficientemente basso" dell'errore commesso sui campioni del training set, in modo tale che la rete presenti una adeguata capacità di generalizzazione. Per questi

motivi, lo studio di specifici algoritmi di ottimizzazione globale non ha costituito uno degli aspetti maggiormente rilevanti nell'ambito della letteratura neurale.

In generale, gli algoritmi di ottimizzazione globale si dividono in metodi *stocastici* e metodi *deterministici*. Nel contesto delle reti neurali la versione on-line dell'algoritmo di backpropagation rappresenta uno dei metodi di tipo stocastico maggiormente impiegati. Sono state inoltre proposte strategie di addestramento *ibride*, costituite da una fase di tipo stocastica seguita da una in cui si applica un metodo standard di ottimizzazione. Le motivazioni di simili strategie sono, da una parte quella di evitare i minimi locali mediante la fase stocastica, dall'altra quella di accelerare la convergenza verso il "minimo desiderato" con la seconda fase. L'algoritmo proposto in [99] rappresenta un metodo di tipo ibrido che combina la backpropagation on-line con un metodo di tipo gradiente coniugato.

La classe di metodi di tipo deterministico per cui esistono risultati più significativi nel campo dell'addestramento di reti neurali è quella costituita dai *metodi di omotopia*, che si basano sull'idea di "deformare" la funzione di errore originaria in una funzione la cui minimizzazione risulti più agevole, e di ridurre progressivamente la deformazione. Il metodo denominato "Expanded Range Approximation" (ERA) [49] appartiene a questa classe. La specificità del metodo ERA è rappresentata dal fatto che la deformazione è introdotta nei valori y^p, per $p = 1, \ldots, P$ delle uscite desiderate del training set. In particolare, i valori y^p sono inizialmente compressi nel valore medio

$$\bar{y} = \frac{1}{P} \sum_{i=1}^{P} y^p$$

e successivamente modificati secondo la regola

$$y^p(\lambda) = \bar{y} + \lambda(y^p - \bar{y})$$

in cui il parametro λ viene aumentato progressivamente da 0 a 1. Sono state proposte differenti versioni del metodo basate su opportune tecniche di variazione del parametro λ. Il metodo è facilmente implementabile ed è risultato efficiente (con un'appropriata scelta dei parametri) nella soluzione di problemi di addestramento.

Un'altra tecnica di omotopia [23], per cui è stato svolto uno studio teorico completo, consiste nell'introduzione di una deformazione che trasforma la funzione di attivazione dei neuroni da lineare a non lineare. In particolare, l'omotopia è definita dalla trasformazione

$$g(t; \lambda) = (1 - \lambda)t + \lambda \tanh(t),$$

in cui λ varia da 0 a 1. L'applicazione di questi risultati all'addestramento di reti multistrato è considerata in [22].

4 Reti neurali di funzioni di base radiali

Le funzioni di base radiali (*radial basis functions*) sono state introdotte nel campo dell'analisi numerica in relazione a problemi di interpolazione in uno spazio multidimensionale. Sono inoltre studiate nella teoria dell'approssimazione di funzioni multi-variabile. Una rete neurale di funzioni di base radiali (rete RBF) è una rete di tipo feedforward a due strati, in cui i nodi dello strato nascosto eseguono una trasformazione non lineare mediante funzioni di base radiali. Nel seguito verranno illustrate le proprietà delle funzioni di base radiali in relazione a problemi di interpolazione, e verranno analizzate le proprietà di approssimazione della classe di reti RBF *regolarizzate* (questa denominazione è motivata dal fatto che tali reti derivano dalla teoria della regolarizzazione). Saranno inoltre introdotte le reti RBF *generalizzate*, che consentono di approssimare il legame funzionale senza necessariamente soddisfare condizioni di interpolazione. Infine, verranno descritti algoritmi di addestramento basati su tecniche di decomposizione.

4.1 Le funzioni di base radiali

Dato un insieme di P punti $\{x^p \in R^n,\ p = 1, \ldots, P\}$ e un corrispondente insieme di P numeri reali $\{y^p \in R,\ p = 1, \ldots, P\}$, il problema dell'interpolazione consiste nel determinare una funzione $y : R^n \to R$, in una classe di funzioni assegnata, che soddisfi la condizione di interpolazione

$$y(x^p) = y^p \qquad p = 1, \ldots, P. \tag{26}$$

Una particolare tecnica di interpolazione multidimensionale consiste nella scelta di una funzione y della forma

$$y(x) = \sum_{p=1}^{P} w_p \phi(\|x - x^p\|), \tag{27}$$

dove $\phi : R^+ \to R$ è una funzione continua, usualmente denominata *funzione di base radiale*, $\|\cdot\|$ è la norma euclidea in R^n. La funzione interpolante è espressa quindi come combinazione lineare di P funzioni, ognuna delle quali ha come argomento la distanza ("raggio") $\|x - x^p\|$. I coefficienti w_p costituiscono i *pesi*, i punti di interpolazione x^p sono denominati *centri*. Alcune scelte di $\phi(r)$ sono riportate nella tabella seguente.

$\phi(r) = r$	lineare
$\phi(r) = e^{-r^2}$	Gaussiana
$\phi(r) = (r^2 + \sigma^2)^{-1/2}$	multiquadrica inversa
$\phi(r) = (r^2 + \sigma^2)^{1/2}$	multiquadrica diretta

dove $r \geq 0$ e σ è una costante positiva.

Scelta la funzione ϕ, per definire la funzione interpolante y occorre determinare il vettore dei pesi $w = (w_1, \ldots, w_P)^T$. A questo fine, utilizzando la (27) e la condizione di interpolazione (26) si definisce il seguente sistema lineare nel vettore incognito w

$$\Phi w = y, \tag{28}$$

dove $y = (y^1, \ldots, y^P)^T$, e Φ è la matrice $P \times P$ il cui generico elemento ϕ_{ji} è $\phi_{ji} = \phi(\|x^j - x^i\|)$. La matrice Φ è denominata *matrice di interpolazione*. Per definire la funzione interpolante y occorre quindi risolvere il sistema (28). In [74] sono definite delle condizioni sufficienti per assicurare la non singolarità della matrice di interpolazione Φ, nell'ipotesi in cui i punti di interpolazione $x^1, \ldots, x^P$ siano distinti. In tale ipotesi, utilizzando i risultati riportati in [74], si può dimostrare l'invertibilità della matrice Φ per ognuna delle scelte della funzione ϕ precedentemente riportate, cioè per la lineare, la Gaussiana e le multiquadriche. Inoltre, per la Gaussiana e la multiquadrica inversa, è possibile dimostrare [89] che le corrispondenti matrici di interpolazione sono matrici definite positive.

4.2 Reti RBF regolarizzate

Le tecniche di *regolarizzazione* [101] sono studiate nell'ambito di problemi di approssimazione di funzioni multivariabile, in cui l'insieme dei campioni disponibili è finito. Sia $f : R^n \to R$ una funzione che si intende approssimare, e si consideri il training set

$$T = \{(x^p, y^p) \in R^n \times R,\ p = 1, \ldots P\},$$

in cui $y^p = f(x^p)$. Come si è detto nell'introduzione, il problema della "ricostruzione" di f a partire dall'insieme T è un problema mal posto. Le tecniche di regolarizzazione determinano la funzione approssimante mediante la minimizzazione di un funzionale costituito da due termini. Il primo di tali termini misura la distanza della funzione approssimante dai campioni disponibili, cioè dai dati del training set. Il secondo termine misura la violazione di vincoli imposti sulla regolarità della funzione approssimante y. Tale termine dipende quindi dalle informazioni, note a priori, riguardanti la funzione f (si può assumere, ad esempio, che f sia almeno differenziabile). Il problema da risolvere diventa quello di determinare la funzione y che minimizza un funzionale del tipo

$$\mathcal{E}(y) = \mathcal{E}_1(y) + \mathcal{E}_2(y) = \frac{1}{2}\sum_{p=1}^{P}[y^p - y(x^p)]^2 + \frac{1}{2}\lambda\|\mathcal{P}y\|^2, \tag{29}$$

dove $\lambda > 0$ è il *parametro di regolarizzazione*, $\mathcal{P}$ è un operatore differenziale, $\|\cdot\|$ è una norma nello spazio di funzioni cui appartiene $\mathcal{P}y$. Le informazioni riguardanti la funzione da approssimare f sono "contenute" nella struttura dell'operatore $\mathcal{P}$, la cui scelta dipende quindi dalla natura del problema considerato.

Si può dimostrare che, sotto opportune ipotesi sull'operatore $\mathcal{P}$, la funzione che minimizza il funzionale (29) assume la forma

$$y(x) = \sum_{p=1}^{P} w_p \phi(\|x - x^p\|), \tag{30}$$

in cui $\phi : R^+ \to R$ è una funzione di base radiale, e $w \in R^P$ è soluzione del sistema lineare

$$(\Phi + \lambda I)w = y \tag{31}$$

dove $y = (y^1, \ldots, y^P)^T$, I è la matrice identità $P \times P$, e Φ è la matrice $P \times P$ il cui generico elemento ϕ_{ji} è

$$\phi_{ji} = \phi(\|x^j - x^i\|).$$

La (30) può essere interpretata come l'uscita di una rete feedforward, in cui la funzione di attivazione dei neuroni dello strato nascosto è proprio la funzione ϕ. Reti di questo tipo sono note in letteratura come reti RBF *regolarizzate* e sono reti feedforward con queste caratteristiche:

- presentano un solo strato nascosto;
- i neuroni dello strato nascosto sono unità di calcolo che hanno come funzione di attivazione una funzione di base e l'argomento della funzione di attivazione è costituito dalla distanza (espressa mediante la norma euclidea) tra il vettore di ingresso ed il centro dell'unità;
- il numero di neuroni dello strato nascosto è pari al numero P degli elementi del training set;
- il neurone dello strato d'uscita effettua una combinazione lineare delle uscite dei neuroni dello strato nascosto.

Per quanto riguarda le proprietà di approssimazione delle reti di regolarizzazione, si ha il seguente risultato (si veda [85]).

Teorema 11 *Per ogni funzione continua f definita su un sottoinsieme compatto $\mathcal{S}$ di R^n, esiste una rete RBF regolarizzata, cioè una funzione della forma*

$$y(x) = \sum_{p=1}^{P} w_p \phi(\|x - x^p\|),$$

tale che per ogni $x \in \mathcal{S}$ e per ogni $\epsilon > 0$ risulta

$$|f(x) - y(x)| < \epsilon.$$

Il teorema stabilisce che le reti RBF sono *approssimatori universali* per le funzioni continue. Le reti RBF hanno quindi la proprietà di approssimazione caratterizzante anche le reti multistrato. Rispetto a questa classe di reti neurali, le reti di regolarizzazione presentano una ulteriore proprietà teorica di approssimazione. In particolare, definiscono un sottoinsieme di funzioni per cui il problema dell'*approssimazione ottima* ammette soluzione.

In termini generali, sia V uno spazio vettoriale dotato di norma $\|\cdot\|$, e sia $\rho(f_1, f_2) : V \times V \rightarrow R^+$ la *funzione distanza* definita come $\rho(f_1, f_2) = \|f_1 - f_2\|$. Sia inoltre A un sottoinsieme assegnato di V. Ciò posto, il problema dell'approssimazione ottima su V è quello di determinare (ove esista) un elemento $y^* \in A$ tale che

$$\rho(y^*, f) = \min_{y \in A} \rho(y, f),$$

dove $f \in V$.

Sia V l'insieme delle funzioni continue definite su un dominio $U \subseteq R^n$, che denotiamo con $C[U]$, e si consideri il sottoinsieme di $C[U]$:

$$A = \{y \in C[U] : \ y(x) = \sum_{p=1}^{P} w_p \phi_p(x), \ w_p \in R\}, \tag{32}$$

dove le funzioni ϕ_p sono funzioni continue assegnate. Osserviamo che A è un sottospazio lineare a dimensioni finite dello spazio vettoriale $C[U]$. Da un noto teorema segue che il problema dell'approssimazione ottima su $C[U]$ ammette soluzione. Si noti che sottoinsiemi del tipo (32) sono definiti, ad esempio, da reti RBF di regolarizzazione. Per questa classe di reti *l'esistenza* dell'approssimazione ottima è quindi provata.

Si può dimostrare [35] che nel caso in cui l'insieme A è costituito da funzioni "rappresentabili" con reti multistrato aventi almeno uno strato nascosto, il problema dell'approssimazione ottima non ammette soluzione.

4.3 Reti RBF generalizzate

Si consideri la relazione ingresso-uscita di una rete RBF definita in (30). La corrispondenza "uno a uno" tra gli elementi del training set e i termini dell'espansione potrebbe rendere troppo onerosa, da un punto di vista computazionale, l'implementazione di una rete RBF. In particolare, la determinazione dei coefficienti w_p dell'espansione richiede la soluzione di un sistema lineare di dimensione $P \times P$, e ciò può essere proibitivo quando il numero P di elementi del training set è molto elevato.

A partire da queste osservazioni sono state introdotte le reti RBF *generalizzate*, in cui il numero N dei termini dell'espansione è in genere minore del numero P degli elementi del training set, e i centri $c_i \in R^n$ non coincidono necessariamente con i vettori x^p appartenenti al training set.

La funzione approssimante (rete RBF generalizzata) assume quindi la forma

$$y(x) = \sum_{i=1}^{N} w_i \phi(\|x - c_i\|).$$

In una rete RBF generalizzata sia i pesi $w_i \in R$ che i centri $c_i \in R^n$ costituiscono i parametri da determinare. Ciò implica che il legame ingresso-uscita definito da una rete RBF generalizzata dipende in modo non lineare dai parametri, come nelle reti multistrato.

Lo shema di una rete RBF è riportato in Fig.5.

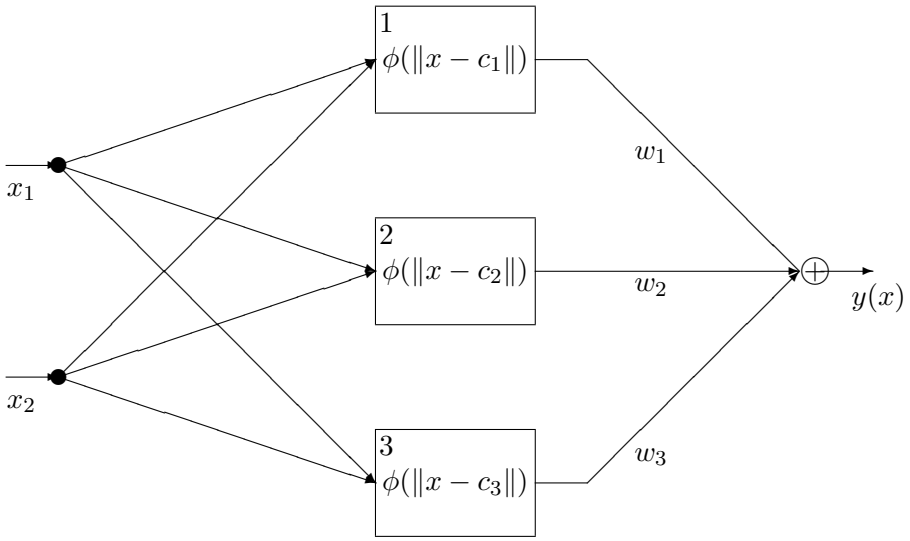


Fig.5 Rete RBF con 3 centri

Per quanto riguarda le proprietà di approssimazione, osserviamo che:

- la classe delle reti generalizzate include quella delle reti regolarizzate, per cui, sulla base del Teorema 11, si può affermare che le reti RBF generalizzate sono approssimatori universali;
- è possibile dimostrare che il problema dell'approssimazione ottima non ammette soluzione nel caso di reti RBF generalizzate.

Nel seguito faremo riferimento esclusivamente a reti RBF generalizzate.

4.4 Algoritmi di addestramento di reti RBF

Sia dato il training set $T = \{(x^p, y^p) \in R^n \times R,\ p = 1, \dots P\}$, e si consideri una rete RBF generalizzata con N centri. Indichiamo con w il vettore dei pesi e con C il vettore dei centri, ossia poniamo:

$$w = (w_1, \dots, w_N)^T \in R^N \qquad c = \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix} \in R^{nN}$$

Analogamente a quanto detto nel caso di reti multistrato, la procedura di addestramento di una rete RBF è finalizzata alla determinazione dei pesi e dei centri sulla base dei dati forniti nel training set.

In relazione a questo problema, sono adottate due diverse strategie di addestramento

- scelta *non supervisionata* dei centri e *supervisionata* dei pesi;
- scelta *supervisionata* sia dei centri che dei pesi.

La prima strategia consiste nel fissare prima i vettori c_i scegliendoli *casualmente* tra i vettori di ingresso del training set oppure con tecniche di *clustering*, e nel determinare successivamente i pesi minimizzando rispetto a w la funzione di errore. In tal caso, la funzione di errore si può assumere della forma

$$E(w) = \frac{1}{2} \sum_{p=1}^{P} \left(\sum_{i=1}^{N} w_i \phi(\|x^p - c_i\|) - y^p \right)^2 + \frac{\rho}{2} \|w\|^2,$$

dove compare anche un termine di regolarizzazione con $\rho > 0$. Ponendo $y = (y^1, \dots, y^P)^T$ e definendo la matrice $\Phi(c)$ di dimensioni $(P \times N)$ con elementi

$$(\Phi)_{pi} = \phi(\|x^p - c_i\|),$$

la minimizzazione rispetto a w si riduce a risolvere il problema di minimi quadrati:

$$\min_w E(w) \equiv \left\| \begin{pmatrix} \Phi(c) \\ \sqrt{\rho} I \end{pmatrix} w - \begin{pmatrix} y \\ 0 \end{pmatrix} \right\|^2. \tag{33}$$

Per la soluzione del problema si possono utilizzare [14] sia metodi *diretti* (in particolare, metodi basati sulla decomposizione a valori singolari) se N non è molto elevato, sia metodi *iterativi*, come, ad esempio, il metodo del gradiente coniugato.

Fissati i centri con una procedura non supervisionata, si ottiene quindi nelle restanti variabili un problema di ottimizzazione che può essere risolto con tecniche efficienti. Tuttavia, la scelta non supervisionata dei centri ha lo svantaggio di non tener conto dei valori di uscita y^p associati ai campioni di ingresso del training set.

Ciò può comportare la necessità di dover introdurre un numero molto elevato di centri per ottenere prestazioni soddisfacenti della rete RBF.
La seconda strategia [85] consiste invece nel minimizzare una funzione di errore sia rispetto ai pesi d'uscita che rispetto alle posizioni dei centri; in tal caso, introducendo anche un termine di regolarizzazione relativo ai centri, il problema di addestramento si può porre nella forma:

$$\min_{w,c} E(w,c) \equiv \frac{1}{2}\sum_{p=1}^{P}\left(\sum_{i=1}^{N} w_i \phi(\|x^p - c_i\|) - y^p\right)^2 + \frac{\rho_1}{2}\|w\|^2 + \frac{\rho_2}{2}\|c\|^2, \qquad (34)$$

con $\rho_1, \rho_2 > 0$. Alcuni esperimenti e confronti [107] sembrano indicare che la scelta supervisionata di centri e pesi consente di ottenere capacità di generalizzazione notevolmente migliori, in confronto con la prima strategia considerata.

4.5 Algoritmi di decomposizione

La soluzione del problema (34) richiede l'impiego di tecniche di ottimizzazione non lineare e può comportare difficoltà computazionali considerevoli per valori elevati della dimensione dello spazio di ingresso n e del numero di campioni P del training set. Per ovviare a tali difficoltà, preservando i vantaggi dell'addestramento supervisionato, è possibile far ricorso a tecniche di *decomposizione*, partizionando le variabili in diversi blocchi ed effettuando minimizzazioni alternate rispetto ai singoli blocchi. Per assicurare la convergenza di tecniche di questo tipo è necessario tuttavia soddisfare opportune condizioni e definire in modo appropriato gli algoritmi di minimizzazione. Sono infatti noti [88] alcuni controesempi relativi alla minimizzazione di funzioni di tre variabili in cui il *metodo delle coordinate con ricerche di linea esatte* cicla indefinitamente senza convergere ad un punto stazionario. Tale metodo può essere visto come caso particolare dell'algoritmo di decomposizione di *Gauss-Seidel* [12], in cui l'aggiornamento dei blocchi di variabili avviene mediante minimizzazioni globali nei rispettivi sottospazi. In [45], [46] sono stati stabiliti alcuni risultati generali sulla convergenza di metodi di decomposizione e sono stati definiti specifici algoritmi di minimizzazione a blocchi. In particolare, assumendo di aver partizionato il vettore delle variabili in NB blocchi e supponendo che la funzione obiettivo sia continuamente differenziabile, la convergenza del metodo di Gauss-Seidel può essere assicurata in uno dei seguenti casi:

- vettore delle variabili decomposto in *due blocchi*, cioè $NB = 2$;
- funzione obiettivo *pseudoconvessa* e insiemi di livello compatti;
- funzione obiettivo *strettamente convessa* rispetto a $NB - 2$ blocchi.

In alternativa al metodo di Gauss-Seidel è possibile definire *metodi di discesa a blocchi* in cui non è richiesta la minimizzazione globale rispetto a ciascun blocco di variabili.

Anche in questo caso è necessario imporre opportune condizioni per assicurare la convergenza globale.
In [18] sono stati studiati algoritmi di decomposizione specifici per l'addestramento di reti RBF. Un primo schema di decomposizione è quello basato sulla *decomposizione nei due blocchi pesi e centri*, e consiste nell'alternare la minimizzazione rispetto ai pesi d'uscita per posizioni fissate dei centri con una minimizzazione approssimata rispetto ai centri. In tal caso, sotto ipotesi usuali su E e sull'algoritmo utilizzato, si può garantire la convergenza a punti stazionari e si può inoltre assicurare, come nell'addestramento non supervisionato, che i pesi d'uscita corrispondano a un punto di minimo globale nello spazio delle w in corrispondenza alla scelta di c.

Osserviamo che se $\rho_1, \rho_2 > 0$, tutti gli insiemi di livello della funzione $E(w, c)$, ossia tutti gli insiemi

$$\mathcal{L}(\alpha) = \{(w, c) \in R^N \times R^{nN} : E(w, c) \leq \alpha\},$$

con $\alpha \in R$ sono compatti. In tale ipotesi, uno schema di decomposizione in due blocchi che assicuri proprietà di convergenza globale può essere, ad esempio, un algoritmo in cui, ad ogni iterazione k:

- i pesi w^{k+1} vengono calcolati risolvendo il problema di minimi quadrati lineare (33) in corrispondenza al vettore corrente dei centri c^k;
- i centri c^{k+1} vengono determinati attraverso un algoritmo di discesa che includa almeno una ricerca unidimensionale opportuna lungo la direzione d^k opposta al gradiente parziale rispetto al vettore dei centri, ossia

$$d^k = -\nabla_c E(w^{k+1}, c^k).$$

Per dimostrare la convergenza dell'algoritmo è necessario che le ricerche unidimensionali lungo le direzioni d^k soddisfino opportuni requisiti, specificati nella condizione successiva.

Condizione 1 *Per ogni k si ha:*

$$E(w^{k+1}, c^k + \eta^k d^k) \leq E(w^{k+1}, c^k);$$

inoltre, se

$$\lim_{k\to\infty} E(w^{k+1}, c^k) - E(w^{k+1}, c^k + \eta^k d^k) = 0,$$

allora per ogni sottosequenza $\{(w^{k+1}, c^k)\}_K$ convergente si ha:

$$\lim_{k\in K, k\to\infty} \frac{\nabla_c E(w^{k+1}, c^k)^T d^k}{\|d^k\|} = 0.$$

Un algoritmo di decomposizione in due blocchi è definito nello schema seguente.

Algoritmo D1: decomposizione in 2 blocchi

Dati. $c^0 \in R^{nN}$, $w^0 \in R^N$, $k = 0$.

While $\nabla E(w^k, c^k) \neq 0$ **do**

Passo 1. (Minimizzazione rispetto ai pesi)

calcola

$$w^{k+1} = \text{Arg}\min_w E(w, c^k),$$

risolvendo il problema di minimi quadrati in w

Passo 2. (Minimizzazione rispetto ai centri)

(a) calcola $d^k = -\nabla_c E(w^{k+1}, c^k)$,

(b) calcola η^k con una ricerca unidimensionale che soddisfi la Condizione 1

(c) scegli qualsiasi c^{k+1} tale che

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k + \eta^k d^k)$$

Passo 3. Poni $k = k + 1$

End While

Osserviamo che nello schema precedente si può utilizzare al Passo 2 un numero finito di iterazioni di un qualsiasi algoritmo di ottimizzazione non vincolata per determinare c^{k+1}, purchè risulti soddisfatta la condizione specificata al punto (c). Tale condizione è soddisfatta, in particolare, se si assume $c^{k+1} = c^k + \eta^k d^k$.

Si può anche introdurre un criterio di arresto nel Passo 2, attraverso una condizione del tipo $\|\nabla_c E(w^{k+1}, c^k)\| \leq \xi^k$, essendo $\xi^k > 0$ tale che $\lim_{k\to\infty} \xi^k = 0$. La Condizione 1 può essere soddisfatta utilizzando un algoritmo standard di ricerca unidimensionale, come ad esempio il metodo di Armijo già illustrato in precedenza. Vale il risultato seguente [18].

Teorema 12

Sia $\{(w^k, c^k)\}$ *la sequenza generata dall'Algoritmo* D1*; allora:*

- $\{(w^k, c^k)\}$ *ha punti di accumulazione;*
- $\{E(w^k, c^k)\}$ *converge;*
- *ogni punto di accumulazione di* $\{(w^k, c^k)\}$ *è un punto stazionario di* E.

Quando le dimensioni del problema (numero di centri, numero di ingressi, numero di campioni) sono elevate anche la minimizzazione approssimata rispetto a c può essere molto costosa. Può essere quindi opportuno decomporre ulteriormente questa fase in una successione sequenziale di minimizzazioni parziali rispetto ai singoli centri, introducendo eventualmente opportuni criteri per selezionare i centri da aggiornare. Ciò corrisponde a partizionare le variabili del problema negli $N+1$ blocchi costituiti dal vettore dei pesi w e dai centri c_i, per $i = 1, \ldots, N$.

Come nell'algoritmo precedente, i pesi possono essere determinati risolvendo un problema di minimi quadrati lineare con un algoritmo diretto o iterativo e si possono utilizzare metodi di discesa per la minimizzazione rispetto ai centri. È necessario tuttavia adottare opportune cautele per garantire la convergenza. Tenendo conto del fatto che E è strettamente convessa rispetto ai pesi w, per assicurare la convergenza occorre imporre, oltre alle condizioni usuali, anche il requisito:

$$\|c_i^{k+1} - c_i^k\| \to 0, \quad i = 1, \ldots, N.$$

In particolare, si può supporre che, nello spazio dei centri, le ricerche dimensionali lungo le direzioni

$$d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k),$$

soddisfino la condizione seguente.

Condizione 2 *Per ogni $k \geq 0$ e ogni $i = 1, \ldots, N$ si ha:*

$$E(w^{k+1}, c_1^{k+1} \ldots c_i^k + \eta_i^k d_i^k \ldots c_N^k) \leq E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k);$$

Inoltre, se

$$\lim_{k\to\infty} E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k) - E(w^{k+1}, c_1^{k+1} \ldots c_i^k + \eta_i^k d_i^k \ldots c_N^k) = 0,$$

allora:

(i) $\lim_{k\to\infty} \eta_i^k \|d_i^k\| = 0$;

(ii) *per ogni sottosequenza $\{(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k)\}_K$ convergente si ha*

$$\lim_{k\in K, k\to\infty} \frac{\nabla_{c_i} E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k)^T d_i^k}{\|d_i^k\|} = 0.$$

Con la scelta fatta per d_i^k, la Condizione 2 può essere soddisfatta, ad esempio, con un algoritmo tipo-Armijo in cui si imponga una limitazione superiore sul passo, oppure con tecniche di ricerca unidimensionale basate su condizioni di accettabilità del tipo

$$E(w^{k+1}, c_1^{k+1} \ldots c_i^k + \eta_i^k d_i^k \ldots c_N^k) \leq E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k) - \gamma_i (\eta_i^k \|d_i^k\|)^2.$$

Un algoritmo di decomposizione negli $N+1$ blocchi $w, c_1 \ldots, c_N$, è riportato nello schema successivo.

Algoritmo D2: decomposizione in $N+1$ blocchi

Dati. $c^0 \in R^{nN}$, $w^0 \in R^N$, $k = 0$, $\tau_i > 0$, $\xi_i^k > 0$ tale che $\xi_i^k \to 0$ per $i = 1, \ldots, N$.

While $\nabla E(w^k, c^k) \neq 0$ **do**

Passo 1. (Minimizzazione rispetto ai pesi)

calcola

$$w^{k+1} = \text{Arg}\min_w E(w, c^k),$$

risolvendo un problema di minimi quadrati in w

Passo 2. (Minimizzazione rispetto ai centri)

For $i = 1, \ldots, N$ **do**

If $\|\nabla_{c_i} E(w^{k+1}, c^k)\| \leq \xi_i^k$ **then**

poni $c_i^{k+1} = c_i^k$

else

(a) determina $d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k)$

(b) calcola η_i^k con una tecnica di ricerca che soddisfi la Condizione 2

(c) calcola $\tilde{c}_i^k$ tale che:

$$E(w^{k+1}, c_1^{k+1} \ldots \tilde{c}_i^k \ldots c_N^k) \leq E(w^{k+1}, c_1^{k+1} \ldots c_i^k + \eta_i^k d_i^k \ldots c_N^k)$$

If $E(w^{k+1}, c_1^{k+1} \ldots \tilde{c}_i^k \ldots c_N^k) \leq E(w^{k+1}, c_1^{k+1} \ldots c_i^k \ldots c_N^k) - \tau_i \|\tilde{c}_i^k - c_i^k\|^2$ **then**

poni $c_i^{k+1} = \tilde{c}_i^k$

else

poni $c_i^{k+1} = c_i^k + \eta_i^k d_i^k$

end if

end if

End For

Passo 3. Poni $k = k + 1$

End While

Le condizioni (b)(c) del Passo 2 garantiscono, fra l'altro, che la distanza $\|c_i^{k+1} - c_i^k\|$ tenda a zero e possono essere soddisfatte, in particolare, assumendo

$$c_i^{k+1} = c_i^k + \eta_i^k d_i^k.$$

La differenza importante rispetto al caso dell'Algoritmo D1 è che non è più possibile effettuare le minimizzazioni parziali con un qualsiasi algoritmo di discesa. Infatti, i punti di tentativo $\tilde{c}_i^k$, che possono essere generati con qualsiasi metodo, potrebbero non essere accettati, anche se producono una riduzione della funzione obiettivo, se hanno l'effetto di produrre spostamenti troppo "grandi" dal punto corrente c_i^k.

Si osservi che il test al Passo 2 consente di evitare la minimizzazione rispetto a c_i quando il gradiente parziale ∇E_{c_i} è sufficientemente piccolo. Inoltre, il calcolo della funzione d'errore in corrispondenza ad una nuovo vettore c_i può essere organizzato in modo da ridurre considerevolmente i tempi di calcolo memorizzando le uscite della rete calcolate nel punto corrente c_i^k. Infatti, si ha

$$\begin{aligned} &y(x^p; w^{k+1}, c_1^{k+1}, \ldots, c_i, \ldots, c_N^k) \\ &\quad = y(x^p; w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_N^k) - w_i^k \phi(\|x^p - c_i^k\|) + w_i^k \phi(\|x^p - c_i\|), \end{aligned}$$

per cui il calcolo della "nuova" uscita richiede poche operazioni elementari. Le proprietà di convergenza dell'algoritmo sono riportate nel teorema seguente [18].

Teorema 13 *Sia* $\{(w^k, c^k)\}$ *la sequenza generata dall'Algoritmo* D2. *Allora:*

(i) $\{(w^k, c^k)\}$ *ammette punti limite;*

(ii) *la sequenza* $\{E(w^k, c^k)\}$ *converge;*

(iii) *si ha* $\lim_{k\to\infty} \|w^{k+1} - w^k\| = 0$;

(iv) *per* $i = 1, \ldots, M$, *segue* $\lim_{k\to\infty} \|c_i^{k+1} - c_i^k\| = 0$;

(v) *ogni punto limite di* $\{(w^k, c^k)\}$ *è un punto stazionario di* E.

L'algoritmo ha dato buoni risultati su problemi test di addestramento ed è risultato più efficiente di un algoritmo Quasi-Newton per la minimizzazione rispetto a w, c anche per dimensioni non elevate. La ricerca in corso riguarda lo studio di metodi iterativi per la soluzione approssimata del problema di minimi quadrati lineari definito al Passo 1, e l'impiego di tecniche non monotone nella fase di minimizzazione rispetto ai centri.

5 Support Vector Machines

Le Support Vector Machines (SVM) costituiscono una classe di "macchine di apprendimento", recentemente introdotte in letteratura [104], che traggono origine da concetti riguardanti la teoria statistica dell'apprendimento. Nel caso di insiemi linearmente separabili, l'idea alla base delle SVM è di costruire un iperpiano di separazione (denominato *iperpiano ottimo*) che massimizzi il margine di separazione dagli elementi delle due classi. Questa tecnica di separazione è motivata da alcuni risultati della teoria statistica dell'apprendimento, in base ai quali, l'errore sui dati di test (cioè l'errore di generalizzazione) di una macchina di apprendimento è limitato superiormente dalla somma dell'errore sui dati di training e di un termine che dipende dalla *Vapnik-Chervonenkis (VC) dimension*, che è una misura della *capacità* di classificazione della famiglia di funzioni realizzata dalla macchina di apprendimento. Nel caso di insiemi linearmente separabili, l'iperpiano ottimo definisce una superficie di separazione che fornisce un errore di training nullo e minimizza il secondo termine dipendente dalla VC dimension. Di conseguenza, le SVM possono presentare buone proprietà di generalizzazione in problemi di classificazione. La tecnica di apprendimento delle SVM si può estendere al caso di insiemi non linearmente separabili, e al caso di problemi di approssimazione.

Nelle brevi note seguenti, gli aspetti riguardanti la teoria statistica dell'apprendimento non saranno analizzati (approfondimenti possono essere trovati in [16], [104], [105]). Verrà invece mostrato che il problema di addestramento di SVM è riconducibile, mediante la teoria della dualità, ad un problema di programmazione quadratica convessa con vincoli lineari (l'insieme dei vincoli è costituito da un vincolo lineare e da vincoli di box). Poichè tale problema è denso e, in molte applicazioni, di dimensione elevata, gli algoritmi di addestramento proposti i letteratura si basano sull'impiego di tecniche di decomposizione. Verranno quindi descritti i principali metodi di decomposizione per l'addestramento di SVM.

5.1 SVM lineari: iperpiano ottimo

Si considerino due insiemi disgiunti di punti $\mathcal{A}$ e $\mathcal{B}$ in R^n, e sia P la cardinalità di $\mathcal{A} \cup \mathcal{B}$. Si assuma che $\mathcal{A}$ e $\mathcal{B}$ siano *linearmente separabili*, cioè, che esista un iperpiano $H = \{x \in R^n : \ w^T x + \theta = 0\}$ tale che

$$\begin{array}{rcll} w^T x^i + \theta & \geq & 1, & x^i \in \mathcal{A} \\ w^T x^i + \theta & \leq & -1, & x^i \in \mathcal{B}. \end{array} \tag{35}$$

Dato un iperpiano di separazione H, cioè, una coppia (w, θ) per cui la (35) è verificata, si definisce *margine di separazione* di H la minima distanza ρ tra i punti in $\mathcal{A} \cup \mathcal{B}$ e l'iperpiano H, cioè

$$\rho(w, \theta) = \min_{x^i \in \mathcal{A} \cup \mathcal{B}} \left\{ \frac{|w^T x^i + \theta|}{\|w\|} \right\}.$$

Il margine di separazione di un iperpiano di separazione è indicato in Figura 6.

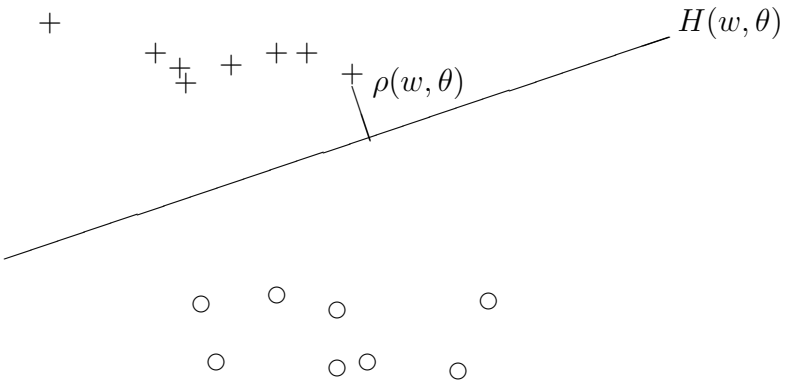


Fig.6 Margine di separazione di un iperpiano

Si definisce *iperpiano ottimo* l'iperpiano di separazione $H(w^\star, \theta^\star)$ avente margine di separazione massimo. Si può dimostrare *l'esistenza e l'unicità* dell'iperpiano ottimo [104]. Determinare l'iperpiano ottimo equivale a risolvere il problema

$$\begin{array}{ll} \max\limits_{w,\theta} & \rho(w,\theta) \\ & \\ & w^T x^i + \theta \geq \quad 1, \quad x^i \in \mathcal{A} \\ & w^T x^i + \theta \leq -1, \quad x^i \in \mathcal{B}. \end{array} \tag{36}$$

Si può dimostrare [104] che il problema (36) è equivalente al problema di programmazione quadratica convessa

$$\begin{array}{ll} \min_{w,\theta} & \frac{1}{2}\|w\|^2 \\ & \\ & y^i(w^T x^i + \theta) - 1 \geq 0, \quad i = 1, \ldots, P, \end{array} \tag{37}$$

in cui $y^i = +1$ se $x^i \in \mathcal{A}$ e $y^i = -1$ se $x^i \in \mathcal{B}$. Considereremo ora *la formulazione duale* del problema (37). Le motivazioni sono le seguenti:

- i vincoli del problema (37) saranno sostituiti da vincoli "più semplici" sulle variabili duali (moltiplicatori di Lagrange);
- nella formulazione duale, i vettori di training compariranno attraverso prodotti scalari tra i vettori stessi, e ciò consentirà di estendere la trattazione al caso di insiemi che non sono linearmente separabili.

Il problema primale (37) è un problema quadratico convesso con vincoli lineari; sappiamo inoltre che esiste la soluzione ottima $(w^\star, \theta^\star)$. Da risultati noti della teoria della dualità segue che il valore ottimo del problema primale è uguale al valore ottimo del problema duale (assenza di *gap di dualità*).

Definita la funzione Lagrangiana

$$L(w,\theta,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \alpha_i[y^i(w^T x^i + \theta) - 1],$$

introduciamo il seguente problema duale, noto come *duale di Wolfe* [68]

$$\begin{aligned} \max_{w,\theta,\alpha} L(w,\theta,\alpha) = \quad & \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \alpha_i[y^i(w^T x^i + \theta) - 1] \\ & w = \sum_{i=1}^{P} \alpha_i y^i x^i \\ & \sum_{i=1}^{P} \alpha_i y^i = 0 \qquad \alpha_i \geq 0 \qquad i = 1,\ldots,P. \end{aligned} \tag{38}$$

Il problema (38) è equivalente al problema di *programmazione quadratica* convesso:

$$\begin{aligned} \min_{\alpha} W(\alpha) = \quad & \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y^i y^j (x^i)^T x^j \alpha_i \alpha_j - \sum_{i=1}^{P} \alpha_i \\ & \sum_{i=1}^{P} \alpha_i y^i = 0 \qquad \alpha_i \geq 0 \qquad i = 1,\ldots,P. \end{aligned} \tag{39}$$

Dalla teoria della dualità segue che:

- il problema (39) ammette almeno una soluzione ottima $\alpha^\star$;
- il vettore $w^\star$ che risolve il problema (38) è dato da

  $$w^\star = \sum_{i=1}^{P} \alpha_i^\star y^i x^i,$$

  e quindi $w^\star$ dipende esclusivamente dai campioni x^i (denominati *support vectors*, *vettori di supporto*) i cui corrispondenti moltiplicatori $\alpha_i^\star$ sono diversi da zero;
- $((w^\star,\theta^\star),\alpha^\star)$ costituisce una coppia (soluzione ottima-vettore di moltiplicatori di Lagrange), per cui valgono le condizioni di complementarità

  $$\alpha_i^\star[y^i((w^\star)^T x^i + \theta^\star) - 1] = 0 \qquad i = 1,\ldots,P; \tag{40}$$

  ne segue che la soglia $\theta^\star$ può essere determinata utilizzando una delle precedenti condizioni corrispondente ad un moltiplicatore $\alpha_i^\star \neq 0$;
- la funzione di decisione assume la forma

  $$f(x) = \mathrm{sgn}((w^*)^T x + \theta^\star) = \mathrm{sgn}(\sum_{i=1}^{P} \alpha_i^\star y^i (x^i)^T x + \theta^\star).$$

Nella figura successiva è indicata la posizione dei vettori di supporto.

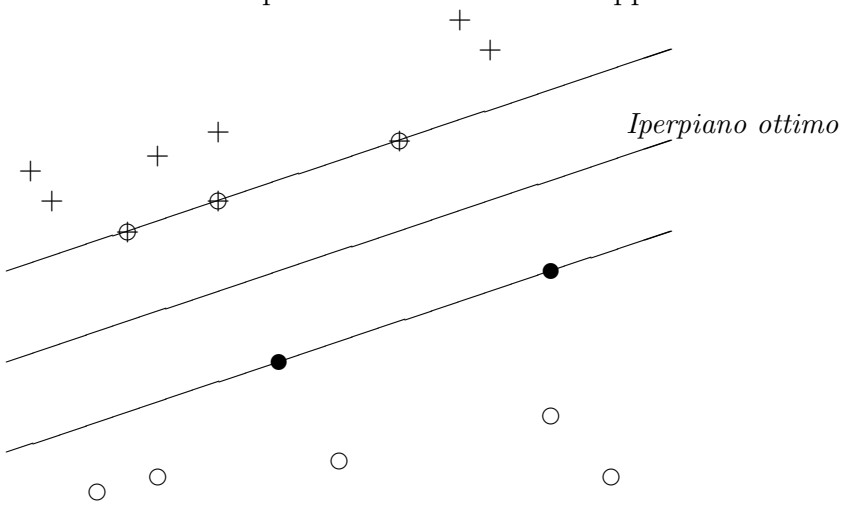


Fig.7 Iperpiano ottimo e vettori di supporto

5.2 Il caso non separabile linearmente

Siano $\mathcal{A}$ e $\mathcal{B}$ due insiemi disgiunti di punti in R^n, e si assuma che $\mathcal{A}$ e $\mathcal{B}$ non siano linearmente separabili (il sistema di disequazioni (35) non ammette quindi soluzione). Si introducano nel sistema (35) delle variabili *artificiali* positive ξ^i, con $i = 1, \ldots, P$:

$$\begin{array}{rll} w^T x^i + \theta & \geq & 1 - \xi^i, \quad x^i \in \mathcal{A} \\ w^T x^i + \theta & \leq & -1 + \xi^i, \quad x^i \in \mathcal{B} \\ \xi^i \geq 0, & & i = 1, \ldots, P. \end{array} \tag{41}$$

Si noti che, se un vettore di ingresso x^i non è classificato correttamente (ad esempio, $x^i \in \mathcal{A}$ e $w^T x^i + \theta < 0$) la corrispondente variabile ξ^i risulta maggiore di 1. Quindi, il termine

$$\sum_{i=1}^{P} \xi^i$$

è un upper bound del numero di errori di classificazione dei vettori di training. Appare naturale perciò aggiungere alla funzione obiettivo del problema (37) il termine $C\sum_{i=1}^{P} \xi^i$, in cui $C > 0$ è un parametro che "pesa" l'errore di training. Il problema primale assume la forma

$$\begin{array}{ll} \min\limits_{w,\theta,\xi} & \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{P} \xi^i \\ & y^i(w^T x^i + \theta) - 1 + \xi^i \geq 0 \\ & \xi^i \geq 0, \qquad\qquad i = 1, \ldots, P. \end{array} \tag{42}$$

Il duale di Wolfe (riformulato) del problema (42) è della forma

$$\begin{aligned} \min_{\alpha} \ W(\alpha) = \ & \tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y^i y^j (x^i)^T x^j \alpha_i \alpha_j - \sum_{i=1}^{P} \alpha_i \\ & \sum_{i=1}^{P} \alpha_i y^i = 0 \\ & 0 \le \alpha_i \le C \qquad i = 1, \ldots, P. \end{aligned} \tag{43}$$

5.3 SVM non lineari

Nel caso di insiemi di R^n non separabili linearmente, per migliorare la capacità di classificazione, possono essere impiegate opportune funzioni non lineari che operano una trasformazione da R^n ad uno spazio di dimensione superiore dove i due insiemi risultano linearmente separabili. Siano $x^i \in R^n$, $i = 1, \ldots, P$ i vettori di training e $y^i \in \{-1, 1\}$ le corrispondenti etichette che individuano la classe di appartenenza di ciascun vettore. Si consideri una trasformazione non lineare

$$\phi : R^n \to \mathcal{H}$$

in cui $\mathcal{H}$ è uno spazio a dimensione maggiore di n (eventualmente infinita). Lo spazio $\mathcal{H}$ viene denominato *feature space*. Si consideri quindi il problema della determinazione dell'iperpiano ottimo nel feature space $\mathcal{H}$ in cui "esistono" i vettori "trasformati" $\phi(x^i)$, $i = 1, \ldots, P$. Si può pensare di applicare la metodologia descritta precedentemente, sostituendo x^i con $\phi(x^i)$. In particolare:

- il corrispondente del problema (43) è il problema

$$\begin{aligned} \min_{\alpha} \ S(\alpha) = \ & \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y^i y^j \phi(x^i)^T \phi(x^j) \alpha_i \alpha_j - \sum_{i=1}^{P} \alpha_i \\ & \sum_{i=1}^{P} \alpha_i y^i = 0 \qquad 0 \le \alpha_i \le C \qquad i = 1, \ldots, P. \end{aligned} \tag{44}$$

- il vettore $w^\star$ risulta definito nel seguente modo

$$w^\star = \sum_{i=1}^{P} \alpha_i^\star y^i \phi(x^i);$$

- la funzione di decisione assume la forma

$$f(x) = \mathrm{sgn}((w^*)^T \phi(x) + \theta^\star) = \mathrm{sgn}(\sum_{i=1}^{P} \alpha_i^\star y^i \phi(x^i)^T \phi(x) + \theta^\star).$$

È importante osservare che non è necessaria la rappresentazione esplicita della trasformazione ϕ, ma è sufficiente utilizzare una *funzione di kernel* K, cioè una funzione simmetrica tale che per $x, z \in R^n$ si ha

$$K(x,z) = \phi(x)^T \phi(z) \tag{45}$$

Il Teorema di Mercer [73] stabilisce delle condizioni necessarie e sufficienti per assicurare che una funzione simmetrica $K(x,z)$ rappresenti un kernel, cioè che esista una trasformazione ϕ di cui K rappresenti il prodotto interno secondo la (45). In particolare, richiedere che la funzione di kernel soddisfi le condizioni del Teorema di Mercer equivale a richiedere che la matrice

$$\Big(K(x^i,x^j)\Big)_{i,j=1}^{P} = \begin{pmatrix} K(x^1,x^1) & \dots & K(x^1,x^P) \\ & \vdots & \\ K(x^P,x^1) & \dots & K(x^P,x^P) \end{pmatrix}$$

risulti semidefinita positiva per ogni insieme di vettori di training $\{x^1,\dots,x^P\}$. Kernel che soddisfano le condizioni del Teorema di Mercer sono denominati *kernel di tipo Mercer*. Il problema (44) può quindi essere scritto come segue

$$\begin{aligned} \min_{\alpha} S(\alpha) = \ & \tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y^i y^j K(x^i,x^j)\alpha_i\alpha_j - \sum_{i=1}^{P}\alpha_i \\ & \sum_{i=1}^{P}\alpha_i y^i = 0 \\ & 0 \le \alpha_i \le C \qquad i = 1,\dots,P. \end{aligned} \tag{46}$$

Si noti che il problema (46), nel caso di kernel di tipo Mercer, risulta essere un *problema di programmazione quadratica convessa*. Esempi di kernel di tipo Mercer sono:

$K(x,z) = (x^T z + 1)^p$ *kernel polinomiale* (p intero ≥ 1)

$K(x,z) = e^{-\|x-z\|^2/2\sigma^2}$ *kernel gaussiano* ($\sigma > 0$)

$K(x,z) = \tanh(\beta x^T z + \gamma)$ *kernel di tipo tangente iperbolica* (opportuni valori dei parametri β e γ).

La funzione di decisione può essere scritta nella forma

$$f(x) = \mathrm{sgn}\Big(\sum_{i=1}^{P}\alpha_i^\star y^i K(x,x^i) + \theta^\star\Big).$$

Osserviamo che una SVM con kernel gaussiano corrisponde ad una *rete neurale di funzioni radial basis*, mentre nel caso di SVM con kernel di tipo tangente iperbolica, si ha una *rete neurale multistrato*, con uno strato di neuroni "nascosti".

5.4 SVM per problemi di regressione

In un problema di regressione, ogni osservazione del training set è costituita da una coppia (x^i, y^i), in cui il pattern $x^i \in R^n$ e l'etichetta $y^i \in R$. Si consideri un modello ingresso-uscita lineare, cioè una funzione $f : R^n \to R$ della forma

$$f(x; w, \theta) = w^T x + \theta.$$

Sia $\epsilon > 0$ il *grado di precisione desiderato* con il quale si vuole approssimare la funzione rappresentata dai campioni del training set, mediante il modello f. La stima del modello relativa ad un pattern x^i è considerata "corretta" se risulta

$$|y^i - w^T x^i - \theta| \leq \epsilon.$$

Introduciamo la *loss function*

$$|y - f(x; w, \theta)|_\epsilon = \max\{0, |y - f(x; w, \theta)| - \epsilon\} \tag{47}$$

e definiamo l'errore di training

$$E = \sum_{i=1}^{P} |y^i - f(x^i; w, \theta)|_\epsilon.$$

L'errore di training è nullo se e solo se il seguente sistema di disequazioni è soddisfatto

$$\begin{gathered} w^T x^i + \theta - y^i \leq \epsilon \\ y^i - w^T x^i - \theta \leq \epsilon \\ i = 1, \ldots, P. \end{gathered} \tag{48}$$

Si introducano in (48) le variabili artificiali ξ^i, $\hat{\xi}^i$, con $i = 1, \ldots, P$:

$$\begin{aligned} & w^T x^i + \theta - y^i \leq \epsilon + \xi^i \\ & y^i - w^T x^i - \theta \leq \epsilon + \hat{\xi}^i \\ & \xi^i, \hat{\xi}^i \geq 0, \qquad\qquad i = 1, \ldots, P. \end{aligned}$$

Si noti che la quantità

$$\sum_{i=1}^{P} \left(\xi^i + \hat{\xi}^i\right)$$

costituisce un upper bound dell'errore di training. Si può procedere perciò come nel caso di SVM lineari per problemi di classificazione, e considerare quindi il problema

$$\begin{aligned} \min_{w,\theta,\xi,\hat{\xi}} \quad & \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \left(\xi^i + \hat{\xi}^i\right) \\ & w^T x^i + \theta - y^i \leq \epsilon + \xi^i \\ & y^i - w^T x^i - \theta \leq \epsilon + \hat{\xi}^i \\ & \xi^i, \hat{\xi}^i \geq 0, \qquad\qquad i = 1, \ldots P. \end{aligned} \tag{49}$$

La teoria della dualità e l'impiego di funzioni kernel consentono di generalizzare la trattazione al caso di modelli di regressione non lineari, in analogia con quanto fatto in problemi di classificazione. In particolare, il problema di addestramento di una SVM per la regressione non lineare risulta definito come segue

$$\begin{aligned} \min_{\alpha,\hat{\alpha}} W(\alpha,\hat{\alpha}) = \quad & \tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}(\hat{\alpha}_i-\alpha_i)(\hat{\alpha}_j-\alpha_j)K(x^i,x^j) - \sum_{i=1}^{P}(\hat{\alpha}_i-\alpha_i)\,y^i + \epsilon\sum_{i=1}^{P}(\hat{\alpha}_i+\alpha_i) \\ & \sum_{i=1}^{P}(\hat{\alpha}_i-\alpha_i) = 0 \\ & 0 \le \alpha \le C \qquad i=1,\ldots,P \\ & 0 \le \hat{\alpha} \le C \qquad i=1,\ldots,P, \end{aligned} \tag{50}$$

dove $K(x^i,x^j)$ è una funzione di kernel. Il problema (50) è un problema di programmazione quadratica convessa, la cui soluzione $(\alpha^\star,\hat{\alpha}^\star)$ permette di definire la funzione di regressione nella forma

$$f(x) = \sum_{i=1}^{P}(\hat{\alpha}_i^\star - \alpha_i^\star)\,K(x,x^i) + \theta^\star,$$

in cui $\theta^\star$ può essere determinato utilizzando le condizioni di complementarità. Le strutture dei problemi di addestramento di SVM per la regressione e per la classificazione sono simili. Tuttavia, in pratica l'addestramento di SVM per problemi di regressione è più complesso dell'addestramento di SVM per la classificazione, a causa del fatto che occorre determinare simultaneamente i valori "ottimali" di due parametri, cioè di ϵ e di C. In genere il *tuning* di tali parametri viene effettuato con tecniche di cross-validation.

5.5 Algoritmi di addestramento di SVM

Come mostrato nei precedenti paragrafi, il problema di addestramento di una SVM si può ricondurre alla soluzione del problema

$$\begin{aligned} \min_{\alpha} f(\alpha) = \quad & \tfrac{1}{2}\alpha' Q\alpha - e'\alpha \\ & y'\alpha = 0 \\ & 0 \le \alpha \le Ce, \end{aligned} \tag{51}$$

dove $\alpha \in R^P$, Q è una matrice $P \times P$ semidefinita positiva, $e \in R^P$ è il vettore costituito da elementi uguali a 1, $y \in \{-1,1\}^P$, C è uno scalare positivo. Il problema (51) è un problema quadratico convesso con vincoli di struttura molto semplice.

Tuttavia, in problemi di dimensione elevata (molto frequenti nelle applicazioni), molti metodi classici di ottimizzazione non possono essere impiegati poichè la matrice Q è densa. Per tale motivo sono stati proposti in letteratura vari metodi di decomposizione, che si basano sulla risoluzione di una sequenza di sottoproblemi di dimensioni inferiori rispetto a quella del problema originario. In una generica strategia di decomposizione, ad ogni iterazione k, il vettore delle variabili α^k è partizionato in due sottovettori $(\alpha^k_W, \alpha^k_{\overline{W}})$, in cui $W \subset \{1, \ldots, P\}$ identifica le variabili del sottoproblema che deve essere risolto ed è denominato *working set*, e $\overline{W} = \{1, \ldots, P\}/W$ (la dipendenza di W e $\overline{W}$ da k è stata omessa per non appesantire la notazione). Quindi, a partire dal punto corrente ammissibile $(\alpha^k_W, \alpha^k_{\overline{W}})$, il sottovettore α^{k+1}_W viene determinato risolvendo il sottoproblema

$$\begin{aligned} \min_{\alpha_W} \quad & f(\alpha_W, \alpha^k_{\overline{W}}) \\ & y'_W \alpha_W = -y'_{\overline{W}} \alpha^k_{\overline{W}} \\ & 0 \leq \alpha_W \leq C e_W \end{aligned} \tag{52}$$

Il sottovettore $\alpha^{k+1}_{\overline{W}}$ non viene modificato, cioè $\alpha^{k+1}_{\overline{W}} = \alpha^k_{\overline{W}}$, e il vettore aggiornato risulta essere $\alpha^{k+1} = (\alpha^{k+1}_W, \alpha^{k+1}_{\overline{W}})$. In generale, la cardinalità q del working set, cioè la dimensione del sottoproblema, è prefissata sulla base del risolutore utilizzato e della capacità computazionale disponibile. Si osservi che, per la presenza del vincolo lineare di uguaglianza, il minimo numero di variabili che possono essere modificate ad ogni iterazione è due. La regola di selezione del working set ha un ruolo fondamentale per assicurare proprietà di convergenza della sequenza generata $\{\alpha^k\}$.

Uno dei metodi più semplici di decomposizione è il *Sequential Minimal Optimization* (SMO) Algorithm, proposto in [84], in cui il working set è costituito esattamente da 2 elementi. In tal caso, una soluzione del sottoproblema (52) può essere ottenuta analiticamente, per cui non è necessario l'impiego di un algoritmo iterativo di ottimizzazione. L'algoritmo SMO è perciò di facile realizzazione, e nelle varie implementazioni proposte, la scelta delle due variabili del sottoproblema viene effettuata mediante tecniche euristiche finalizzate all'individuazione di quelle variabili la cui ottimizzazione possa determinare un decremento significativo della funzione obiettivo del problema originario.

Una versione modificata di SMO è stata proposta in [56], dove le due variabili selezionate ad ogni iterazione sono quelle corrispondenti alla "massima violazione" delle condizioni di Karush-Kuhn-Tucker (KKT). Tale implementazione di SMO può essere vista come caso particolare dell'algoritmo SVM^{light} proposto in [55], in cui la dimensione q del working set può essere un qualsiasi intero pari, e il working set è costruito mediante una procedura che inserisce successivamente coppie di variabili connesse alla violazione delle condizioni KKT.

La convergenza asintotica dell'algoritmo SVM^{light} è stata dimostrata in [61], nell'ipotesi che

$$\min_{I:\ |I|\leq q}(\mathrm{eig}_{\min}(Q_{II})) > 0, \tag{53}$$

dove I è un qualsiasi sottoinsieme di $\{1,\ldots,P\}$ con $|I| \leq q$, $\mathrm{eig}_{\min}(Q_{II})$ denota il minimo autovalore della matrice Q_{II}. Si osservi che l'assunzione (53) implica che la funzione obiettivo è strettamente convessa rispetto ad ogni blocco di variabili di cardinalità minore o uguale a q. Nel caso particolare $q = 2$, in [62] è stata dimostrata la convergenza asintotica dell'algorimo anche se l'ipotesi (53) non è soddisfatta (il caso $q = 2$ corrisponde all'algoritmo SMO). In [80] è stata definita una versione modificata dell'algoritmo SVM^{light} in cui ad ogni iterazione viene risolto un sottoproblema della forma

$$\begin{aligned}\min_{\alpha_W} \quad & f(\alpha_W, \alpha^k_{\overline{W}}) + \tau\|\alpha_W - \alpha^k_W\|^2 \\ & y'_W\alpha_W = -y'_{\overline{W}}\alpha^k_{\overline{W}} \\ & 0 \leq \alpha_W \leq Ce_W\end{aligned}$$

dove la funzione obiettivo presenta il termine addizionale di tipo *proximal point* $\tau\|\alpha_W - \alpha^k_W\|^2$, con $\tau > 0$. L'impiego di tale tecnica consente di assicurare la convergenza dell'algoritmo anche se l'ipotesi (53) non è verificata e senza restrizioni sulla dimensione q del working set.

Una strategia alternativa ai metodi proposti per risolvere il problema (51) è stata definita in [71]. In particolare, il problema primale (42) viene modificato aggiungendo alla funzione obiettivo il termine $1/2\theta^2$. Di conseguenza, il problema duale risulta essere un problema quadratico convesso con vincoli di box. Rispetto alla formulazione (51) non compare quindi il vincolo lineare di uguaglianza. Ciò consente l'impiego dell'algoritmo di decomposizione denominato *Successive OverRelaxation* (SOR), che è un metodo originariamente sviluppato per la soluzione di sistemi lineari di grande dimensione [79], e successivamente applicato a problemi di programmazione matematica [25], [69]. L'algoritmo SOR converge *linearmente* alla soluzione del problema quadratico convesso, ed è stato applicato in [71] a problemi di addestramento di SVM con milioni di variabili. In [32], viene introdotta una ulteriore modifica nel problema primale, per cui si ottiene un problema duale quadratico in cui compaiono esclusivamente vincoli di non negatività sulle variabili. Il problema viene quindi trasformato in uno equivalente non vincolato in cui la funzione obiettivo è una funzione quadratica a tratti. Per la soluzione del problema non vincolato è stato quindi proposto un metodo tipo Newton in cui per il calcolo della direzione si utilizza la *matrice Hessiana generalizzata* al posto della matrice Hessiana.

Infine, *metodi a punti interni* per l'addestramento di SVM *lineari* sono stati studiati in [30] e sono risultati efficienti nella soluzione di problemi a larga scala.

References

[1] Agmon, S. (1954), "The relaxation method for linear inequalities," *Canadian Journal of Mathematics*, vol. 6, pp. 382-392.

[2] Al–Baali, M. (1985), "Descent property and global convergence of the Fletcher-Reeves method with inexact line search," *IMA Journal on Numerical Analysis*, vol. 5, pp. 121-124.

[3] Ampazis, N., S.J. Perantonis (2002), "Two highly efficient second-order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, vol. 13, pp. 1064-1074.

[4] Anthony, M., P.L. Bartlett (1999), *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, Cambridge.

[5] Barron, A.R. (1993), "Universal approximation bounds for superposition of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, pp. 930-945.

[6] Barzilai, J., J.M. Borwein (1988), "Two-point step size gradient method," *IMA Journal on Numerical Analysis*, vol. 8, pp. 141-148.

[7] Battiti, R., F. Masulli (1990), "BFGS optimization for faster and automated supervised learning," *Proceedings of the International Neural Network Conference (INNC 90*, pp. 757-760.

[8] Baum, E.B., D.Haussler (1989), "What size net gives valid generalization ?," *Neural Computation*, vol. 1, pp. 151-160.

[9] Bertsekas, D.P., J. Tsitsiklis (1996), *Neuro-Dynamic Programming*, Athena, Boston, MA.

[10] Bertsekas, D.P. (1996), "Incremental least squares methods and the extended Kalman filter," *SIAM Journal on Optimization*, vol. 6, pp. 807-822.

[11] Bertsekas, D.P. (1997), "A new class of incremental gradient methods for least squares problems," *SIAM Journal on Optimization*, vol. 7, pp. 913-926.

[12] Bertsekas, D.P. (1999), *Nonlinear programming*, Athena Scientific, Boston.

[13] Bishop, C. (1995), *Neural Networks for Pattern Recognition*, Clarendon, Oxford, U.K.

[14] Bjorck, A. (1996), *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia.

[15] Bryson, A.E., Jr., Y.C. Ho (1969), *Applied Optimal Control*, Blaisdell. (Ristampa, 1975, Hemisphere Publishing, Washington, DC).

[16] Burges, C.J.C. (1998), "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167.

[17] Burke, L.L., J.P. Ignizio (1992), "Neural networks and operations research: An overview," *Computers and Operations Research*, vol. 19, pp. 179-189.

[18] Buzzi, C., L. Grippo, M. Sciandrone (2001), "Convergent decomposition techniques for training RBF neural networks," *Neural Computation*, Vol. 13, pp. 1891-1920.

[19] Cetin, B.C., J. Barhen, J. Burdick (1993), "Terminal repeller unconstrained subenergy tunneling (TRUST) for fast global optimization," *Journal of Optimization Theory and Applications*, vol. 77, pp. 97-126.

[20] Chamberlain, M.R., M.J.D. Powell, C. Lemaréchal, H.C. Pedersen (1982), "The watchdog technique for forcing convergence in algorithms for constrained optimization," *Mathematical Programming Study*, vol. 16, pp. 1-17.

[21] Cichocki, A., R. Unbehauen (1993), *Neural Networks for Optimization and Signal Processing*, Wiley, New York.

[22] Coetzee, F.M., V.L. Stonick (1995), "Topology and geometry of weight solutions in multilayer networks," *Neural Computation*, vol. 7, pp. 672-705.

[23] Coetzee, F.M., V.L. Stonick (1996), "On a natural homotopy between linear and nonlinear single-layer networks," *IEEE Transactions on Neural Networks*, vol. 7, pp. 307-317.

[24] Cristianini, N., J. Shawe-Taylor (2000), *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge.

[25] Cryer, C.W. (1971), "The solution of a quadratic programming problem using systematic overrelaxation," *SIAM Journal on Control and Optimization*, vol. 9, pp. 385-392.

[26] Dan, H., N. Yamashita, M. Fukushima (2002), "Convergence properties of the inexact Levenberg-Marquardt method under local error bound conditions," *Optimization Methods and Software*, vol. 17, pp. 605-626.

[27] Dennis ,J.E., R.B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[28] Di Pillo, G., L. Grippo (1979), "A new class of augmented Lagrangians in nonlinear programming," *SIAM Journal on Control and Optimization*, vol. 17, pp. 618-628.

[29] Ferris, M.C., S. Lucidi (1994), "Nonmonotone stabilization methods for nonlinear equations," *Journal of Optimization Theory and Applications*, vol. 81, pp. 815-832.

[30] Ferris, M.C., T.S. Munson (2003), "Interior-point methods for massive support vector machines," *SIAM Journal on Optimization*, vol. 13, pp. 783-804.

[31] Fletcher, R. (2001), "On the Barzilai-Borwein method," *Numerical Analysis Report* NA/207.

[32] Fung, G., O.L. Mangasarian (2002), "Finite Newton method for Lagrangian support vector machine classification," *Data Mining Institute Technical Report 02-01.*

[33] Gaivoronski, A.A. (1994), "Convergence analysis of parallel backpropagation algorithm for neural networks," *Optimization Methods and Software*, vol. 4, pp. 117-134.

[34] Girosi F., T. Poggio (1989), "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, pp. 465-469.

[35] Girosi F., T. Poggio (1990), "Networks and the best approximation property," *Biological Cybernetics*, vol. 63, pp. 169-176.

[36] Girosi, F. (1993), "Regularization theory, radial basis functions and networks," *Computer and Systems Sciences*, vol. 136, pp. 166-187.

[37] Gilbert, J., C. Lemaréchal (1989), "Some numerical experiments with variable-storage quasi-Newton algorithms," *Mathematical Programming, Series B*, vol. 45, pp. 407-435.

[38] Gilbert, J., J. Nocedal (1992), "Global convergence properties of conjugate gradient methods for optimization," *SIAM J. Optimization*, vol. 2, pp. 21-42.

[39] Gori, M., A. Tesi (1992), "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 14, pp. 76-86.

[40] Grievank, A., G.F. Corliss eds. (1992), *Automatic Differentiation of Algorithms: theory, implementation and application*, SIAM, Philadelphia.

[41] Grippo, L., F. Lampariello, S. Lucidi (1986), "A nonmonotone line search technique for Newton's method," *SIAM Journal on Numerical Analysis*, vol. 23, pp. 707-716.

[42] Grippo, L., F. Lampariello, S. Lucidi (1991), "A class of nonmonotone stabilization methods in unconstrained optimization," *Numerische Mathematik*, vol. 59, pp. 779-805.

[43] Grippo, L. (1994), "A class of unconstrained minimization methods for neural network training," *Optimization Methods and Software*, vol. 4, pp. 135-150.

[44] Grippo, L., S. Lucidi (1997), "A globally convergent version of the Polak-Ribiére conjugate gradient method," *Mathematical Programming*, vol. 78, pp. 375-391.

[45] Grippo, L., M. Sciandrone (1999), "Globally convergent block-coordinate techniques for unconstrained optimization," *Optimization Methods and Software*, vol. 10, pp. 587-637.

[46] Grippo, L., M. Sciandrone (2000), "On the convergence of the block nonlinear Gauss-Seidel method under convex constraints," *Operations Research Letters*, vol. 26, pp. 587-637.

[47] Grippo, L. (2000), "Convergent on-line algorithms for supervised learning in neural networks," *IEEE Transactions on Neural Networks*, vol. 11, pp. 1284-1295.

[48] Grippo, L., M. Sciandrone (2002), "Nonmonotone globalization techniques for the Barzilai-Borwein gradient method," *Computational Optimization and Applications*, vol. 23, pp. 143-169.

[49] Gorse, D., A. Shepherd, J. Taylor (1997), "The new ERA in supervised learning," *Neural Networks*, vol. 10, pp.343-352.

[50] Hagan, M.T., M. Menhaj (1994), "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989-993.

[51] Haykin, S. (1999), *Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ.

[52] Hestnes, M.R. (1980), *Conjugate Direction Methods in Optimization*, Springer-Verlag, New York.

[53] Hestnes, M.R., E. Stiefel (1952), "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 29, pp. 409-439.

[54] Hastie, T., R. Tibshirani, J. Friedman (2001), *The Elements of Statistical Learning. Data mining, Inference and Prediction*, Springer, New York.

[55] Joachims, T. (1998), "Making large-scale SVM learning practical," in: B. Schiölkopf, C. Burges and A. Smola, eds., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, MA.

[56] Keety, S.S., E.G. Gilbert (2002), "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, vol. 46, pp. 351-360.

[57] Kirkpatrick, S., C.D. Gelatt, M.P. Vecchi (1983), "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680.

[58] Kurková, V. (1992), "Kolmogorov's theorem is relevant," *Neural Computation*, vol. 3, pp. 617-622.

[59] Kurková, V., M. Sanguineti (2002), "Comparison of worst case errors in linear and neural network approximation," *IEEE Transactions on Information Theory*, vol. 48, pp. 264-275.

[60] Lampariello, F., M. Sciandrone (2001), "Use of the "minimum norm" search direction in a nonmonotone version of the Gauss-Newton method," in corso di pubblicazione su *Journal of Optimazion Theory and Applications*.

[61] Lin, C.J. (2001), "On the convergence of the decomposition method for support vector machines," *IEEE Transactions on Neural Networks*, vol. 12, pp. 1288-1298.

[62] Lin, C.J. (2002), "Asymptotic convergence of an SMO algorithm without any assumptions," *IEEE Transactions on Neural Networks*, vol. 13, pp. 248-250.

[63] Lisboa, P.J.G., S.J. Perantonis (1991), "Complete solution of the local minima in the XOR problem," *Network*, vol. 2, pp. 119-124.

[64] Liu, D.C., J. Nocedal (1989), "On the limited-memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503-528.

[65] Lorentz, G.G., M. von Golitschek, Y. Makovoz. (1996), *Constructive Approximation: Advanced Problems*, Vol. 304, Springer, Berlin.

[66] Luo, Z.Q., P. Tseng (1994), "Analysis of an approximate gradient projection method with applications to the backpropagation algorithm," *Optimization Methods and Software*, vol. 4, pp. 85-102.

[67] Maiorov, V., A. Pinkus (1999), "Lower bounds for approximating by MLP neural networks," *Neurocomputing*, vol. 25, pp. 81-91.

[68] Mangasarian, O.L. (1994), *Nonlinear Programming*, SIAM, Philadelphia.

[69] Mangasarian, O.L. (1977), "Solution of symmetric linear complementarity problems by iterative methods," *Journal of Optimization Theory and Applications*, vol. 22, pp. 465-485.

[70] Mangasarian, O.L., M.V. Solodov (1994), "Serial and parallel backpropagation convergence via nonmonotone perturbed minimization," *Optimization Methods and Software*, vol. 4, pp. 85-102.

[71] Mangasarian, O.L., D.R. Musicant (1999), "Successive overrelaxation for support vector machines," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1032-1037.

[72] McCulloch, W.S., W. Pitts (1943), "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.

[73] Mercer, J. (1909), "Functions of positive and negative type, and their connection with the theory of integral equations," *Transactions of the London Philosophical Society (A)*, vol. 209, pp. 415-446.

[74] Micchelli, C.A. (1986), "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Constructive Approximation*, vol. 2, pp. 11-22.

[75] Minsky, M.L., S.A. Papert (1969), *Perceptrons*, MIT Press, Cambridge, MA.

[76] Møller, M. (1993), "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533.

[77] Motzkin, T.S., I.J. Schoenberg (1954), "The relaxation method for linear inequalities," *Canadian Journal of Mathematics*, vol. 6, pp. 393-404.

[78] Nocedal, J. (1980), "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation*, vol. 35, pp.773-782.

[79] Ortega, J.M., W.C. Rheinboldt (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.

[80] Palagi, L., M. Sciandrone (2002), "On the convergence of a modified version of SVM^{light} algorithm," Tech. Rep. IASI-CNR.

[81] Pinkus, A. (1996), "TDI-Subspaces of $C(R^d)$ and some density problems from neural networks," *Journal of Approximation Theory*, vol. 85, pp. 269-287.

[82] Pinkus, A. (1999), "Approximation theory of the MLP model in neural networks," *Acta Numerica*, pp. 143-195.

[83] Plagianakos, V. P., G.D. Magoulas, M.N. Vrahatis (2003), "Deterministic nonmonotone strategies for effective training of multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 13, pp. 1268-1284.

[84] Platt, J.C. (1998), "Fast training of support vector machines using sequential minimal optimization," in: B. Schölkopf, C. Burges and A. Smola, eds., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, MA.

[85] Poggio, T., F. Girosi (1990), "Networks for approximation and learning," *Proceedings of the IEEE*, VOL. 78, pp. 1481-1497.

[86] Polak, E., G. Ribière" (1969), "Note sur la convergence de méthodes de directions conjuguées," *Revue Francaise d'Informatique et de Recherche Opérationnelle*, vol. 16, pp. 35-43.

[87] Polyak, B.T. (1964), "Some methods of speeding up the convergence of iteration methods", *U.S.S.R. Comput. Maths. Math. Phys.*, vol. 4, pp. 1-17.

[88] Powell, M.J.D. (1973), "On search directions for minimization algorithms," *Mathematical Programming*, vol. 4, pp. 193-201.

[89] Powell, M.J.D. (1992), "The theory of radial basis function approximation in 1990," in W. Light, ed., *Advances in Numerical Analysis Vol. II: Wavelets, Subdivision Algorithms and Radial Basis Functions*, Oxford Science Publications, Oxford.

[90] Raydan, M. (1993), "On the Barzilai and Borwein choice of the steplength for the gradient method," *IMA Journal on Numerical Analysis*, vol. 13, pp. 618-622.

[91] Raydan, M. (1997), "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem," *SIAM Journal on Optimization*, vol. 7, pp. 26-33.

[92] Rosenblatt, F. (1962), *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington, DC.

[93] Schrijver, A. (1986), *Theory of Linear and Integer Programming*, Wiley, New York.

[94] Rumelhart, D.E., G.E. Hinton, R.J. Williams (1986), "Learning representations of back-propagation errors," *Nature (London)*, vol. 323, pp. 533-536.

[95] Rumelhart, D.E., J.L. McClelland, eds. (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol.1, MIT Press, Cambridge, MA.

[96] Saad, D. editor (1998), *On-line Learning in Neural Networks*, Cambridge University Press, Cambridge.

[97] Scarselli, F., A.C. Tsoi (1998), "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results," *Neural Networks*, vol. 11, pp. 15-37.

[98] Sharda, R.,J. Wang (1996), "Neural networks and operations research/management science," *European Journal of Operational Research*, vol. 93, pp. 227-229

[99] Shepherd, A., (1997), *Second-Order Methods for Neural Networks*, Springer-Verlag, New York.

[100] Sprecher, D.A. (1993), "A universal mapping for Kolmogorov's superpositions theorem," *Neural Networks*, vol. 6, pp. 1089-1094.

[101] Tikhonov, A.N., V.Y. Arsenin (1977), *Solutions of Ill-Posed Problems*, DC: V.H. Winston, Washington.

[102] Tseng, P. (1998), "Incremental gradient (-projection) method with momentum term and adaptive stepsize rule," *SIAM Journal on Optimization*, vol. 8, pp. 506-531.

[103] Vapnik, V.N., A.Y. Chervonenkis (1971), "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability and its Applications*, vol. 6, pp. 264-280.

[104] Vapnik, V.N. (1995), *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

[105] Vapnik, V.N. (1998), *Statistical Learning Theory*, Wiley, New York.

[106] Vogl,T.P., J.K.Mangis, A.K.Rigler, W.T.Zink and D.L.Alkon (1988), "Accelerating the convergence of the back-propagation method," *Biological Cybernetics*, vol. 59, pp. 257-263.

[107] Wettschereck, D., T. Dietterich (1992), "Improving the performance of radial basis function networks by learning locations," in J.E. Moody, S.J. Hanson, R.P. Lippman (Eds.), *Advances in neural information processing systems*, vol. 4, pp. 1133-1140.

[108] White, H. (1989), "Some asymptotic results for learning in single hidden-layer feedforward network models," *J. Amer. Statist. Assoc.*, vol. 84, pp. 117-134.

[109] White, H. (1992) *Artificial Neural Networks*, Blackwell, Oxford, UK.

[110] Wilson, R.L., R.Sharda (1992), "Neural networks," *OR/MS Today*, vol. 19, pp. 36-42.

[111] Zhang, X.-S. (2000), *Neural Networks in Optimization*, Kluwer Academic Publishers, Norwell, USA.

[112] Zhang, J.Z., L.H. Chen (1997), "Nonmonotone Levenberg-Marquardt algorithms and their convergence analysis," *Journal of Optimization Theory and Applications*, vol. 92, pp. 393-418.