Reti Neuronali

Gabriele Filosofi

Settembre 2006

1 Reti neurali

Le reti neurali artificiali (**Artificial Neural Networks**) sono modelli matematici ad architettura distribuita, formate da molte unità interconnesse, dotate o meno di memoria, con parametri scalari continui associati a ciascuna connessione. I parametri possono essere modificati durante una fase di *apprendimento*, in modo tale che la rete abbia il comportamento desiderato. Consideriamo NNs con evoluzione a tempo discreto. Adattività, elaborazione nonlineare ed elevato parallelismo sono le caratteristiche peculiari e desiderabili delle NNs.

1.1 Principali tipologie di unità

Fra i tipi di unità maggiormante usate:

unità binarie a soglia L'uscita è pari a 1 se il potenziale di attivazione è maggiore o uguale a una soglia, altrimenti è 0,

$$P(t) = \sum_{i=1}^{N} w_i x_i(t) \tag{1.1.1}$$

$$u(t+1) = stepf(P(t) - \theta) \tag{1.1.2}$$

In MATCAD la funzione $stepf(\bullet)$ è chiamata $hardlim(\bullet)$. Sono reti binarie tutte quelle con uscita delle unità a 2 valori. Le rappresentazioni $u \in \{0, 1\}$ e $v \in \{-1, 1\}$ sono equivalenti, essendo legate dalla trasformazione $u = \frac{1}{2}(v + 1)$.

unità booleane Sia gli ingressi che l'uscita sono variabili booleane. L'uscita è una funzione booleana degli n ingressi

unità con uscita lineare a soglia

$$u(t+1) = k[P(t) - \theta] stepf[P(t) - \theta] \tag{1.1.3}$$

In MATCAD potremo utilizzare $(\bullet) * hardlim(\bullet)$.

unità con uscita sigmoidale

$$u(t+1) = \frac{1}{1+e^{-[P(t)-\theta]}} \tag{1.1.4}$$

oppure

$$u(t+1) = \frac{1}{2} + \frac{1}{\pi}\arctan[P(t)-\theta] \tag{1.1.5}$$

In MATCAD la funzione $\frac{1}{1+e^{-\bullet}}$ è chiamata $logsig(\bullet)$.

unità sigma-pi ($\Sigma\Pi$) Il potenziale è una somma di prodotti degli ingressi. P.es.

$$u(t+1) = w_1 x_1(t) x_2(t) + w_2 x_3(t) x_4(t) \tag{1.1.6}$$

unità con legge di attivazione probabilistica L'uscita, binaria, ha distribuzione di probabilità

$$Pr\{u(t+1)=1\} = \frac{1}{1+e^{-k[P(t)-\theta]}} \tag{1.1.7}$$

Per utilizzare queste unità occorre disporre di un generatore di numeri casuali con distribuzione uniforme in $[0,1]$.

1.2 Principali tipologie di rete

Fra i tipi di reti maggiormante usate:

reti prive di localizzazione Non si tiene conto della distanza tra le varie unità. Le unità sono contrassegnate da un indice. E' definita la matrice delle connessioni $\mathbf{W} = [w_{ij}]$, dove w_{ij} è il coefficiente tra l'uscita dell'unità j e l'ingresso dell'unità i. La rete è **simmetrica** $\Leftrightarrow$ $\mathbf{W}$ è simmetrica ($w_{ij} = w_{ji}$). La rete è **antisimmetrica** $\Leftrightarrow$ $\mathbf{W}$ è antisimmetrica ($w_{ij} = -w_{ji}$). Il potenziale di attivazione dell'unità i-esima è la somma pesata delle uscite delle N unità afferenti:

$$P_i(t) = \sum_{j=1}^{N} w_{ij} x_i(t) \tag{1.2.1}$$

reti localizzate Si tiene conto della distanza d_{ij} tra le unità i-esima e j-esima

$$w_{ij} = w_{ij}^0 f(d_{ij}) \tag{1.2.2}$$

dove potremmo avere

$$f(d_{ij}) = e^{-\frac{d_{ij}}{\lambda}}$$

$$f(d_{ij}) = (1 - \frac{d_{ij}^2}{2\sigma^2})e^{-\frac{d_{ij}^2}{2\sigma^2}}$$

Se $w_{ij}^0 > 0$, la funzione a cappello messicano trasforma in inibitorie le connessioni nell'intorno $\sqrt{2}\sigma < |d_{ij}| < 2\sigma$.

reti con connessioni di ordine superiore al primo Sono le reti formate da unità $\Sigma\Pi$. L'**ordine** della rete, n, è il numero massimo di ingressi a moltiplicare che un'unità può ricevere. La descrizione usa le n matrici generalizzate

$$\mathbf{W}_1 = [w_{ij}^1], \mathbf{W}_2 = [w_{ijk}^2], \mathbf{W}_n = [w_{ij_1 \ldots j_n}^n]$$

dove il primo indice indica l'unità ricevente, mentre gli altri n indici individuano le unità il prodotto delle cui uscite eccita l'unità in questione.

Le architetture a strati (*lineari*, 2D o 3D) si dividono in **monostrato** e **pluristrato**. Le connessioni **intra-strato** (o connessioni *laterali*) collegano unità di uno stesso strato, le connessioni **inter-strato** collegano unità appartenenti a strati differenti. Le connessioni interstrato si dividono in **feedforward** e **feedback** (o *ricorrenti*) a seconda della direzione.

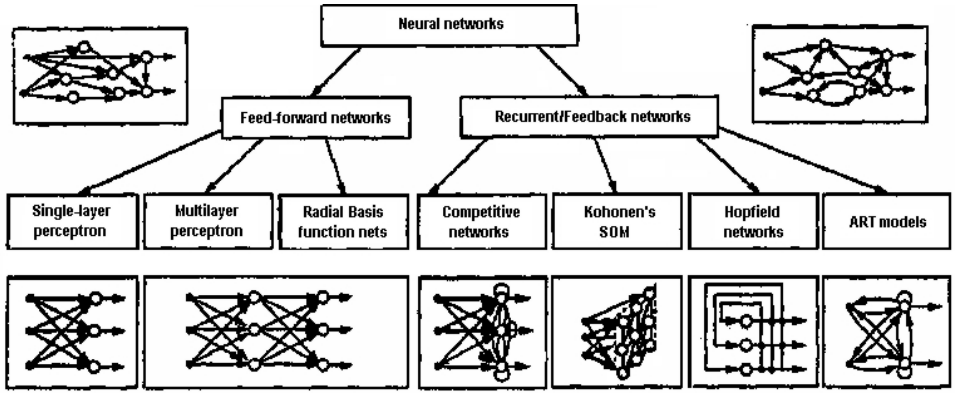

Figure 1: Tassonomia di Jain (1997).

Una suddivisione dei vari modelli di NN è quella tra reti *feedforward* (esiste un legame funzionale tra ingresso e uscita, ingressi diversi producono uscite diverse) e **Attractor NNs** (l'output è l'attrattore raggiunto, ingressi diversi possono convergere sullo stesso attrattore). Nel primo gruppo gli associatori lineari, il perceptrone multiplo, nel secondo gruppo le reti booleane, di Caianiello (1961), Amari (1972), Grossberg (1969), Little (1974) , Hopfield (1982).

Una unità è di **input** se non riceve retroazioni dalle altre unità della rete. Una unità è di **output** se la sua uscita non è collegata alle altre unità.

Talvolta si presentano reti ad attrattori, dove tutte le unità sono sia di ingresso che di uscita. Ma in generale la rete ha M unità di ingresso, N unità di uscita e H unità nascoste. Quindi, per reti binarie, il numero di stati di ingresso è 2^M, quello di uscita 2^N, mentre gli stati interni sono 2^H.

1.3 Principali tipologie di dinamica

La legge di evoluzione della rete tra i suoi stati interni può essere

- a **dinamica parallela** (o *sincrona*, o *di Little*) se tutte le unità aggiornano l'uscita negli stessi istanti
- a **dinamica sequenziale** (o *asincrona*, o *di Hopfield*) se a ogni istante solo una unità scelta a caso aggiorna l'uscita. Un'alternativa è stabilire un ordine fisso con cui le unità vengono aggiornate.

1.4 Reti binarie di McCulloch e Pitts

Le reti di McCulloch e Pitts sono reti completamente connesse di unità binarie a soglia. Nella versione iniziale, le singole unità di McCulloch e Pitts (**neuroni formali**) avevano solo uno o due ingressi.

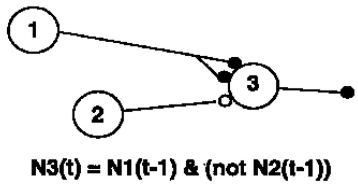

Figure 2: Modello di McCulloch e Pitts (1943).

I valori dei pesi delle connessioni potevano assumere valori sia positivi che negativi. Per una rete con un numero finito di unità di ingresso e una sola unità di uscita si poterono dimostrare i seguenti teoremi:

Teorema I. *Qualunque funzione booleana è equivalente a una rete di McCulloch-Pitts*

P.es. un OR si ottiene con una sola unità binaria avente soglia $\theta = \frac{1}{2}$ e due ingressi con peso $w_1 = w_2 = 1$. L'OR esclusivo (xor) è realizzato in tre passi dalla rete seguente

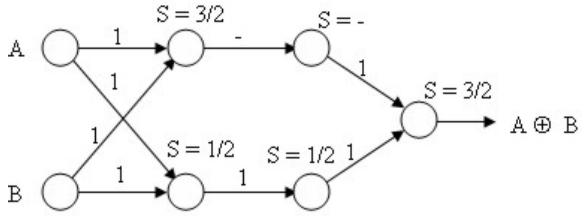

Figure 3: Rete di McCulloch e Pitts che realizza la funzione booleana XOR.

Il libro "Perceptrons" di M.Minsky e S.Papert (1969) dimostrava che non è possibile risolvere il problema dello XOR in un solo passo. In realtà è possibile implementare qualunque funzione booleana, pur di avere a disposizione un tempo sufficientemente lungo.

Teorema II. *La classe dei FA (automi finiti) è equivalente alla classe delle reti di McCulloch-Pitts*

1.5 Dinamica

Supponiamo di avere un'unità di ingresso e un'unità di uscita collegate a ciascuna delle N unità nascoste. Quello che era un trasduttore lineare è diventato un sistema dinamico complesso. Applichiamo inoltre le ipotesi

- Coefficienti di connessione sia positivi che negativi
- Unità completamente connesse

Applicato uno stato iniziale $\mathbf{x}_0$ in t_0, l'evoluzione libera della rete (cioè con ingressi nulli) può essere di tre tipi

- terminante in uno stato di equilibrio, o stato stazionario, o punto fisso $\Leftrightarrow \quad \exists t_{tr} \quad | \quad per \quad t > t_{tr} \quad \mathbf{x}(t) = cost.$
- terminante in un ciclo $\Leftrightarrow \quad \exists t_{tr}, \tau \quad | \quad per \quad t > t_{tr} \quad \mathbf{x}(t+\tau) = \mathbf{x}(t)$. τ è la **durata del ciclo**, mentre t_{tr} è la **durata del transitorio**
- Caotico

Il problema principale è sapere sotto quali condizioni è possibile associare a uno stato iniziale una evoluzione libera tra i tipi citati.

Teorema III. *L'evoluzione libera di una rete binaria sincrona termina sempre in un ciclo o in uno stato di equilibrio*

Dim. La rete ha un numero finito di stati, quindi uno stato prima o poi si ripresenta. Dato che la rete è deterministica, c.v.d.

Teorema IV. *L'evoluzione libera di una rete binaria asincrona può essere indefinitamente caotica*

I ricercatori hanno cercato le condizioni cui deve sottostare una rete asincrona per avere stati di equilibrio

Teorema V. *Data una rete binaria asincrona e una funzione univoca $E(\mathbf{x})$ definita sui suoi stati, tale che*

$$\forall t, \quad E(\mathbf{x}(t+1)) \leq E(\mathbf{x}(t))$$

allora l'evoluzione libera della rete termina in uno stato di equilibrio, punto di minimo per E.

Dim. Il numero di stati è finito, e così pure i valori assunti da E... c.v.d.
E è detta **energia** o **funzione di Ljapunov**.

Teorema VI. *Data una rete con unità binarie a soglia (sia $\mathbf{x} = 1$), asincrona simmetrica priva di autoconnessioni, con potenziali di attivazione mai nulli ($P_i(t) - \theta_i \neq 0$), ha un'evoluzione libera che termina in uno stato di equilibrio, e una possibile funzione di Ljapunov è*

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{i,j=1}^{N} w_{ij} x_i x_j + \sum_{k=1}^{N} \theta_k x_k$$

dove θ_k è la soglia dell'unità k-esima.

Teorema VII (o primo teorema di Goles). *Data una rete con unità binarie a soglia, sincrona simmetrica priva di autoconnessioni, con potenziali di attivazione mai nulli, ha un'evoluzione libera che termina in uno stato di equilibrio o in un ciclo di lunghezza 2. In particolare, se $\theta_k = 0$, una possibile funzione di Ljapunov è*

$$E(\mathbf{x}) = -\sum_{i=1}^{N} | \sum_{j=1, j\neq i}^{N} w_{ij} x_j |$$

Teorema VIII (o secondo teorema di Goles). *Data una rete con unità binarie a soglia, sincrona antisimmetrica, con potenziali di attivazione mai nulli, ha un'evoluzione libera che termina sempre in un ciclo di lunghezza 4.*

Tra gli aspetti che restano da chiarire, la durata dei transitori, la relazione tra stato iniziale e evoluzione libera, lunghezza media dei cicli.

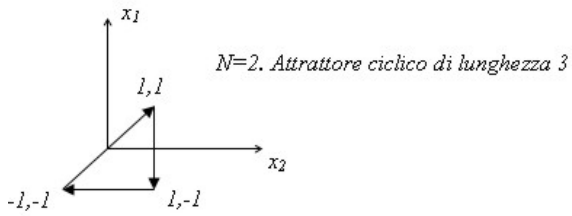

Figure 4: Attrattore ciclico di lunghezza 3.

1.6 Perceptrone

Nel 1962 F.Rosenblatt definisce il **Perceptrone**, estensione naturale del *neurone formale* di McCullocs e Pitts. Sostanzialmente viene aumentato il numero di ingressi delle unità, da uno o due a un numero arbitrario.

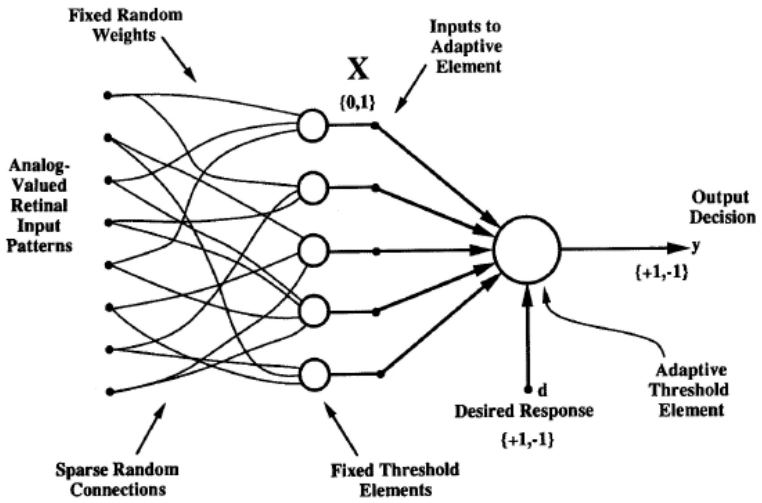

Figure 5: α-Perceptrone di Rosenblatt (1962).

Il Perceptrone è un combinatore lineare adattativo con una funzione binaria tipo $stepf()$ applicata sull'uscita. E' capace di apprendere sulla base del confronto tra uscita e risposta attesa fornita da un supervisore.

Le reti feedforward formate da perceptroni, con funzione di classificazione, sono reti con uno strato di ingresso, a M unità fisse, e uno strato di uscita, a N unità adattative. Il fatto di limitare a uno il numero di strati effettivi era legato al seguente teorema:

Teorema IX. *Un Perceptrone multistrato con sole unità lineari ha la stesse capacità di classificazione di un perceptrone con un solo strato (escludendo quello*

di ingresso).

1.6.1 Algoritmo di addestramento di Rosenblatt (1962)

La fase di apprendimento consiste nei seguenti passi:

1. assegnazione di valori piccoli e casuali ai coefficienti w_{ij}
2. scelta dell'errore ammissibile sull'insieme dei pattern di addestramento $\mathbf{x}^1, \ldots, \mathbf{x}^R$ (*training set*), E_{max}
3. errore effettivo sul training set $E = 0$
4. indice pattern di addestramento $s = 1$
5. si forzano le uscite delle unità di ingresso al pattern $\mathbf{x}^s$
6. si calcolano le uscite della rete

$$u_i^s = stepf(\sum_{i=1}^{N} w_{ij} x_j^s - \theta_i)$$

7. si somma a E l'errore sul pattern $\mathbf{x}^s$,

$$\Delta E = \frac{1}{R} |\mathbf{u}^s - \mathbf{t}^s|^2 = \frac{1}{R} \sum_{i=1}^{N} (u_i^s - t_i^s)^2$$

8. si correggono i coefficienti di connessione con

$$\Delta w_{ij} = \beta x_j^s (t_i^s - u_i^s)$$

   dove $\beta > 0$

9. se $s < R$, $s = s + 1$ e torna a 5)
10. se $E > E_{max}$ torna a 3)
11. fine

I passi compresi tra 3) e 9) sono detti **epoca** (ciclo di presentazione di tutto il training set). L'algoritmo converge verso una configurazione della rete che classifica tutti i pattern del training set nei limiti dell'errore prefissato, ma solo se i pattern sono linearmete separabili nello spazio di ingresso (*Teorema di convergenza del Preceptrone*).

1.6.2 Algoritmo di addestramento LMS di B.Widrow e M.E.Hoff (1961)

B.Widrow e M.E.Hoff generalizzarono l'algoritmo di addestramento ai Perceptroni, con un solo strato adattativo, aventi funzione di attivazione di tipo qualunque ($u_i = f(P_i)$). Siccome l'algoritmo prevede il calcolo di derivate, l'uscita delle unità deve essere una funzione continua e derivabile del potenziale, a valori in un range prefissato (supponiamo l'intervallo $[0, 1]$). L'errore su tutto il *training set* è l'errore quadratico medio (MSE)

$$E = \sum_{s=1}^{R} E^s = \frac{1}{2} \sum_{s=1}^{R} |\mathbf{u}^s - \mathbf{t}^s|^2 = \frac{1}{2} \sum_{s=1}^{R} \sum_{i=1}^{N} (u_i^s - t_i^s)^2$$

La regola di aggiornamento dei coefficienti di connessione somma ai coefficienti correnti la quantità

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

dove η è un **parametro di apprendimento**, positivo e di solito < 1. Questa regola esprime il metodo della **discesa lungo il gradiente** (*gradient descent*, o *steepest descent*). La precedente può essere esplicitata nel modo seguente

$$\Delta w_{ij} = -\eta \sum_{s=1}^{R} (u_i^s - t_i^s) f'(\sum_{j=1}^{M} w_{ij} x_j^s - \theta_i) x_j^s$$

A seconda della forma di $f(\bullet)$ avremo diversi casi:

- funzione di trasferimento sigmoidale

$$f(P) = \frac{1}{1 + e^{-P}}$$

$$f'(P) = f(P)[1 - f(P)]$$

- funzione di trasferimento lineare

$$f(P) = \alpha P$$

$$f'(P) = \alpha$$

- funzione di trasferimento iperbolica

$$f(P) = \tanh(P)$$

$$f'(P) = 1 - f(P)^2$$

Il problema di questo algoritmo è che non garantisce la convergenza a un minimo assoluto di $E(\mathbf{W})$ per qualsiasi assegnazione iniziale dei coefficienti $\mathbf{W}$ e del *training set*. Il sistema può convergere infatti su un qualsiasi minimo locale, fornendo una prestazione non ottima.

La regola di Widrow-Hoff prevede il calcolo dell'errore globale E su tutto il training set prima di ogni aggiornamento di **W** (modalità **batch** o **cumulativa**). E' ammesso pure il caso in cui l'aggiornamento viene fatto sull'errore relativo a ciascun pattern E^s (modalità **on-line**). L'addestramento on-line è intrinsecamente più randomico, e questo può aiutare a superare dei minimi locali. Ma perchè ciò avvenga è conveniente randomizzare l'ordine di presentazione dei pattern da epoca a epoca.

L'algoritmo che generalizza la regola LMS (o *Delta Rule*) di Widrow-Hoff a Perceptroni multistrato fu trovato da Rumelhart, Hinton e Williams, ed è noto col nome di *error backpropagation*.

1.7 MLP e Error Backpropagation (1985)

Nel 1969 Minsky and Papert misero in evidenza questa fondamentale limitazione del Perceptrone, ma clamorosamente affermarono nel loro libro: *The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension to multi-layer systems is sterile.* Ma questa conclusione si rivelò affrettata. Nel 1985 D.E.Rumelhart, G.E.Hinton e R.J.Williams introducono una nuova regola di apprendimento supervisionato, detta **Error Back-propagation**, da applicare a **Perceproni multistrato** (*Multi Layer Perceptrons*, MLP). Queste reti *feedforward* con strati intermedi di cosiddette unità nascoste (*hidden units*) superano il vincolo della separabilità lineare dei pattern di addestramento nello spazio di ingresso. Un MLP risolve il problema di classificazione suddividendo lo spazio di ingresso con N iperpiani, uno per ciascuna unità di uscita.

Ricordiamo che quando si parla di rete neuronale a X strati, si intende escluso lo strato di ingresso. Infatti lo strato di ingresso è formato da unità fisse, cioè senza parametri adattativi. La potenza del MLP è evidenziata dal seguente risultato:

Teorema X (di Kolmogorov). *Data una funzione continua* $F : [0,1]^M \rightarrow \Re^N$*, con un numero finito di discontinuità,* F *puo essere implementata esattamente da un Perceptrone feedforward a tre strati, con* M *unità nello strato di ingresso,* $H = 2M + 1$ *unità nello strato nascosto, e* N *unità nello strato di uscita. Alcune delle unità devono essere però non lineari, per esempio quelle di uno strato.*

Siano:

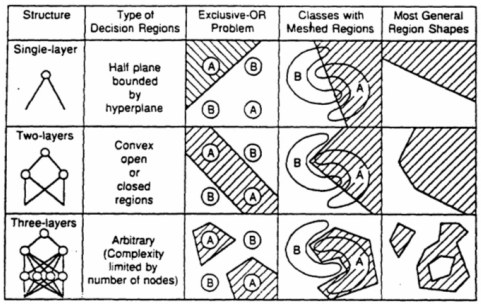

Figure 6: Capacità di classificazione del MLP (Lippman, 1987).

e_r^s	uscite dello strato di ingresso (durante la presentazione del pattern s-esimo)
c_{jr}	coefficienti di connessione tra strato di ingresso e strato nascosto
x_j^s	uscite dello strato nascosto
w_{ij}	coefficienti di connessione tra strato nascosto e strato di uscita
u_i^s	uscite dello strato di uscita
t_i^s	uscita desiderata
θ_j, θ_i	soglie
$f(\bullet)$	funzione di trasferimento valida per tutte le unità (escluse quelle di ingresso)

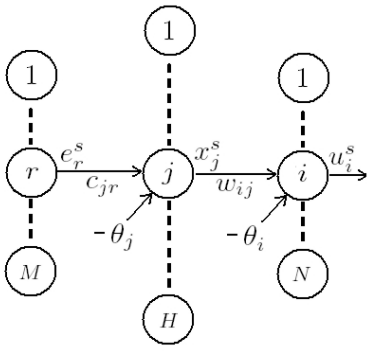

Figure 7: Backpropagation sul MLP con due strati.

Al termine della generica epoca della fase di apprendimento, anche i coefficienti c_{jr} devono essere corretti con

$$\Delta c_{jr} = -\eta \frac{\partial E}{\partial c_{jr}} = -\eta \sum_{s=1}^{R} (x_j^s - \tilde{x}_j^s) f'(\sum_{r=1}^{M} c_{jr} e_r^s - \theta_j) e_r^s$$

dove $\tilde{\mathbf{x}}^s$ è la risposta desiderata delle unità nascoste. Rumelhart, Hinton e Williams hanno dimostrato che questa quantità può essere espressa in funzione di $\mathbf{x}^s$, $\mathbf{u}^s$, $\mathbf{t}^s$ e $\mathbf{W}$ nel seguente modo:

$$\tilde{x}_j = \sum_{i=1}^{N}(u_i^s - t_i^s) f'(\sum_{k=1}^{H} w_{ik} x_k^s - \theta_i) w_{ij}$$

In generale, con h strati, q^n unità nello strato n-esimo ($n = 1, \ldots, h$), $\mathbf{W}^n$ matrice $q^n \times q^{n-1}$ delle connessioni tra strato $n-1$ e n-esimo, $\mathbf{x}^{n,s}$ vettore delle uscite dello strato n-esimo in corrispondenza dell's-esimo pattern del training set, $\mathbf{P}^n$ vettore dei potenziali di attivazione delle unità dello strato n-esimo, θ^n vettore delle soglie delle unità dello strato n-esimo,

$$\Delta w_{ij}^n = -\eta \sum_{s=1}^{R} \delta_i^{n,s} x_j^{n-1,s}$$

dove

$$\delta_i^{n,s} = \begin{cases} f'(P_i^{h,s})(u_i^s - t_i^s) & i = 1, \ldots, q^h \quad se \quad n = h \\ f'(P_i^{n,s}) \sum_{r=1}^{q^{n+1}} \delta_r^{n+1,s} w_{ri}^{n+1} & i = 1, \ldots, q^n \quad se \quad n < h \end{cases}$$

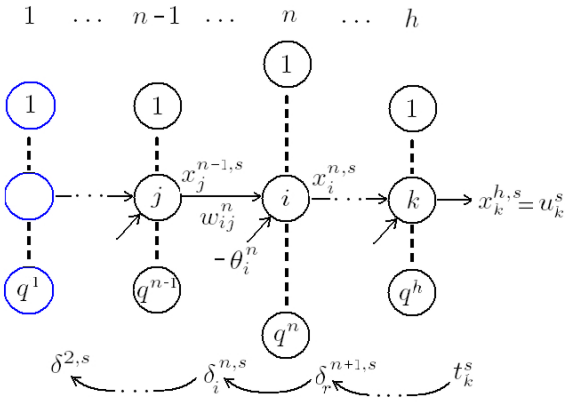

Figure 8: Backpropagation sul MLP con $h-1$ strati. Lo strato di ingresso non viene contato perchè è formato da unità fisse, cioè senza parametri adattativi.

Nella generica epoca di apprendimento, si parte dal calcolo di $\delta^{h,s}$, con una legge che è identica a quella di Widrow-Hoff. Poi si calcola ricorsivamente $\delta^{h-1,s}$,$\delta^{h-2,s}$,.. fino a $\delta^{2,s}$. Qui ci si arresta, in quanto lo strato di ingresso ($n = 1$) non ha altri ingressi. Quindi si passa ad aggiornare i coefficienti di connessione. L'algoritmo di Backpropagation è euristico, nel senso che non garantisce la convergenza dell'errore su un minimo, nè raggiunto un minimo è garantito che questo sia un minimo globale.

1.7.1 Varianti

Per aumetare la velocità di convergenza della rete in fase di apprendimento si può scegliere la modalità **on-line** invece che la modalità **batch**. Un'altra tecnica è aggiungere un **momento** nella legge di aggiornamento dei coefficienti:

$$\Delta w_{ij}^n = -\eta \sum_{s=1}^{R} \delta_i^{n,s} x_j^{n-1,s} + \alpha (w_{ij}^n)'$$

dove $(\bullet)'$ sta a indicare la quantità $\bullet$ relativa all'epoca precedente. Tipicamente $\alpha = 0$ e η piccolo (maggiore tempo di apprendimento e minore uso di memoria), oppure α e η abbastanza grandi (p.es. $\alpha = 0.9$ e $\eta = 0.6$).

Il metodo della backpropagation si può usare anche per modificare le soglie delle unità,

$$\Delta \theta_i = -\eta \frac{\partial E}{\partial \theta_i}$$

Si arriva alla

$$\Delta \theta_i^n = -\eta \sum_{s=1}^{R} \delta_i^{n,s}$$

Nella variante **quickpropagation** di Fahlman (1988) il tasso di variazione dell'errore quadratico globale rispetto ai coefficienti di connessione è ancora

$$E = \sum_{s=1}^{R} \delta_i^{n,s} x_j^{n-1,s}$$

ma stavolta, al termine di ogni epoca, si devono memorizzare tre quantità:

$$\frac{\partial E}{\partial w_{ij}^n}, \quad \left(\frac{\partial E}{\partial w_{ij}^n}\right)', \quad (\Delta w_{ij}^n)'$$

che definiscono una parabola nel piano E, w_{ij}^n, nell'ipotesi che E sia una funzione approssimativamente quadratica rispetto a ciascun coefficiente (nel caso di unità lineari lo sarebbe esattamente). L'idea è allora far variare w_{ij}^n in modo che vada a coincidere col vertice di questa parabola (minimo):

$$\Delta w_{ij}^n = \frac{(\Delta w_{ij}^n)'}{1 - \frac{\frac{\partial E}{\partial w_{ij}^n}}{(\frac{\partial E}{\partial w_{ij}^n})'}}$$

Nelle applicazioni pratiche occorre introdurre dei correttivi per evitare che la quantità a secondo membro diverga quando il denominatore è molto piccolo (parabola con piccolissima curvatura). La quickpropagation si dimostra molto più veloce della backpropagation, ma è anche molto più soggetta a convergere sui minimi locali di E.

Per evitare quest'ultimo inconveniente si usa una funzione di trasferimento con un parametro di temperatura T, tipo

$$f(P) = \frac{1}{1 + e^{-\frac{P}{T}}}$$

che per $T = 0$ diventa la funzione gradino $f(P) = stepf(P)$, mentre per $T = 1$ dà la sigmoide $f(P) = \frac{1}{1+e^{-P}}$. Tipicamente si fa variare T nel tempo secondo una determinata legge, come ad esempio

$$T(t) = T_0 - \frac{(T_0 - 1)(t - t_0)}{t_M - t_0}$$

Si tratta di un "raffreddamento", dove t_M è il numero di passi necessario affiché T raggiunga valore 1. Le **macchine di Boltzmann**, introdotte da Sejnowski, Kienker e Hinton (1986), hanno legge di attivazione probabilistica, con una funzione di temperatura detta **ricottura simulata** (*simulated annealing*) che permette di raggiungere sempre il minimo assoluto di E con l'ausilio di un supervisore.

1.7.2 Applicazioni

Una importante applicazione è la compressione dei dati. Si utilizza un singolo strato nascosto avente M unità, mentre gli strati di ingresso e di uscita hanno $N > M$ unità. L'addestramento prevede che i vettori target coincidano con i pattern di ingresso. Infine la rete assocerà a ciascun pattern una rappresentazione interna a dimensionalità inferiore (il vettore di attivazione delle unità nascoste).

1.8 Reti binarie ad attrattori (Attractor Neural Networks)

La **connettività** K è il numero di ingressi delle unità, di solito inferiore al numero totale di unità N.

Fu S.A.Kauffman il primo a simulare le reti booleane sincrone (1969). Egli definì **attrattore** a ogni ciclo limite o stato di equilibrio, e il relativo **bacino di attrazione** A l'insieme degli stati iniziali che portano l'evoluzione a quel ciclo. Egli poté calcolare la dimensione media dei bacini di attrazione $|A|$, e la loro forma nello spazio degli stati basandosi su una **metrica di Hemming**

$$H(\mathbf{x}^1, \mathbf{x}^2) = \sum_{i=1}^{N} x_i^1 \oplus x_i^2 = \sum_{i=1}^{N} |x_i^1 - x_i^2|$$

La complessità di una rete è

$$C = \sum_{q=1}^{Q} \frac{|A_q|}{2^N} \log \frac{|A_q|}{2^N}$$

dove Q è il numero di attrattori. Uno stato di equilibrio è **instabile** $\Leftrightarrow$ cambiando anche una sola delle sue componenti, l'evoluzione libera della rete a partire da questo stato modificato non termina nello stato di equilibrio considerato.

Per le reti a connettività totale ($K = N$) Kauffman calcolò la lunghezza media dei cicli

$$\langle \tau \rangle = 2^{\frac{N}{2}-1}$$

e il numero medio di attrattori

$$\langle Q \rangle = \frac{N}{e}$$

Per le reti a connettività $K = 2$ Kauffman trovò

$$\langle \tau \rangle = \langle Q \rangle = \sqrt{N}$$

un risultato molto diverso da quello ottenuto già per $K > 4$:

$$\langle \tau \rangle = \frac{1}{2}\left(\frac{1}{2} + 2^{-(2^K+1)}\binom{2^K}{2^{K-1}}\right)^{-\frac{N}{2}}$$

Per spiegare questo comportamento suddivise le unità in tre categorie a seconda del loro comportamento dopo che la rete è entrata in un ciclo qualsiasi:

unità stabili che mantengono l'uscita costante, $\forall \mathbf{x}^0$

unità oscillanti con uscita periodica, $\forall \mathbf{x}^0$

unità incerte con uscita costante o periodica a seconda di $\mathbf{x}^0$

Le reti con $K = 2$ sono caratterizzate da pochi attrattori, con piccoli bacini stabili, separati da larghe regioni di unità stabili. Il motivo di questo è nel fatto che la percentuale di funzioni booleane aventi variabili di ingresso forzanti (che determinano l'uscita indipendentemente dalle altre variabili di ingresso) decresce rapidamente col numero di ingressi K.(75% per K=2, 5% per K=4).

Secondo C.Langton le reti a connettività non troppo elevata ma maggiore di 2 avrebbero le capacità maggiori di reagire a stimoli esterni, cioè in modo né troppo caotico né troppo prevedibile.

Def. Altri parametri di interesse sono: l'**attività media**

$$\delta(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t)$$

la **varianza media**

$$\sigma^2(t) = \frac{1}{N}\sum_{i=1}^{N}[x_i(t) - x_i(t_0)]^2$$

la **distanza media** di Hemming tra stati consecuitvi

$$H(t) = \frac{1}{N}\sum_{i=1}^{N}|x_i(t) - x_i(t-1)| = \frac{1}{N}H(\mathbf{x}(t), \mathbf{x}(t-1))$$

1.8.1 Reti di Caianiello (1961)

Una **rete di Caianiello** (1961) è una rete binaria di McCulloch-Pitts in cui il potenziale di attivazione di ogni unità dipende non solo dai valori degli ingressi nell'istante considerato, ma anche dai valori assunti in istanti precedenti, cioè

$$x_i(t+1) = stepf(P_i(t) - \theta_i)$$

$$P_i(t) = \sum_{j=1}^{N}\sum_{r=0}^{L} w_{ij}^r x_j(t-r)$$

Le equazioni sopra riportate furono chiamate da Caianiello *equazioni neuroniche*, accanto alle quali egli pose le *equazioni mnemoniche*, che descrivono la dinamica dei coefficienti w_{ij}^r nei processi di apprendimento, e che evolverebbero con costanti di tempo più lunghe rispetto alle equazioni neuroniche (ipotesi dell'apprendimento adiabatico).

Una **memoria associativa** è un sistema capace di immagazzinare dei pattern di informazione e, una volta stimolato con un pattern di ingresso (*probe*), emette uno dei pattern memorizzati. Di solito di tratta di *content addressable memory* (CAM), come nel modello di Hintzman. Distinguiamo memorie **autoassociative** (il probe è una copia deteriorata di uno dei pattern immagazzinati, e il sistema deve recuperare il pattern prototipo) e memorie **eteroassociative**. Generalmente ciascun pattern memorizzato corrisponde a un attrattore per una qualche funzione di energia. Diversamente da altri tipi di NN (Feedforward, RBF networks, Kohonen's SOM, ecc.) le NN ad attrattori possono essere addestrate con un procedimento non iterativo, quindi assai più veloce.

1.8.2 Modello di Hopfield

Una **rete di Hopfield** è totalmente connessa, simmetrica e priva di autoconnessioni. Le unità sono binarie ($x = \pm 1$) a soglia. Questa rete permette di immagazzinare fino a $M \simeq 0.7N$ prototipi di dimensioni contenute, e fino a $M \simeq 0.2N$ prototipi da un flusso continuo.

1.8.3 Generalizzazioni del modello di Hopfield

Le generalizzazioni della rete di Hopfield riguardano:

- l'aggiunta di autoconnessioni
- la regola di memorizzazione (*Learning Rule*, LR)
- la dinamica nella fase di richiamo
- il ruolo delle soglie

1.8.4 Topologia e LR

Le prestazioni della rete vanno valutate nei seguenti termini:

- Error correction capability (how much noise can be tolerated by the network)
- Capacity (how many patterns can be stored)
- Training complexity (the amount of computations needed to memorize a pattern)
- Memory requirements (the amount of computer memory required to operate)
- Execution time (how many computations are needed in pattern retrieval)

Le prestazioni della rete dipendono unicamente dall'architettura della rete e dall'algoritmo di apprendimento.

Sparsely connected models of Associative Neural Networks (SAsNN), which use only a subset of all possible inter-neuron connections.

Topologia. Per ogni neurone è definito

$$N_i \subseteq \{1, \ldots, N\}, \quad i = 1, \ldots, N$$

L'uscita del neurone j-esimo è connessa a un ingresso del neurone i-esimo $\Leftrightarrow j \in N_i$.

La **densità delle connessioni** è

$$\rho = \frac{1}{N^2} \sum_{i=1}^{N} |N_i|$$

La **lunghezza totale delle connessioni** è

$$l = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{|N_i|} dist(i, N_i[j])$$

dove *dist* è una funzione di distanza opportuna.

Il **potenziale di attivazione** è

$$P_i = \sum_{k \in N_i} w_{ik} x_k$$

con $\mathbf{W} = [w_{ij}]$ matrice $N \times N$ dei pesi delle connessioni. La precedente viene normalmente corretta con

$$P_i = \sum_{k \in N_i} (1 - (1 - D)\delta_{ik}) w_{ik} x_k$$

dove D (tipicamente $0.05 \leq D \leq 0.15$) è il **coefficiente di desaturazione**, che serve ad attenuare il peso delle autoconnessioni (Gorodnichy, 1997). La desaturazione ha importanza quando la rete ha gia memorizzato un numero di patterns prossimo al limite massimo e il recupero in presenza di rumore è difficoltoso. In queste condizioni gli attrattori principali hanno un raggio ridotto ed esistono molti attrattori spuri (Reznik, 1993) che impediscono alla rete di convergere sugli attrattori principali. Tipicamente la desaturazione aumenta l'area degli attrattori principali (Kanter e Sompolinsky,1987), riduce il numero degli attrattori spuri stabili, ma aumenta (a parità di rumore) la possibilità di avere attrattori dinamici e aumenta il numero di cicli necessari alla convergenza verso un attrattore stabile. In presenza di molti attrattori dinamici è conveniente passare alla dinamica asincrona.

L'uscita del neurone è

$$x_i(t+1) = sign(P_i(t))$$

Si noti l'assenza di soglia. L'evoluzione della rete può essere sincrona o asincrona. Anche se biologicamente meno plausibile, l'evoluzione sincrona semplifica l'implementazione in HW della rete e produce proprietà associative migliori. La funzione di energia (Reznik)

$$E(t) \equiv -\frac{1}{2}\mathbf{x}(t+1)\mathbf{P}(t) = -\frac{1}{2}\sum_{i=1}^{N} x_i(t+1)P_i(t) = -\frac{1}{2}\sum_{i=1}^{N} |P_i(t)|^2$$

garantisce che la rete sincrona, a partire da uno stato qualsiasi, evolve verso uno stato stabile (*attrattore statico*). Se la rete sincrona è anche simmetrica (sicchè $\mathbf{W} = \mathbf{W}^T$) allora può convergere su un ciclo di lunghezza 2 (*attrattore dinamico*). Infatti si dimostra facilmente che $E(t)$ decresce monotonicamente $\Leftrightarrow \mathbf{x}(t+1) = \mathbf{x}(t) = \mathbf{x}(t-1)$ oppure $\mathbf{x}(t+1) = \mathbf{x}(t-1)$. Se invece $\mathbf{W}$ non è simmetrica i cicli asintotici possono avere lunghezza maggiore, anche se con bassa probabilità.

1.8.5 PINN: una AsNN con Projection Learning Rule

Una *Pseudo-Inverse Neural Network* (Brucoli, 1995)(PINN) è una rete auto-organizzante autoassociativa totalmente connessa ($K = N$), sincrona, che utilizza l'algoritmo di apprendimento PLR (*Projection Learning Rule*), per il quale

$$\mathbf{W} = proj(\{\mathbf{x}^s\}) = \mathbf{X}\mathbf{X}^+$$

dove $\mathbf{X} = [\mathbf{x}^s]$ è la matrice $N \times M$ formata dagli $M < N$ vettori colonna del *training set*, e $\mathbf{X}^+$ è la matrice *pseudo-inversa* di $\mathbf{X}$

$$\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

Espressa in questo modo, $\mathbf{W}$ è la matrice di proiezione ortogonale nel sottospazio generato dai vettori di *training* (Personnaz, 1986). Essa è soluzione della relazione di stabilità $\mathbf{W}\mathbf{X} = \mathbf{X}$ solo se i vettori del *training set* possono essere assunti linearmente indipendenti. Il calcolo di $\mathbf{W}$ può essere svolto con l'ortogonalizzazione di Gram-Schmidt direttamente in forma matriciale, nel qual caso avremmo una LR in modalità *batch*, assumendo che tutti i vettori di ingresso siano noti al momento dell'addestramento. Questa assunzione cade nelle applicazioni *real-time*, dove è auspicabile che i parametri della rete possano essere modificati sulla base di un solo vettore alla volta. La formula iterativa di Greville ci permette di ottenere un addestramento incrementale

$$w_{ij}^0 = 0 \tag{1.8.1}$$

$$w_{ij}^s = w_{ij}^{(s-1)} + \frac{(x_i^s - P_i^s)(x_j^s - P_j^s)}{N - \sum_{k=1}^N x_k^s P_k^s} \qquad se \quad \mathbf{X}^s \neq \mathbf{W}^{s-1}\mathbf{X}^s \tag{1.8.2}$$

$$w_{ij}^s = w_{ij}^{(s-1)} \qquad se \quad \mathbf{X}^s = \mathbf{W}^{s-1}\mathbf{X}^s \tag{1.8.3}$$

con

$$P_i^s = \sum_{k=1}^N w_{ki}^{s-1} x_k^s$$

Questa LR garantisce la convergenza della rete verso un attrattore stabile, cioè uno dei vettori di *training*, solo se la rete è simmetrica. Per la sua velocità è utilizzata nelle applicazioni *real-time*. Tra tutte le LR non-iterative, le *Projection Learning rules* sono le più efficienti. Notiamo che applicando la formula iterativa di Greville non si ottiene la stessa matrice W che si otterrebbe calcolando direttamente $\mathbf{X}\mathbf{X}^+$, e infatti la matrice che si ottiene non soddisfa $\mathbf{W}\mathbf{X} = \mathbf{X}$, ma soddisfa $stepf(\mathbf{W}\mathbf{X}) = \mathbf{X}$.

La **Update Flow Technique** è una tecnica, utilizzata nelle PINN, che permette di velocizzare la fase di riconoscimento. Consiste nell'aggiornare il potenziale di attivazione e l'uscita dei soli neuroni (J) il cui potenziale è cambiato rispetto alla precedente iterazione

$$P_i(t) = P_i(t-1) + \sum_{j=1}^{J} w_{ij} x_j(t)$$

Il risultato non cambia, ma il numero di moltiplicazioni da svolgere è molto minore, infatti l'insieme dei neuroni che cambiano stato si riduce drasticamente man mano che la rete evolve.

Per la PINN così definita valgono le seguenti definizioni e approssimazioni:

$$\langle w_{ii} \rangle \equiv \frac{1}{N} \sum_{i,j=1}^{N} w_{ij} \simeq \frac{M}{N}$$

$$\langle w_{ij}^2 \rangle \equiv \frac{1}{N^2} \sum_{i,j=1}^{N} w_{ij}^2 \simeq \frac{M(N-M)}{N^3}$$

$$\langle |w_{ij}| \rangle \equiv \frac{1}{N^2} \sum_{i,j=1}^{N} |w_{ij}|$$

dove M è il numero di pattern già memorizzati dalla rete. Il **raggio di attrazione medio** $\langle |A| \rangle$ può essere stimato con la formula di Gorodnichy (1995)

$$\langle |A| \rangle = \frac{\frac{1}{2} - \langle w_{ii} \rangle}{\langle |w_{ij}| \rangle}$$

Questa quantità è direttamente correlabile alla capacità della rete di recuperare un pattern memorizzato a partire da una versione rumorosa del pattern stesso. Quando $\langle |A| \rangle < 1$ questa capacità è perduta. Ciò accade grossomodo quando $M > \frac{N}{2}$. Per valutare analiticamente la **capacità residua** della rete, senza conoscere il numero M di pattern che questa ha già memoriazzato, è possibile utilizzare il grafico mostrato nella figura seguente (per $N = 100$).

Ricavato M dal grafico, la capacità residua sarà ovviamente $N - M$. Un parametro importante per valutare le prestazioni della rete è il **rapporto di correzione di errore**

$$\frac{H}{H_0} \equiv \frac{H(\mathbf{x}(\infty), \mathbf{a})}{H(\mathbf{x}(0), \mathbf{a})}$$

dove $\mathbf{a}$ è l'attrattore.

Un altro parametro è il numero di cicli necessario affinchè la rete converga.

Esiste la possibilità di **eliminare dalla memoria un pattern** degli M senza dovere ripartire da zero e memorizzare nuovamente i rimanenti $M - 1$. Sia $\mathbf{x}^k$ $(1 \leq k \leq M)$ il pattern che si desidera cancellare. Sia $\widehat{\mathbf{X}}$ la matrice avente per vettori colonna i pattern $\mathbf{x}^s$, escluso il k-esimo. La matrice iniziale

$$(\mathbf{X}^T \mathbf{X})^{-1}$$

può essere decomposta in tre parti:

1. c, il termine scalare in posizione (k, k)

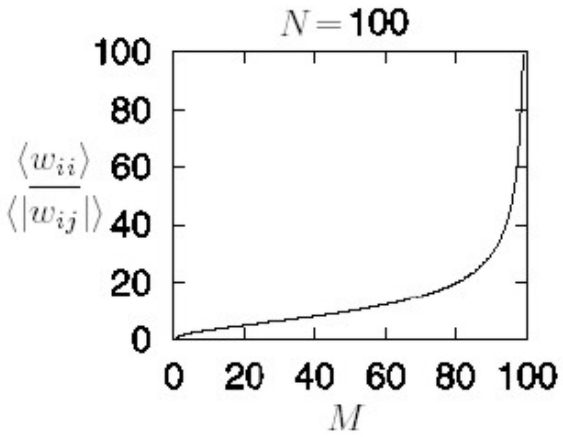

Figure 9: Tipico andamento di M in funzione dei parametri statistici della rete.

2. $\mathbf{b}$, la colonna k-esima privata dell'elemento k-esimo (c)
3. $\mathbf{A}$, la sottomatrice $(M-1) \times (M-1)$ ottenuta escludendo k-esima riga e k-esima colonna

Si dimostra allora che

$$(\widehat{\mathbf{X}}^T \widehat{\mathbf{X}})^{-1} = \mathbf{A} - \frac{1}{c} \mathbf{b}\mathbf{b}^T$$

e quindi la matrice dei pesi relativa alla rete privata del pattern $\mathbf{x}^k$ è

$$\widehat{\mathbf{W}} = \widehat{\mathbf{X}}(\mathbf{A} - \frac{1}{c} \mathbf{b}\mathbf{b}^T)\widehat{\mathbf{X}}^T$$

Esempio. Supponiamo di avere

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

e di volere cancellare il pattern corrispondente alla seconda colonna di $\mathbf{X}$. Troviamo che

$$(\mathbf{X}^T \mathbf{X})^{-1} = \begin{pmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

quindi

$$c = 1; \quad \mathbf{b} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad \mathbf{A} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

D'altra parte, togliendo da $\mathbf{X}$ la seconda colonna, abbiamo

$$\widehat{\mathbf{X}} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & -1 \end{pmatrix}$$

da cui si ricava che

$$(\widehat{\mathbf{X}}^T\widehat{\mathbf{X}})^{-1} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

che coincide proprio con

$$\mathbf{A} - \frac{1}{c}\mathbf{b}\mathbf{b}^T$$

1.9 Pattern Recognition statistico

Il **problema della classificazione** è quello di associare un'assegnazione del vettore

$$\mathbf{x} = (x_1, \ldots, x_d)^T \qquad \mathbf{x} \in \mathcal{X} \subseteq \Re^d$$

a una delle classi C_i $(i = 1, \ldots, c)$. In generale $\mathbf{x}$ potrebbe avere componenti continue o discrete. Possiamo inizialmente ipotizzare $\mathcal{X} = [0, 1]^d$.

Un **training set** (o *data set*) è un insieme di vettori $\mathbf{x}^1, \ldots, \mathbf{x}^N$ per i quali l'associazione già esiste. Formalmente, il nostro algoritmo di classificazione è una funzione

$$y_i = f(\mathbf{x}, \mathbf{w}) \qquad i = 1, \ldots, c$$

dove $\mathbf{w}$ è un vettore di parametri, e $y_i = 1(0) \Leftrightarrow x$ (non) è classificato in C_i.

In una rete neurale $\mathbf{w}$ sono i pesi, e la loro determinazione sulla base del *data set* è detta **addestramento** (*training*) o **apprendimento** (*learning*). Sia i problemi di classificazione che di regressione sono casi particolari del problema dell'approssimazione di funzioni.

Esempio - Un'applicazione con $d = 1$ e $c = 1$ è il fitting di una curva polinomiale di grado S:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_S x^S$$

Il *data set* consiste di N punti (x^n, t^n). Risolvere il problema equivale a determinare il set di parametri $\mathbf{w}$ che ottimizza il fitting in base a un criterio prestabilito.

1.9.1 Il problema della dimensionalità

Un modo inefficiente di specificare una funzione multivariata non lineare da $\mathbf{x}$ in $\mathbf{y}$ sarebbe quello di suddividere il range di ciascuna variabile di ingresso x_i in M intervalli, cioè lo spazio di $\mathbf{x}$ in M^d celle discrete, e specificare il valore di $\mathbf{y}$ per ciascuna di esse. Chiaramente questo metodo è impraticabile in quanto il *data set* richiesto aumenterebbe esponenzialmente con d (problema della dimensionalità dello spazio di ingresso), infatti bisognerebbe assicurarsi di avere campioni su tutte le celle. Per aggirare il problema è opportuno utilizzare informazioni a priori (*prior knowledge*), da aggiugere al *data set* di addestramento, come le proprietà di invarianza. P.es. Un sistema di riconoscimento visivo è tipicamente invariante alla traslazione e alla scala, cioè opera indipendentemente dalla posizione e dalla grandezza del pattern di ingresso. Metodi più efficienti tengono conto delle correlazioni che esistono tra le variabili di ingresso, da cui il concetto di *dimensionalità intrinseca*.

Esempio - I vettori $\mathbf{x}$ siano immagini 256x256 a risoluzione 8 bit per pixel, che rappresentano graficamente una "a" oppure una "b". Vogliamo classificare correttamente ciascuna immagine, associandola a una delle due classi C_1 (l'immagine di una "a") e C_2 (l'immagine di una "b").

E' assurdo pensare di classificare tutte le $2^{256\times256\times8}$ immagini possibili. Una tecnica per aggirare il problema è quella combinare molte di queste variabili in un piccolo numero di nuove variabili $\widetilde{\mathbf{x}}$, dette **features**. Nel nostro esempio una di queste nuove variabili potrebbe essere il rapporto tra altezza e larghezza del carattere, $\widetilde{x}_1$. La distribuzione di $\widetilde{x}_1$ sugli elementi del *data set* associati a C_1 avrà una media inferiore rispetto a quella della distribuzione sugli elementi del *data set* associati a C_2. Tuttavia, le due distribuzioni potranno sovrapporsi in un certo intervallo, sicché il parametro $\widetilde{x}_1$ non permetterebbe una distinguibilità assoluta. Una possibilità è quella di fissare un valore di soglia, proprio nel punto di intersezione delle due distribuzioni. L'aggiunta di un altro parametro $\widetilde{x}_2$ riduce ulteriormente la possibilità di errori di classificazione.

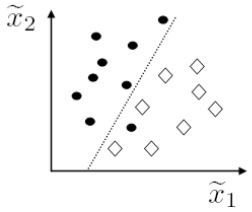

Figure 10: Separazione delle immagini in due classi nello spazio delle features.

Notiamo che con due parametri la separazione tra le due classi (linea tratteggiata) è migliore rispetto a quella che si avrebbe con uno solo dei due (le distribuzioni dei punti di proiezione su uno dei due assi hanno sempre una maggiore sovrapposizione). Aggiungendo sempre più parametri (indipendenti) i risultati migliorano, ma fino a un certo punto.

Lo schema di principio di una applicazione di classificatore sarà quindi

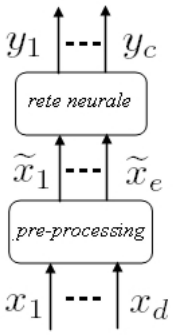

Figure 11: Schema a blocchi.

dove lo stadio di *pre-processing*, ed eventualmente di *post-processing*, sono trasformazioni fisse, mentre la rete neurale è una trasformazione a parametri adattativi. D'ora in poi chiamiamo $\mathbf{x}$ il vettore di input già processato dallo stadio di *pre-processing* (quello che in figura è indicato $\widetilde{\mathbf{x}}$).

1.9.2 Metodo della minimizzazione di una funzione di errore

Supponiamo di conoscere la forma della

$$y_k = f(\mathbf{x}, \mathbf{w})$$

dove M è il numero dei parametri liberi (gradi di libertà di f). Un procedimento per risolvere il problema della classificazione consiste nel determinare $\mathbf{w}$ in modo da minimizzare una qualche funzione d'errore, per esempio

$$E(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{c}[y_k(\mathbf{x}^n, \mathbf{w}) - t_k^n]^2$$

Sia $\mathbf{w}^*$ tale che $E(\mathbf{w}^*) = min\{E(\mathbf{w})\}$. $\mathbf{y}(\mathbf{x}, \mathbf{w}^*)$ è il modello risolvente. Notiamo che $\mathbf{y}$ è non lineare in $\mathbf{x}$, ma lineare in $\mathbf{w}$. Funzioni che sono lineari nei parametri adattatitivi sono dette **modelli lineari**.

La minimizzazione di E è un caso di **apprendimento supervisionato**, in quanto sono noti a priori i valori target $\{\mathbf{t}^n\}$. In altri problemi, come la determinazione della distribuzione $p(\mathbf{x})$, in cui non si dispone di $\{\mathbf{t}^n\}$, si tratta di un **apprendimento non supervisionato**. Dato un *training data set* $(\mathbf{x}^n, \mathbf{t}^n)$, e un *test data set* $(\mathbf{x}^s, \mathbf{t}^s)$ su cui verificare la capacità di generalizzazione del modello $\mathbf{y}(\mathbf{x}, \mathbf{w}^*)$, i risultati migliori si ottengono per M non troppo piccolo né troppo grande. Un numero di gradi di libertà troppo grande aumenta infatti la variabilità del modello (**over-fitting**).

Riassumendo, la complessità di un modello può essere fatta coincidere col numero di parametri liberi M del modello stesso (l'ordine del polinomio nell'esempio precedente, o il numero di unità nascoste in una rete neurale). A parità di *training data set*, la complessità del modello con le migliori capacità di generalizzazione non è né troppo alta né troppo bassa. Infatti, mentre $E(M)$ decresce monotonicamente se valutato sul *training data set*, raggiunge un minimo se valutato sul *test data set*. Un M troppo piccolo produce modelli con alto bias e bassa varianza, mentre elevati valori di M producono modelli con alta varianza e basso bias. L'ottimo si ha con un compromesso tra queste due opposte tendenze.

Estendendo il concetto del fitting polinomiale a $d > 1$ ingressi avremo bisogno di d^M parametri liberi. In tal caso per determinare tutti i parametri con sufficiente precisione occorreranno *data sets* di cardinalità proporzionale. Ma esistono modi alternativi di rappresentare una funzione non lineare multivariata, p.es. come sovrapposizione di funzioni non lineari univariate. Questo è il modello di una rete neurale. Il prezzo da pagare con le reti neuronali è il costo computazionale dell'algoritmo di apprendimento, che è non lineare. Una complicazione aggiuntiva è che $E(\mathbf{w})$ ha dei minimi locali sui quali la rete può convergere ma che non rappresentano la soluzione ottima.

1.9.3 Metodo di Bayes

Il **Teorema di Bayes** (Thomas Bayes, 1702-1761) è alla base dell'approccio statistico al pattern recognition. Il problema è quello di minimizzare la probabilità di errore di classificazione di un nuovo pattern $\mathbf{x}$ in una delle classi C_k. Ora, per un pattern di features $\mathbf{x}$, la probabilità di classificazione errata è minima se associamo $\mathbf{x}$ alla classe C_k per la quale è massima

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|C_k)P(C_k)}{\sum_{i=1}^{c} P(\mathbf{x}|C_i)P(C_i)} \tag{1.9.1}$$

dove $P(C_k|\mathbf{x})$ sono le **probabilità a posteriori**, $P(C_k)$ le **probabilità a priori**, e $P(\mathbf{x}|C_k)$ le **probabilità condizionali** di classe (*class conditional*), cioè

$$P(C_k|\mathbf{x}) \geq P(C_j|\mathbf{x}) \qquad \forall j \neq k$$

Osserviamo che $\sum_{i=1}^{c} P(C_i) = 1$

Nel caso di variabili continue, disponendo di una stima delle densità di probabilità $p(\mathbf{x})$ e $p(\mathbf{x}|C_k)$ il teorema di Bayes può scriversi anche

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)P(C_k)}{\sum_{i=1}^{c} p(\mathbf{x}|C_i)P(C_i)} \tag{1.9.2}$$

1.9.4 Bordi di decisione

Un classificatore di patterns è un'applicazione che associa ogni punto nello spazio delle features a una delle c classi. Possiamo immaginare lo spazio delle features suddiviso in c regioni di decisione $\mathcal{X}_1, \ldots, \mathcal{X}_c$, tali che un punto che cade nella regione $\mathcal{X}_k$ viene associato alla classe C_k. I bordi di queste regioni sono noti come **bordi di decisione** (*decision boundaries*) o **superfici di decisione** (*decision surfaces*).

Il punto di decisione ottimo per un problema di classificazione di una variabile su due classi ($c = 2$) si ottiene minimizzando la probabilità di errore. Detta $P(x \in \mathcal{X}_u, C_v)$ la probabilità congiunta che x appartenga a $\mathcal{X}_u$ e che sia C_v la vera classe di appartenenza di x, abbiamo

$$P(errore) = P(x \in \mathcal{X}_1, C_2) + P(x \in \mathcal{X}_2, C_1) = \tag{1.9.3}$$

$$P(x \in \mathcal{X}_1 | C_2) P(C_2) + P(x \in \mathcal{X}_2 | C_1) P(C_1) = \tag{1.9.4}$$

$$P(C_2) \int_{\mathcal{X}_1} p(x|C_2) dx + P(C_1) \int_{\mathcal{X}_2} p(x|C_1) dx \tag{1.9.5}$$

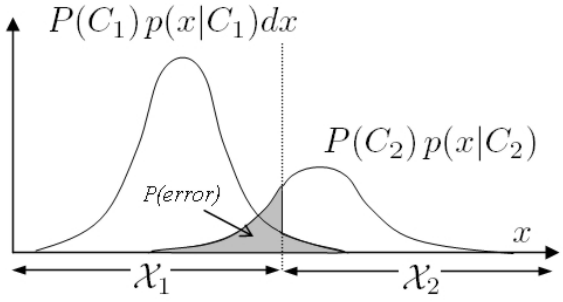

Figure 12: Probabilità associare x alla classe sbagliata.

Si possono definire funzioni discriminanti $y_1(\mathbf{x}), \ldots, y_k(\mathbf{x})$ tali che $\mathbf{x}$ è assegnato alla classe C_k se

$$y_k(\mathbf{x}) > y_j(\mathbf{x}) \qquad \forall j \neq k$$

La scelta che minimizza la probabilità di errore è

$$y_k(\mathbf{x}) \equiv p(\mathbf{x}|C_k) P(C_k) \tag{1.9.6}$$

o una qualunque funzione monotona della 1.9.6.

1.9.5 Stima della densità di probabilità

I metodi di stima della $p(\mathbf{x})$ sono

1. Metodi parametrici, basati su un modello funzionale prestabilito
2. Metodi non parametrici
3. Metodi semiparametrici (*mixture distributions*)

Tra i modelli parametrici più utilizzati vi è la distribuzione Gaussiana

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)}{2}}$$

dove μ è il **vettore di media** $\mathcal{E}[\mathbf{x}]$. In base al **criterio di massima verisimiglianza** (*maximum likelihood*) la stima migliore per μ è

$$\widehat{\mu} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^n$$

mentre la matrice simmetrica $\mathbf{\Sigma}$ è la **matrice di covarianza** $\mathcal{E}[(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T]$. Sempre in base al criterio di massima verisimiglianza la stima migliore per $\mathbf{\Sigma}$ è

$$\widehat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}^n - \widehat{\mu})(\mathbf{x}^n - \widehat{\mu})^T$$

In generale, la stima di $p(\mathbf{x})$ richiede $\frac{d(d+3)}{2}$ parametri. In $\mathcal{X}$, le regioni a $p(\mathbf{x}) = cost.$ sono iperellissoidi i cui assi principali e le rispettive varianze sono dati dagli autovettori e dagli autovalori corrispondenti di $\mathbf{\Sigma}$

$$\mathbf{\Sigma u_i} = \lambda_i \mathbf{u_i}$$

Se $(\mathbf{\Sigma})_{ij} = \delta_{ij}\sigma_i^2$ (diagonale) allora gli assi $\mathbf{u_i}$ sono allineati con gli assi coordinati, le componenti di $\mathbf{x}$ sono statisticamente indipendenti, e

$$p(\mathbf{x}) = \prod_{i=1}^{d} p(x_i)$$

Un metodo alternativo al criterio della massima verisimiglianza è il metodo di **inferenza Bayesiana** (*Bayesian inference*), che per N grande produce le stesse stime.

- Deciding upon the appropriate number of hidden nodes and layers is largely a matter of experience. With many problems, suffcient accuracy can be obtained with one or two hidden layers and 5–10 hidden nodes in those layers. There is a theorem which states that a network with one hidden layer can approximate any continuous function to arbitrary

accuracy, provided it has a suffcient number of hidden nodes (Hornick, Stinchcombe and White 1990). In practice, such a large number of nodes may be required that it is more effcient to go to a second hidden layer.

- An important consideration in setting up the network is the ratio of data to weights. If we train a network with one output using N training vectors, we have a total of N error measurements with which to determine the W weights. Thus we have a system of N equations with W unknowns, and, if the weights and training vectors are independent of each other, we require $N > W$ to find a solution. With fewer data the solution for the weights will be underdetermined. Thus a 10:5:5:1 network (indicating 10 inputs, 5 nodes in each of two hidden layers, and one output) has 91 weights (including the bias nodes), so will require at least as many training examples. The data/weight ratio estimation is more complex with multiple outputs (especially if they are correlated, as will be the case in probabilistic mode), but if there are two independent outputs, then the number of error measures is 2N.

2 Nozioni su matrici e sistemi lineari

Definizione 2.0.1. $\mathbb{C}^{m,n}$ ($\mathbb{R}^{m,n}$) sia lo spazio delle matrici complesse (reali) $m \times n$, cioè con m righe e n colonne, e in particolare $\mathbb{R}^n \equiv \mathbb{R}^{n,1}$

Definizione 2.0.2. Il **rango** (*Rank*) di $\mathbf{A}$ è il numero di righe o di colonne linearmente indipendenti di $\mathbf{A}$. Il rango di una matrice in $\mathbb{C}^{m,n}$ è uguale al rango della più grande sottomatrice quadrata con determinane non nullo

Definizione 2.0.3. Data $\mathbf{U}$ la **trasposta coniugata** è $\mathbf{U}^H \equiv (\mathbf{U}^T)^*$

Definizione 2.0.4. $\mathbf{U}$ è **simmetrica** $\Leftrightarrow$ $\mathbf{U}^T = \mathbf{U}$

Definizione 2.0.5. $\mathbf{U}$ è **Hermitiana** $\Leftrightarrow$ $\mathbf{U}^H = \mathbf{U}$. Una matrice simmetrica reale è Hermitiana.

Definizione 2.0.6. $\mathbf{U} \in \mathbb{C}^{n,n}$ è **unitaria** $\Leftrightarrow$ $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$ $\Leftrightarrow$ $\mathbf{U}^{-1} = \mathbf{U}^H$

Definizione 2.0.7. I **valori singolari** σ_i di $\mathbf{A} \in \mathbb{C}^{n,n}$ sono la radice quadrata degli autovalori di $\mathbf{A}^H\mathbf{A}$

Definizione 2.0.8. La **decomposizione a valori singolari** di $\mathbf{A} \in \mathbb{C}^{n,n}$ (*Singular value decomposition*) è

$$\mathbf{A} = \mathbf{U}^H\mathbf{\Sigma}\mathbf{V}$$

dove $\mathbf{U}$ e $\mathbf{V}$ sono matrici unitarie e $\Sigma = diag(\sigma_i)$

Definizione 2.0.9. La pseudoinversa (o *inversa generalizzata*) di $\mathbf{A} \in \mathbb{C}^{m,n}$ soddisfa

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$$
$$\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$$
$$(\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+$$
$$(\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}$$

La definizione 2.0.9 è equivalente alla seguente

Definizione 2.0.10. La pseudoinversa di $\mathbf{A} \in \mathbb{C}^{m,n}$ è quella matrice $\mathbf{A}^+ \in \mathbb{C}^{n,m}$ tale che $\forall \mathbf{y} \in \mathbb{C}^m$, soddisfa

$$\mathbf{A}^+\mathbf{y} = argmin_{\mathbf{x} \in \{\mathbf{x}\}} \|\mathbf{x}\|$$

dove

$$\{\mathbf{x}\} = argmin_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|$$

Questa definizione ci permette di utilizzare la pseudoinversa per calcolare la soluzione dell'equazione $\mathbf{Ax} = \mathbf{y}$ che, tra tutte le soluzioni che minimizzano $\|\mathbf{Ax} - \mathbf{y}\|$, ha norma minima.

Proposizione 2.0.11. *Una permutazione di riga in* $\mathbf{A}$ *produce la stessa permutazione di colonna nella matrice pseudoinversa* $\mathbf{A}^+$.

Definizione 2.0.12. La **pseudoinversa** di $\mathbf{A} \in \mathbb{C}^{m,n}$, con $Rank(\mathbf{A}) = m \leq n$ è

$$\mathbf{A}^+ = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$$

La pseudoinversa è in $\mathbb{C}^{n,m}$.

Se invece è $\mathbf{A} \in \mathbb{C}^{n,m}$, con $Rank(\mathbf{A}) = m \leq n$ è

$$\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

In questo caso $\mathbf{A}^+ \in \mathbb{C}^{m,n}$.

2.1 Formula di Greville diretta

La formula di Greville diretta ci permette di calcolare in modo ricorsivo $\mathbf{A}^+$. Supponiamo di conoscere $\mathbf{A}_m$ e la sua pseudoinversa $\mathbf{A}_m^+$, e di voler calcolare la pseudoinversa di $\mathbf{A}_m$ aumentata di una riga

$$\mathbf{A}_{m+1} \equiv \begin{pmatrix} \mathbf{A}_m \\ \mathbf{a}^T \end{pmatrix}$$

La formula ci dice che

$$\mathbf{A}_{m+1}^+ = \begin{pmatrix} \mathbf{A}_m^+ - \mathbf{b}_{m+1}\mathbf{a}^T\mathbf{A}_m^+ & \mathbf{b} \end{pmatrix}$$

dove

$$\mathbf{b} = \begin{cases} \frac{\mathbf{c}}{\|\mathbf{c}\|^2} & \text{se } \|\mathbf{c}\| > 0 \\ \frac{\mathbf{A}_m^+(\mathbf{A}_m^+)^T\mathbf{a}}{1+\|(\mathbf{A}_m^+)^T\mathbf{a}\|^2} & \text{se } \|\mathbf{c}\| = 0 \end{cases}$$

$$\mathbf{c} = \mathbf{a} - \mathbf{A}_m^T(\mathbf{A}_m^+)^T\mathbf{a}$$

Nel calcolo dell'algoritmo conviene calcolare il risultato intermedio $(\mathbf{A}_m^+)^T\mathbf{a}$. Inoltre, come criterio di scelta si può usare $t \equiv \frac{\|\mathbf{c}\|}{\|\mathbf{a}\|} > \epsilon > 0$.

2.2 Formula di Greville inversa

La formula di Greville inversa ci permette di calcolare in modo ricorsivo $\mathbf{A}^+$ avendo tolto l'ultima riga di $\mathbf{A}$. Sia

$$\mathbf{A}_{m+1} \equiv \begin{pmatrix} \mathbf{A}_m \\ \mathbf{a}^T \end{pmatrix}$$

e la sua pseudoinversa

$$\mathbf{A}^+_{m+1} \equiv \begin{pmatrix} \mathbf{B} & \mathbf{b} \end{pmatrix}$$

Quello che segue vale per qualsiasi riga di $\mathbf{A}$, pur di selezionare la colonna corrispondente in $\mathbf{A}^+$. Allora

$$\mathbf{A}^+_m = \begin{cases} \mathbf{B} - \frac{\mathbf{b}\mathbf{b}^T\mathbf{B}}{\|\mathbf{b}\|^2} & \text{se } \mathbf{a}^T \cdot \mathbf{b} - 1 = 0 \\ \\ \mathbf{B} + \frac{\mathbf{b}\mathbf{a}^T\mathbf{B}}{1-\mathbf{a}^T\cdot\mathbf{b}} & \text{se } \mathbf{a}^T \cdot \mathbf{b} - 1 \neq 0 \end{cases}$$

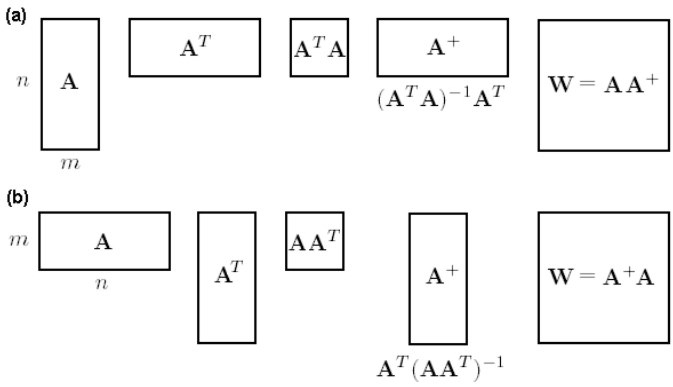

Figure 13: Matrice dei pesi $\mathbf{W}$ di una ANN con PLR nei casi in cui il *training set* sia organizzato in colonne (a) ovvero in righe (b) della matrice $\mathbf{A}$.

References

[1] Resurces on the web.

[2] Personal communications with expert people.