BLS: Broad Learning System

Tesi di Laurea in Programmazione Matematica

Corso di Laurea Magistrale in Matematica Applicata

Marco Sacchetti (Matricola 1974451)

Anno accademico 2024/2025

SAPIENZA
UNIVERSITÀ DI ROMA

Indice.

Le reti neurali.

1 Definizioni preliminari.

Una **rete neurale artificiale** è un modello matematico che approssima una funzione sconosciuta

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

tramite la composizione di trasformazioni lineari e non lineari.
L'unità fondamentale di una rete è il *neurone* che, ricevuto in input un vettore, produce una singola uscita pesata.
Organizzando più neuroni in L *strati* successivi si ottiene una rete profonda (o **deep neural network**).
Lo strato l-esimo esegue una operazione del tipo:

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right)$$

dove:

- $h^{(0)} = x \in \mathbb{R}^n$ rappresenta l'input e $h^{(L)} = \hat{y}$ l'output stimato;
- $W^{(l)}$ e $b^{(l)}$ sono i *parametri* (pesi e bias) da determinare;
- $\sigma(\cdot)$ è una funzione di attivazione, generalmente non lineare (ReLU, sigmoide, tanh, ecc.).

Le reti neurali.

1 Definizioni preliminari.

L'addestramento di una rete neurale consiste nel determinare l'insieme dei parametri

$$\Theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \ldots, W^{(L)}, b^{(L)}\}$$

che minimizza una funzione di errore tra le uscite previste $\hat{Y}$ e i valori reali Y:

$$\min_{\Theta} \ \mathcal{L}(Y, \hat{Y})$$

dove $\mathcal{L}$ è la *loss function* (ad esempio MSE).
La ricerca del minimo avviene mediante algoritmi di ottimizzazione iterativi, come la **discesa del gradiente** o le sue varianti (SGD, Adam, RMSProp, ecc.), che aggiornano i pesi nella direzione opposta al gradiente della perdita:

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

con η *tasso di apprendimento* (oppure *learning rate*).

Indice.

Broad Learning System

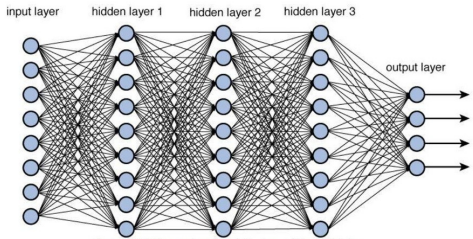
Un'alternativa alle reti tradizionali.

Nel 2018 è stato presentato un modello alternativo alle reti neurali tradizionali, che privilegia l'espansione in ampiezza, anziché in profondità: il *Broad Learning System* [1]. Le principali criticità delle deep networks sono:

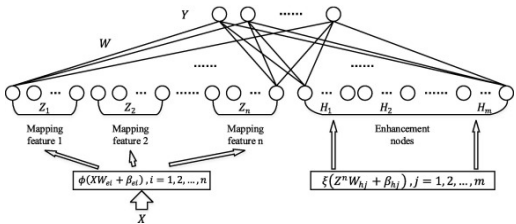
- necessità di dataset estesi e architetture complesse (molti strati e parametri);
- backpropagation su molti strati $\rightarrow$ training lento e costoso;
- instabilità dei gradienti (*vanishing/exploding*) che rende difficile l'addestramento;
- aggiornamento *incrementale* oneroso: bisogna riaddestrare l'intero modello.

Broad Learning System

Architettura del BLS.



(a) Rete neurale tradizionale.



(b) Rete neurale di tipo BLS.

Broad Learning System

Architettura del BLS.

In una rete BLS abbiamo tre componenti fondamentali:

1. **Mapping Feature Nodes**: a partire dai dati in input X, si generano n famiglie di nodi $Z_1, \ldots, Z_n$ (*feature mapping nodes*), applicando trasformazioni del tipo

$$Z_i = \phi_i(XW_i + \beta_i), \qquad i = 1, \ldots, n,$$

dove W_i e β_i sono inizializzati casualmente e ϕ_i, dette *funzioni di attivazione*, sono funzioni qualsiasi.

2. **Enhancement Nodes**: i *nodi di enhancement (o potenziamento)* ricevono in ingresso la concatenazione dei nodi generati $Z^n \equiv \left[Z_1, \ldots, Z_n\right]$ ed applicano una trasformazione non lineare, ottenendo m famiglie di nodi di potenziamento:

$$H_j = \xi_j(Z^n W_j + \beta_j), \qquad j = 1, \ldots, m.$$

dove W_j e β_j sono inizializzati casualmente e ξ_j sono funzioni di attivazione non lineari (ad es. ReLU, tangente iperbolica, sigmoide, ...). Si pone $H^m \equiv \left[H_1, \ldots, H_m\right]$.

Broad Learning System

Architettura del BLS.

3. **Output Layer:** definita la matrice complessiva dei nodi $A = \left[Z^n \mid H^m \right]$, l'uscita della rete e i pesi di uscita, che collegano l'unico strato nascosto all'output, si scrivono come:

$$\hat{Y} = AW_{\text{out}}, \quad \text{da cui } W_{\text{out}} = A^{\dagger} Y,$$

dove $A^{\dagger}$ è la *pseudoinversa* di Moore–Penrose e W_{out} è la matrice dei pesi di uscita.

Broad Learning System

Addestramento di una rete BLS: stima dei pesi di output.

Per addestrare una rete BLS si seguono i passaggi [1]:

- Si costruisce la matrice espansa tipica del BLS arricchita con una colonna di 1 per il *bias*, un parametro che permette di modellare funzioni che non passano per l'origine:

$$A = \left[Z^n \mid H^m \mid \mathbf{1}_N \right] \in \mathbb{R}^{N \times (D+1)}, \qquad D = N_f + N_e.$$

- L'uscita della rete è

$$\hat{Y} = AW_{\text{out}}, \quad \text{con } \hat{Y} \in \mathbb{R}^{N \times C},\ W_{\text{out}} \in \mathbb{R}^{(D+1) \times C},$$

pertanto la stima di W_{out} si ottiene risolvendo il problema:

$$\min_{W \in \mathbb{R}^{(D+1) \times C}} \tfrac{1}{2} \|AW - Y\|^2 + \tfrac{1}{2} \lambda \|W\|^2, \qquad \lambda > 0. \tag{1}$$

- La soluzione in forma chiusa è

$$W_{\text{out}} = \left(A^\top A + \lambda I_{D+1} \right)^{-1} A^\top Y.$$

Broad Learning System

Basic BLS — algoritmo di addestramento.

Algorithm 0: Basic BLS.

Input : X dataset, Y target, N numero di osservazioni, n numero di gruppi di nodi di mapping feature, m numero di gruppi di nodi di enhancement, $\lambda > 0$, attivazioni $\{\phi_i\}_{i=1}^{n}$, $\{\xi_j\}_{j=1}^{m}$.

Output: W_{out} matrice dei pesi di output.

for $i = 1$ **to** n **do**
    genera $W_i,\ \beta_i$ random;
    $Z_i \leftarrow \phi_i(XW_i + \beta_i)$;
Calcola i nodi feature mapping $Z^n \leftarrow [Z_1, \dots, Z_n]$;
for $j = 1$ **to** m **do**
    genera $W_j,\ \beta_j$ random;
    $H_j \leftarrow \xi_j(Z^n W_j + \beta_j)$;
Calcola i nodi enhancement $H^m \leftarrow [H_1, \dots, H_m]$.
Genera la matrice del BLS $A \leftarrow [Z^n \mid H^m \mid 1_N]$.
Calcola i pesi di uscita $W_{\text{out}} \leftarrow (A^\top A + \lambda I)^{-1} A^\top Y$.

Indice.

Obiettivi e contributi

Sintesi del lavoro svolto.

1. studiare in termini di qualità della soluzione e tempi di training un algoritmo iterativo per l'addestramento di una rete BLS;
2. proporre nuovi algoritmi di tipo *on-line*, in scenari in cui il dataset non è interamente disponibile dall'inizio, ma i campioni vengono collezionati progressivamente.

Indice.

CGM per il Broad Learning System

Il metodo del gradiente coniugato (*CGM*) per il BLS.

Per addestrare una rete, bisogna risolvere un problema di ottimizzazione regolarizzato:

$$\min_{W \in \mathbb{R}^{(D+1)\times 1}} \frac{1}{2} \|AW - Y\|^2 + \frac{1}{2} \lambda \|W\|^2, \qquad \lambda > 0,$$

che, senza perdita di generalità, si può scrivere nella forma classica ai minimi quadrati:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|^2.$$

dove $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, mentre $A \in \mathbb{R}^{m \times n}$.

Algorithm 1: Pseudocodice CGM per il sistema $Qx = c$, con Q semidefinita positiva.

Input : $x_0 \in \mathbb{R}^n$ un punto iniziale, tale che $g_0 \equiv \nabla f(x_0) = Qx_0 - c \neq 0$

Output: vettore soluzione x.

Poni $d_0 \leftarrow -g_0, \quad k \leftarrow 0$.

while $\|\nabla f(x_k)\| > 0$ **do**

Calcola il passo $\alpha_k \leftarrow -\dfrac{\nabla f(x_k)^\top d_k}{d_k^\top Q d_k}$;

Aggiorna lo stato $x_{k+1} \leftarrow x_k + \alpha_k d_k$;

Aggiorna il gradiente $\nabla f(x_{k+1}) \leftarrow \nabla f(x_k) + \alpha_k Q d_k$;

Calcola il coefficiente di aggiornamento di *Fletcher-Reeves* $\beta_{k+1} \leftarrow \dfrac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2}$;

Aggiorna la direzione di ricerca $d_{k+1} \leftarrow -\nabla f(x_{k+1}) + \beta_{k+1} d_k$;

$k \leftarrow k + 1$.

CGM per il Broad Learning System

Primo esperimento: tabelle di confronto.

Task: partendo dal dataset `slice_localization_data.csv`, formato da circa 54000 campioni e 386 features, addestrare una rete che, data la fotografia di una parte del corpo, preveda la corretta posizione.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	R^2_te	t_tr [s]
100	1000	0.1	0.12114	0.13230	0.13252	0.86777	0.44427
100	10000	0.1	0.00862	0.03029	0.03196	0.96788	68.7910
200	1000	0.1	0.10666	0.11368	0.11522	0.88503	0.46234
200	10000	0.1	0.00878	0.02951	0.03075	0.96909	82.4210
400	1000	0.1	0.08118	0.09141	0.09137	0.90883	0.61586
400	10000	0.1	0.00746	0.02747	0.02804	0.97181	81.6560
500	1000	0.1	0.07482	0.08568	0.08641	0.91377	0.71903
500	10000	0.1	0.00760	0.02644	0.02744	0.97243	85.1320
1000	1000	0.1	0.05696	0.06610	0.06724	0.93291	1.32270
1000	10000	0.1	0.00692	0.02737	0.02734	0.97253	99.2900

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	R^2_te	ifail	t_tr [s]
100	1000	0.1	0.11231	0.12389	0.12453	0.87566	3	1.9751
100	10000	0.1	0.01338	0.02718	0.02742	0.97262	3	18.450
200	1000	0.1	0.09199	0.10195	0.10316	0.89701	3	2.0541
200	10000	0.1	0.01122	0.02366	0.02402	0.97602	3	18.703
400	1000	0.1	0.07819	0.08782	0.08849	0.91165	3	2.5851
400	10000	0.1	0.01113	0.02157	0.02199	0.97804	2	18.274
500	1000	0.1	0.07298	0.08279	0.08495	0.91518	3	2.7847
500	10000	0.1	0.01171	0.02267	0.02373	0.97631	2	17.863
1000	1000	0.1	0.05830	0.06993	0.06903	0.93108	2	3.5915
1000	10000	0.1	0.01258	0.02355	0.02399	0.97604	2	15.040

Tabella: CGM. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Primo esperimento: considerazioni.

Nel testing, **MSE** sovrapponibili per entrambi i modelli, indipendentemente dalle configurazioni della rete (fino a $\sim \mathcal{O}(10^{-2})$).

Coefficiente $\mathbf{R^2}$ buono per entrambi i modelli (~ 1).

Tempi di training nettamente inferiori nel metodo iterativo: a parità di nodi, fino a **99 s** per il diretto e **15 s** per il CGM.

CGM per il Broad Learning System

Secondo esperimento: tabelle di confronto.

Task: partendo dal dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, addestrare una rete BLS in grado di prevedere la superconduttività di alcuni materiali metallici.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	R^2_te	t_tr [s]
100	1000	0.1	0.15485	0.18026	0.17894	0.83045	0.25331
100	10000	0.1	0.07470	0.17384	0.17882	0.83056	68.688
200	1000	0.1	0.14736	0.17370	0.19101	0.81901	1.5427
200	10000	0.1	0.05900	0.28019	0.29295	0.72241	260.51
400	1000	0.1	0.13713	0.17984	0.17948	0.82993	0.36409
400	10000	0.1	0.04801	0.62696	0.65926	0.37531	73.230
500	1000	0.1	0.13393	0.17535	0.18504	0.82466	0.40398
500	10000	0.1	0.04589	0.76072	0.55349	0.47553	75.693
1000	1000	0.1	0.11451	0.19200	0.18974	0.82021	0.83356
1000	10000	0.1	0.03838	1.37600	1.39100	-0.31805	178.63

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	R^2_te	ifail	t_tr [s]
100	1000	0.1	0.16583	0.18268	0.18317	0.82644	3	1.0808
100	10000	0.1	0.13315	0.15379	0.15764	0.85063	3	7.9993
200	1000	0.1	0.15335	0.17476	0.18913	0.82079	3	1.0242
200	10000	0.1	0.11460	0.14430	0.15241	0.85558	3	8.1823
400	1000	0.1	0.14962	0.17444	0.17626	0.83298	3	1.2163
400	10000	0.1	0.10347	0.13950	0.14774	0.86001	2	8.2021
500	1000	0.1	0.14067	0.16994	0.18018	0.82926	3	1.2388
500	10000	0.1	0.10056	0.13838	0.14368	0.86386	2	8.3084
1000	1000	0.1	0.13975	0.16588	0.16765	0.84114	2	1.4356
1000	10000	0.1	0.09124	0.13187	0.13792	0.86931	2	8.6651

Tabella: CGM. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Secondo esperimento: considerazioni.

Miglior **MSE** nella fase di test ottenuto con modello iterativo: nella configurazione $N_f = 1000, N_e = 10000$, il metodo con CGM ha un MSE $\sim \mathcal{O}(10^{-1})$ mentre il diretto $\sim \mathcal{O}(1)$.

Coefficiente $\mathbf{R^2}$ buono per la rete con CGM; nel caso diretto, degrada all'aumentare dei nodi.

Tempi di training nettamente migliori nel metodo iterativo: fino a **178 s** per il diretto e **8 s** per il CGM.

L'**early stopping** permette di concludere l'addestramento prima che il modello inizi a replicare fedelmente i dati di training, restituendo una soluzione corretta e risparmiando risorse.

Indice.

5 L'*Incremental Least Squares* per l'addestramento di una rete BLS.

ILS per il Broad Learning System

Il Filtro di Kalman: derivazione.

Si consideri il problema ai minimi quadrati lineare con funzione obiettivo:

$$f^{(k)}(\mathbf{x}) = \sum_{i=1}^{k} (\mathbf{a}_i^T \mathbf{x} - b_i)^2,$$

dove $\mathbf{a_i} \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ e si indichi con x_k^* la soluzione ottima:

$$\mathbf{x}_k^* = \underset{\mathbf{x} \in \mathbb{R}^n}{\operatorname{Arg\,min}} \sum_{i=1}^{k} (\mathbf{a}_i^T \mathbf{x} - b_i)^2.$$

All'arrivo di un nuovo campione $(\mathbf{a}_{k+1}, b_{k+1})$, la funzione obiettivo viene aggiornata ed assume la seguente forma:

$$f^{(k+1)}(\mathbf{x}) = f^{(k)}(\mathbf{x}) + (\mathbf{a}_{k+1}^T \mathbf{x} - b_{k+1})^2,$$

dando vita ad un nuovo problema di ottimizzazione:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \sum_{i=1}^{k+1} (\mathbf{a}_i^\top \mathbf{x} - b_i)^2$$

ILS per il Broad Learning System

Il Filtro di Kalman: derivazione.

Introducendo le notazioni matriciali:

$$A_k = \begin{pmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_k^T \end{pmatrix} \in \mathbb{R}^{k \times n}, \qquad \mathbf{b_k} = \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} \in \mathbb{R}^k,$$

$$A_{k+1} = \begin{pmatrix} A_k \\ \mathbf{a}_{k+1}^T \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_k^T \\ \mathbf{a}_{k+1}^T \end{pmatrix} \in \mathbb{R}^{(k+1) \times n}, \qquad \mathbf{b_{k+1}} = \begin{pmatrix} \mathbf{b}_k \\ b_{k+1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_k \\ b_{k+1} \end{pmatrix} \in \mathbb{R}^{k+1},$$

le funzioni obiettivo possono essere scritte anche come:

$$f^{(k)}(\mathbf{x}) = \tfrac{1}{2}\|A_k\mathbf{x} - \mathbf{b}_k\|^2,$$

$$f^{(k+1)}(\mathbf{x}) = \tfrac{1}{2}\|A_{k+1}\mathbf{x} - \mathbf{b}_{k+1}\|^2.$$

Se $\mathbf{x}_k^*$ è il punto di minimo di $f^{(k)}$, deve soddisfare le equazioni note come *equazioni normali*:

$$A_k^\top A_k \, \mathbf{x}_k^* = A_k^\top \mathbf{b}_k. \tag{2}$$

La soluzione del nuovo problema, indicata con $\mathbf{x}_{k+1}^*$, deve soddisfare a sua volta le equazioni normali:

$$A_{k+1}^\top A_{k+1} \, \mathbf{x}_{k+1}^* = A_{k+1}^\top \mathbf{b}_{\mathbf{k+1}}, \tag{3}$$

che in forma matriciale corrispondono alla seguente espressione:

$$\begin{pmatrix} A_k \\ \mathbf{a}_{k+1}^T \end{pmatrix}^T \begin{pmatrix} A_k \\ \mathbf{a}_{k+1}^T \end{pmatrix} \mathbf{x}_{k+1}^* = \begin{pmatrix} A_k \\ \mathbf{a}_{k+1}^T \end{pmatrix}^T \begin{pmatrix} \mathbf{b}_k \\ b_{k+1} \end{pmatrix}.$$

ILS per il Broad Learning System

Il Filtro di Kalman: derivazione.

Ponendo

$$H(k+1) \equiv A_{k+1}^{\top} A_{k+1} = A_k^{\top} A_k + \mathbf{a}_{k+1} \mathbf{a}_{k+1}^{T}, \tag{4}$$

le equazioni normali diventano:

$$H(k+1)\mathbf{x}_{k+1}^{*} = A_k^{\top} \mathbf{b}_k + \mathbf{a}_{k+1} b_{k+1}.$$

Poiché

$$A_k^{\top} A_k \, \mathbf{x}_k^{*} = A_k^{\top} \mathbf{b}_k,$$

si ha:

$$H(k+1)\mathbf{x}_{k+1}^{*} = A_k^{\top} A_k \, \mathbf{x}_k^{*} + \mathbf{a}_{k+1} b_{k+1} \tag{5}$$

$$= \left(A_k^{\top} A_k + \mathbf{a}_{k+1} \mathbf{a}_{k+1}^{T}\right) \mathbf{x}_k^{*} - \mathbf{a}_{k+1} \mathbf{a}_{k+1}^{T} \mathbf{x}_k^{*} + \mathbf{a}_{k+1} b_{k+1}, \tag{6}$$

cioè

$$H(k+1)\mathbf{x}_{k+1}^{*} = H(k+1)\mathbf{x}_k^{*} + \mathbf{a}_{k+1}\left(b_{k+1} - \mathbf{a}_{k+1}^{T} \mathbf{x}_k^{*}\right). \tag{7}$$

ILS per il Broad Learning System

Il Filtro di Kalman: derivazione.

Si supponga che per un certo k, $H(k) = A_k^\top A_k$ sia non singolare. Anche $H(k+1)$ sarà non singolare.
In tal caso la soluzione può essere scritta come:

$$\mathbf{x}_{k+1}^* = \mathbf{x}_k^* + H(k+1)^{-1}\mathbf{a}_{k+1}\big(b_{k+1} - \mathbf{a}_{k+1}^T\mathbf{x}_k^*\big). \tag{8}$$

In sintesi, abbiamo ricavato le *formule di aggiornamento*, anche note come *Filtro di Kalman:*

$$\boxed{\begin{aligned} H(k+1) &= H(k) + \mathbf{a}_{k+1}\mathbf{a}_{k+1}^T, \quad k = k_0, k_0+1, \dots \\ \mathbf{x}_{k+1}^* &= \mathbf{x}_k^* + H(k+1)^{-1}\mathbf{a}_{k+1}\big(b_{k+1} - \mathbf{a}_{k+1}^T\mathbf{x}_k^*\big). \end{aligned}}$$

ILS per il Broad Learning System

La formula di Sherman–Morrison–Woodbury (SWM).

L'aggiornamento della soluzione richiede l'inversione di $H(k+1)$, ma calcolare esplicitamente $H(k+1)^{-1}$ ad ogni passo avrebbe costo $\mathcal{O}(n^3)$ (ad esempio con Cholesky o LU).

Nell'*ILS* si utilizza la *formula di Sherman–Morrison–Woodbury (SMW)*, che permette di calcolare l'aggiornamento dell'inversa di una matrice A soggetta a una perturbazione di rango unitario con costo $\mathcal{O}(n^2)$:

$$(A + \mathbf{u}\mathbf{u}^\top)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{u}^\top A^{-1}}{1 + \mathbf{u}^\top A^{-1}\mathbf{u}}.$$

ILS per il Broad Learning System

Identità di Woodbury e formula di Sherman–Morrison–Woodbury (SWM).

Proposizione *(Identità di Woodbury)*: Siano $A \in \mathbb{R}^{n\times n}$ e $C \in \mathbb{R}^{k\times k}$ invertibili, $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$ e sia $S \equiv C^{-1} + VA^{-1}U$ invertibile.
Allora

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\,S^{-1}\,VA^{-1}.$$

Dimostrazione: omessa.

Corollario *(Formula SMW)*: Ponendo $k = 1$, $C = 1$, $U = \mathbf{u} \in \mathbb{R}^n$ e $V = \mathbf{u}^\top \in \mathbb{R}^{1\times n}$, si ottiene immediatamente:

$$(A + \mathbf{u}\mathbf{u}^\top)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{u}^\top A^{-1}}{1 + \mathbf{u}^\top A^{-1}\mathbf{u}}.$$

Dimostrazione: omessa.

ILS per il Broad Learning System

L'algoritmo ILS.

Algorithm 2: Pseudocodice Incremental Least Squares (ILS)

Input : Training set $T = \{(\mathbf{u}^i, d^i) \mid \mathbf{u}^i \in \mathbb{R}^n,\ d^i \in \mathbb{R},\ i = 1, \ldots, m\}$; $c_w > 0,\ c_\theta \geq 0$

Output: soluzione $\mathbf{x}_m$.

Inizializza H_0^{-1} e $\mathbf{b}_0$;

Poni $\mathbf{x}_0 \leftarrow H_0^{-1}\mathbf{b}_0, \quad k \leftarrow 1.$

while $k \leq m$ **do**

Definisci $\mathbf{a}_k \leftarrow (\mathbf{u}^k, 1)^\top$;

Calcola il vettore $\mathbf{s}_{k-1} \leftarrow H_{k-1}^{-1}\mathbf{a}_k$;

Aggiorna l'inversa con SMW $H_k^{-1} \leftarrow H_{k-1}^{-1} - \dfrac{\mathbf{s}_{k-1}\mathbf{s}_{k-1}^\top}{1 + \mathbf{a}_k^\top \mathbf{s}_{k-1}}$;

Calcola $\mathbf{v}_k \leftarrow H_k^{-1}\mathbf{a}_k$;

Aggiorna la soluzione $\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} + \mathbf{v}_k\left(d^k - \mathbf{a}_k^\top \mathbf{x}_{k-1}\right)$;

$k \leftarrow k + 1$.

return $\mathbf{x}_m$.

ILS per il Broad Learning System

Terzo esperimento: tabelle di confronto.

Task: partendo dal dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, addestrare una rete BLS in grado di prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	t_tr [s]
50	100	0.32161	0.29534	0.56711	0.54346	0.44238	0.42492	0.47608
50	500	0.30930	0.27090	0.55615	0.52048	0.42554	0.39477	6.96970
100	100	0.28364	0.24826	0.53258	0.49826	0.40225	0.37480	0.55406
100	500	0.27035	0.23345	0.51995	0.48317	0.38991	0.35883	7.51420
150	100	0.27030	0.23647	0.51990	0.48628	0.39027	0.36537	0.74217
150	500	0.24707	0.21026	0.49706	0.45854	0.36236	0.33085	8.33770
750	600	0.20757	0.16206	0.45561	0.40258	0.31375	0.27758	31.244

Tabella: ILS. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	t_tr [s]
50	100	0.31756	0.29545	0.56352	0.54356	0.44095	0.42504	98.587
50	500	0.30493	0.27088	0.55221	0.52046	0.42409	0.39474	678.72
100	100	0.27980	0.24828	0.52896	0.49827	0.40112	0.37481	127.21
100	500	0.26529	0.23350	0.51506	0.48322	0.38824	0.35880	786.44
150	100	0.26450	0.23631	0.51430	0.48611	0.38867	0.36536	182.07
150	500	0.24086	0.21027	0.49078	0.45855	0.36032	0.33099	933.54

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

ILS per il Broad Learning System

Terzo esperimento: considerazioni.

MSE in fase di testing sovrapponibili: per entrambi i modelli $\sim \mathcal{O}(2 * 10^{-1})$.

Il **tempo di training** risulta molto più dilatato per il metodo diretto (fino a $\mathbf{933s}$) rispetto al metodo incrementale (fino a $\mathbf{9s}$).

Il metodo diretto con pseudoinversa presenta **limiti intrinseci**: per essere applicato richiede la memorizzazione completa di A e di $A^\top A$. Al contrario, l'algoritmo incrementale conserva solo H^{-1} e pochi vettori ausiliari, risultando molto più parsimonioso in memoria ed applicabile a scenari on–line estesi.

Indice.

Incremental learning via mini–batch

Setting e derivazione algoritmo.

Per l'inizializzazione dell'algoritmo, si pongano

$$A_0 \equiv 0_{n+1 \times n+1}, \quad \mathbf{x_0} \equiv \mathbf{0}_{n+1}.$$

Si definisca

$$G_0 \equiv A_0^\top A_0 + \lambda I_{n+1} = \lambda I_{n+1}, \quad \Rightarrow \quad H_{\text{inv},0} \equiv G_0^{-1} = \frac{1}{\lambda} I_{n+1}. \tag{9}$$

Si supponga che all'iterazione corrente tutte le osservazioni siano state raccolte in A_{old}.

Si definisca

$$G_{\text{old}} \equiv A_{\text{old}}^\top A_{\text{old}} + \lambda I_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{10}$$

Incremental learning via mini–batch

Setting e derivazione algoritmo.

All'arrivo di un nuovo insieme di campioni, detto *mini-batch*, $A_b \equiv A_{\text{new}} \in \mathbb{R}^{m_b \times (n+1)}$ si ha

$$A_{\text{tot}} \equiv \begin{bmatrix} A_{\text{old}} \\ A_b \end{bmatrix}, \qquad G_{\text{tot}} \equiv A_{\text{tot}}^\top A_{\text{tot}} + \lambda I_{n+1},$$

da cui

$$G_{\text{tot}} = A_{\text{old}}^\top A_{\text{old}} + A_b^\top A_b + \lambda I_{n+1} = G_{\text{old}} + A_b^\top A_b.$$

Si può definire quindi

$$G_{\text{new}} \equiv G_{\text{old}} + A_b^\top A_b \qquad (G_{\text{new}} = G_{\text{tot}}).$$

Incremental learning via mini–batch

Setting e derivazione algoritmo.

Tramite l'identità di Woodbury per G_{new}, ponendo $A = G_{\text{old}},\ U = A_b^\top,\ C = I_{m_b},\ V = A_b$, si ottiene:

$$G_{\text{new}}^{-1} = G_{\text{old}}^{-1} - G_{\text{old}}^{-1} A_b^\top \left(I_{m_b} + A_b G_{\text{old}}^{-1} A_b^\top\right)^{-1} A_b G_{\text{old}}^{-1}. \tag{11}$$

Definendo le seguenti matrici

$$S \equiv I_{m_b} + A_b G_{\text{old}}^{-1} A_b^\top \in \mathbb{R}^{m_b \times m_b}, \quad K \equiv G_{\text{old}}^{-1} A_b^\top S^{-1} \in \mathbb{R}^{(n+1)\times m_b}$$

che risultano non singolari, l'aggiornamento dei pesi segue il passo tradizionale dell'ILS:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + K\big(d_b - A_b \mathbf{x}_{\text{old}}\big).$$

Osservazione: il vantaggio di questa proposta risiede nel fatto che, a ogni iterazione, è necessario invertire solamente la matrice S, quadrata di dimensione $m_b \times m_b$, operazione pienamente gestibile dal punto di vista computazionale.

Incremental learning via mini–batch

Algoritmo via mini-batch.

Algorithm 3: Incremental learning via mini–batch (ILMB)

Input : Training set $T = \{(\mathbf{u}^i, d^i)\}_{i=1}^{m}$ con $\mathbf{u}^i \in \mathbb{R}^p$; $N_f,\ N_e$; mini-batch size m_b.
Output: $\mathbf{x}_L = [\mathbf{w}, \theta] \in \mathbb{R}^{D+1}$, con $D = N_f + N_e$.

Inizializza $H_0^{-1} \in \mathbb{R}^{(D+1)\times(D+1)}$ e $\mathbf{x}_0 \in \mathbb{R}^{D+1}$;
Assegna $L \leftarrow \lceil m/m_b \rceil,\ b \leftarrow 1$;
while $b \leq L$ **do**
  Estrai $I_b = \{i_1, \ldots, i_{m_b}\}$;
  Assegna $\mathbf{d}_b \leftarrow \left[d^{i_1}, \ldots, d^{i_{m_b}}\right]^\top$;
  Assegna $U_b \leftarrow \left[(\mathbf{u}^{i_1})^\top, \ldots, (\mathbf{u}^{i_{m_b}})^\top\right]$;
  Calcola $Z_b^{\text{bls}} \leftarrow \sigma(U_b W_{\text{feat}} + \mathbf{1}_{m_b} b_{\text{feat}})$; $H_b^{\text{bls}} \leftarrow \sigma(Z_b^{\text{bls}} W_{\text{enha}} + \mathbf{1}_{m_b} b_{\text{enha}})$;
  Assegna $A_b \leftarrow \left[Z_b^{\text{bls}} \mid H_b^{\text{bls}} \mid \mathbf{1}_{m_b}\right]$;
  Calcola $S_b \leftarrow I_{m_b} + A_b H_{b-1}^{-1} A_b^\top$; $K_b \leftarrow H_{b-1}^{-1} A_b^\top S_b^{-1}$;
  Calcola $H_b^{-1} \leftarrow (I_{D+1} - K_b A_b)\, H_{b-1}^{-1}$;
  Assegna $\mathbf{x}_b \leftarrow \mathbf{x}_{b-1} + K_b(\mathbf{d}_b - A_b \mathbf{x}_{b-1})$;
  $b \leftarrow b + 1$;
return $\mathbf{x}_L$

Incremental learning via mini-batch

Quarto esperimento: tabelle di confronto.

Task: partendo dal dataset `WEC_Perth_Sydney.csv`, formato da circa 55000 campioni con 179 features, addestrare una rete in grado di prevedere la quantità totale di potenza prodotta dalle fabbriche di Perth e Sydney in base alle features.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094809	0.0098081	0.09737	0.099036	0.068701	0.069520	0.99014	11.96
200	300	0.0063861	0.0069698	0.079913	0.083485	0.055527	0.056666	0.99300	16.159
200	500	0.0067974	0.0072709	0.082446	0.085270	0.055713	0.057675	0.99269	24.440
300	200	0.0044335	0.0047447	0.066584	0.068882	0.045672	0.046710	0.99523	17.609
300	300	0.0034750	0.0041476	0.058950	0.064402	0.039822	0.041383	0.99583	20.361
300	500	0.0030551	0.0035103	0.055273	0.059248	0.037069	0.038063	0.99647	30.891
500	200	0.0020707	0.0024193	0.045505	0.049186	0.030003	0.030950	0.99757	25.144
500	300	0.0018186	0.0021151	0.042645	0.045991	0.027553	0.028802	0.99787	31.098
500	500	0.0016535	0.0021496	0.040663	0.046363	0.026471	0.028200	0.99784	44.724

Tabella: ILS, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094652	0.0097666	0.097289	0.098826	0.068686	0.069438	0.99019	0.97347
200	300	0.0063790	0.0069885	0.079868	0.083597	0.055515	0.056725	0.99298	1.2229
200	500	0.0067543	0.0072802	0.082184	0.085324	0.055597	0.057656	0.99268	2.5335
300	200	0.0044230	0.0047226	0.066506	0.068721	0.045557	0.046545	0.99525	1.2391
300	300	0.0034560	0.0041355	0.058787	0.064308	0.039726	0.041260	0.99584	1.8939
300	500	0.0030188	0.0034748	0.054944	0.058947	0.036822	0.037843	0.99651	3.1631
500	200	0.0020564	0.0024102	0.045348	0.049094	0.029878	0.030859	0.99758	2.4649
500	300	0.0018043	0.0021052	0.042477	0.045882	0.027427	0.028673	0.99788	3.1427
500	500	0.0016214	0.0021268	0.040267	0.046118	0.026161	0.027918	0.99786	4.5406

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini–batch

Quarto esperimento: considerazioni.

MSE nella fase di test sovrapponibili per i tre modelli: fino a $\sim \mathcal{O}(2 * 10^{-3})$. Anche le altre metriche risultano sovrapponibili.

Per il **tempo di training**, la rete con l'ILS richiede fino a **45 s** nelle configurazioni grandi. Con l'algoritmo ILMB, l'addestramento non supera i **5 s**.

Incremental learning via mini-batch

Quinto esperimento: tabelle di confronto.

Task: partendo dal dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, addestrare una rete BLS in grado di prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21366	0.20722	0.46223	0.45522	0.33793	0.33018	0.35412
200	300	0.21583	0.20717	0.46458	0.45516	0.34199	0.33223	0.49514
200	500	0.20416	0.19717	0.45184	0.44403	0.32571	0.31866	0.98949
500	200	0.18127	0.17754	0.42576	0.42135	0.30297	0.29758	0.96841
500	300	0.17846	0.17521	0.42244	0.41858	0.29954	0.29514	1.2507
500	500	0.17361	0.17054	0.41666	0.41297	0.29349	0.28955	1.7692
750	650	0.15926	0.16033	0.39907	0.40041	0.27626	0.27601	3.6874
750	700	0.15956	0.16285	0.39945	0.40354	0.27605	0.27717	3.7549
750	1200	0.15273	0.15601	0.39081	0.39498	0.26731	0.26827	6.5469
1000	650	0.15188	0.15774	0.38971	0.39717	0.26687	0.27087	4.6598
1000	700	0.15038	0.15462	0.38779	0.39321	0.26461	0.26753	4.8194
1000	1200	0.14582	0.15080	0.38187	0.38833	0.25926	0.26409	8.8745

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21353	0.20719	0.46209	0.45518	0.33762	0.33002	4.6215
200	300	0.21582	0.20717	0.46456	0.45516	0.34185	0.33213	6.4389
200	500	0.20414	0.19718	0.45182	0.44405	0.32565	0.31861	9.7225
500	200	0.18114	0.17750	0.42561	0.42131	0.30278	0.29761	10.151
500	300	0.17851	0.17531	0.42250	0.41870	0.29973	0.29528	12.609
500	500	0.17372	0.17082	0.41679	0.41330	0.29356	0.28976	18.272
750	650	0.15927	0.16068	0.39909	0.40085	0.27635	0.27626	34.259
750	700	0.15965	0.16339	0.39956	0.40421	0.27636	0.27783	37.701
750	1200	0.15284	0.15626	0.39095	0.39530	0.26752	0.26868	73.340
1000	650	0.15193	0.15777	0.38978	0.39721	0.26685	0.27084	51.032
1000	700	0.15043	0.15506	0.38785	0.39378	0.26472	0.26793	54.584
1000	1200	0.14580	0.15072	0.38183	0.38822	0.25951	0.26408	103.920

Tabella: ILS, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini-batch.

Quinto esperimento: considerazioni.

Gli **MSE** in fase di testing sono comparabili: $\sim \mathcal{O}(10^{-1})$.

Per il **tempo di training**, il modello con l'ILS richiede da **34 s** a **104 s** nelle configurazioni grandi. L'algoritmo ILMB non supera i **9 s**, permettendo di aumentare le dimensioni della rete, con conseguente miglioramento delle prestazioni.

Grazie a tutti per l’attenzione!

Bibliografia

Principali referenze.

C. L. P. Chen and Z. Liu, “Broad Learning System: An Effective and Efficient Incremental Learning System Without the Need for Deep Architecture,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 10–24, 2018.

A. Cassioli, A. Chiavaioli, C. Manes, and M. Sciandrone, “An Incremental Least Squares Algorithm for Large Scale Linear Classification,” *European Journal of Operational Research*, vol. 224, no. 3, pp. 560–565, 2013.

CGM per il Broad Learning System

Primo esperimento: tabelle di confronto.

Task: partendo dal dataset `slice_localization_data.csv`, formato da circa 54000 campioni e 386 features, addestrare una rete che, data la fotografia di una parte del corpo, preveda la corretta posizione.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max
100	1000	0.1	0.12114	0.13230	0.13252	0.44427	0.86777	0.99966
100	10000	0.1	0.00862	0.03029	0.03196	68.7910	0.96788	0.99981
200	1000	0.1	0.10666	0.11368	0.11522	0.46234	0.88503	1.00000
200	10000	0.1	0.00878	0.02951	0.03075	82.4210	0.96909	1.00000
400	1000	0.1	0.08118	0.09141	0.09137	0.61586	0.90883	1.00000
400	10000	0.1	0.00746	0.02747	0.02804	81.6560	0.97181	1.00000
500	1000	0.1	0.07482	0.08568	0.08641	0.71903	0.91377	1.00000
500	10000	0.1	0.00760	0.02644	0.02744	85.1320	0.97243	1.00000
1000	1000	0.1	0.05696	0.06610	0.06724	1.32270	0.93291	1.00000
1000	10000	0.1	0.00692	0.02737	0.02734	99.2900	0.97253	1.00000

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max	ifail
100	1000	0.1	0.11231	0.12389	0.12453	1.9751	0.87566	0.99976	3
100	10000	0.1	0.01338	0.02718	0.02742	18.450	0.97262	0.99982	3
200	1000	0.1	0.09199	0.10195	0.10316	2.0541	0.89701	1.00000	3
200	10000	0.1	0.01122	0.02366	0.02402	18.703	0.97602	1.00000	3
400	1000	0.1	0.07819	0.08782	0.08849	2.5851	0.91165	1.00000	3
400	10000	0.1	0.01113	0.02157	0.02199	18.274	0.97804	1.00000	2
500	1000	0.1	0.07298	0.08279	0.08495	2.7847	0.91518	1.00000	3
500	10000	0.1	0.01171	0.02267	0.02373	17.863	0.97631	1.00000	2
1000	1000	0.1	0.05830	0.06993	0.06903	3.5915	0.93108	1.00000	2
1000	10000	0.1	0.01258	0.02355	0.02399	15.040	0.97604	1.00000	2

Tabella: CGM. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Primo esperimento: tabelle di confronto.

Task: partendo dal dataset `slice_localization_data.csv`, formato da circa 54000 campioni e 386 features, addestrare una rete che, data la fotografia di una parte del corpo, preveda la corretta posizione.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max
100	1000	0.1	0.12114	0.13230	0.13252	0.44427	0.86777	0.99966
100	10000	0.1	0.00862	0.03029	0.03196	68.7910	0.96788	0.99981
200	1000	0.1	0.10666	0.11368	0.11522	0.46234	0.88503	1.00000
200	10000	0.1	0.00878	0.02951	0.03075	82.4210	0.96909	1.00000
400	1000	0.1	0.08118	0.09141	0.09137	0.61586	0.90883	1.00000
400	10000	0.1	0.00746	0.02747	0.02804	81.6560	0.97181	1.00000
500	1000	0.1	0.07482	0.08568	0.08641	0.71903	0.91377	1.00000
500	10000	0.1	0.00760	0.02644	0.02744	85.1320	0.97243	1.00000
1000	1000	0.1	0.05696	0.06610	0.06724	1.32270	0.93291	1.00000
1000	10000	0.1	0.00692	0.02737	0.02734	99.2900	0.97253	1.00000

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max	ifail
100	1000	0.1	0.11231	0.12389	0.12453	1.9751	0.87566	0.99976	3
100	10000	0.1	0.01338	0.02718	0.02742	18.450	0.97262	0.99982	3
200	1000	0.1	0.09199	0.10195	0.10316	2.0541	0.89701	1.00000	3
200	10000	0.1	0.01122	0.02366	0.02402	18.703	0.97602	1.00000	3
400	1000	0.1	0.07819	0.08782	0.08849	2.5851	0.91165	1.00000	3
400	10000	0.1	0.01113	0.02157	0.02199	18.274	0.97804	1.00000	2
500	1000	0.1	0.07298	0.08279	0.08495	2.7847	0.91518	1.00000	3
500	10000	0.1	0.01171	0.02267	0.02373	17.863	0.97631	1.00000	2
1000	1000	0.1	0.05830	0.06993	0.06903	3.5915	0.93108	1.00000	2
1000	10000	0.1	0.01258	0.02355	0.02399	15.040	0.97604	1.00000	2

Tabella: CGM. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Primo esperimento: considerazioni.

- **Accuratezza**: miglior **Mean Squared Error (MSE)** in fase di testing $\sim \mathbf{0.022}$ ottenuto mediante **CGM** nel caso di rete con $N_f = 400$, $N_e = 10000$;
  nel modello con **pseudoinversa**, $\mathrm{MSE\_test} \sim \mathbf{0.028}$ nel modello con $N_f = 1000$, $N_e = 10000$. I risultati restano tutti sovrapponibili.
- **Tempo di training**: nelle reti con `enha` = 10000, **CGM** $\sim 15-19$ s vs **diretto** $\sim 82-99$ s $\Rightarrow$ $\sim$ **5** volte più veloce.
- **Coefficiente $\mathbf{R^2}$**: **CGM** fino a $\mathbf{0.978}$; **diretto** fino a $\mathbf{0.973}$. Sono pressoché equivalenti.

CGM per il Broad Learning System

Secondo esperimento: tabelle di confronto.

Task: addestrare una rete BLS sul dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, per prevedere la superconduttività di alcuni materiali metallici.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max
100	1000	0.1	0.15485	0.18026	0.17894	0.25331	0.83045	0.99712
100	10000	0.1	0.07470	0.17384	0.17882	68.688	0.83056	0.99848
200	1000	0.1	0.14736	0.17370	0.19101	1.5427	0.81901	0.99983
200	10000	0.1	0.05900	0.28019	0.29295	260.51	0.72241	0.99994
400	1000	0.1	0.13713	0.17984	0.17948	0.36409	0.82993	1.00000
400	10000	0.1	0.04801	0.62696	0.65926	73.230	0.37531	1.00000
500	1000	0.1	0.13393	0.17535	0.18504	0.40398	0.82466	1.00000
500	10000	0.1	0.04589	0.76072	0.55349	75.693	0.47553	1.00000
1000	1000	0.1	0.11451	0.19200	0.18974	0.83356	0.82021	1.00000
1000	10000	0.1	0.03838	1.37600	1.39100	178.63	-0.31805	1.00000

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max	ifail
100	1000	0.1	0.16583	0.18268	0.18317	1.0808	0.82644	0.99712	3
100	10000	0.1	0.13315	0.15379	0.15764	7.9993	0.85063	0.99848	3
200	1000	0.1	0.15335	0.17476	0.18913	1.0242	0.82079	0.99983	3
200	10000	0.1	0.11460	0.14430	0.15241	8.1823	0.85558	0.99994	3
400	1000	0.1	0.14962	0.17444	0.17626	1.2163	0.83298	1.00000	3
400	10000	0.1	0.10347	0.13950	0.14774	8.2021	0.86001	1.00000	2
500	1000	0.1	0.14067	0.16994	0.18018	1.2388	0.82926	1.00000	3
500	10000	0.1	0.10056	0.13838	0.14368	8.3084	0.86386	1.00000	2
1000	1000	0.1	0.13975	0.16588	0.16765	1.4356	0.84114	1.00000	2
1000	10000	0.1	0.09124	0.13187	0.13792	8.6651	0.86931	1.00000	2

Tabella: CGM. Intel Core i7-13650HX NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Secondo esperimento: tabelle di confronto.

Task: addestrare una rete BLS sul dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, per prevedere la superconduttività di alcuni materiali metallici.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max
100	1000	0.1	0.15485	0.18026	0.17894	0.25331	0.83045	0.99712
100	10000	0.1	0.07470	0.17384	0.17882	68.688	0.83056	0.99848
200	1000	0.1	0.14736	0.17370	0.19101	1.5427	0.81901	0.99983
200	10000	0.1	0.05900	0.28019	0.29295	260.51	0.72241	0.99994
400	1000	0.1	0.13713	0.17984	0.17948	0.36409	0.82993	1.00000
400	10000	0.1	0.04801	0.62696	0.65926	73.230	0.37531	1.00000
500	1000	0.1	0.13393	0.17535	0.18504	0.40398	0.82466	1.00000
500	10000	0.1	0.04589	0.76072	0.55349	75.693	0.47553	1.00000
1000	1000	0.1	0.11451	0.19200	0.18974	0.83356	0.82021	1.00000
1000	10000	0.1	0.03838	1.37600	1.39100	178.63	-0.31805	1.00000

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	λ	MSE_tr	MSE_val	MSE_te	t_tr [s]	R^2_te	H_max	ifail
100	1000	0.1	0.16583	0.18268	0.18317	1.0808	0.82644	0.99712	3
100	10000	0.1	0.13315	0.15379	0.15764	7.9993	0.85063	0.99848	3
200	1000	0.1	0.15335	0.17476	0.18913	1.0242	0.82079	0.99983	3
200	10000	0.1	0.11460	0.14430	0.15241	8.1823	0.85558	0.99994	3
400	1000	0.1	0.14962	0.17444	0.17626	1.2163	0.83298	1.00000	3
400	10000	0.1	0.10347	0.13950	0.14774	8.2021	0.86001	1.00000	2
500	1000	0.1	0.14067	0.16994	0.18018	1.2388	0.82926	1.00000	3
500	10000	0.1	0.10056	0.13838	0.14368	8.3084	0.86386	1.00000	2
1000	1000	0.1	0.13975	0.16588	0.16765	1.4356	0.84114	1.00000	2
1000	10000	0.1	0.09124	0.13187	0.13792	8.6651	0.86931	1.00000	2

Tabella: CGM. Intel Core i7-13650HX NVIDIA GeForce RTX 4060 - 32GB RAM.

CGM per il Broad Learning System

Secondo esperimento: considerazioni.

- **Accuratezza**: il miglior **MSE** sul test si ottiene con il modello addestrato via **CGM** ed è pari a $\mathbf{0.138}$, con $\mathbf{R^2 = 0.869}$; il modello con **pseudoinversa** si ferma a $\sim \mathbf{0.179}$ e degrada fortemente all'aumentare dei nodi.
  Nel caso $N_f = 1000,\ N_e = 10000$, il MSE del CGM è *un ordine di grandezza* migliore rispetto al diretto (0.137 vs 1.391), a conferma della qualità del metodo.
- **Tempo di training**: utilizzando il **CGM**, i tempi di training non superano gli **8 s**; il metodo diretto invece, oltre ad essere meno accurato, è anche molto meno efficiente: $\sim \mathbf{69 - 180}$ **s**.
- **Coefficiente $\mathbf{R^2}$**: **CGM** $\sim \mathbf{0.87}$; metodo **diretto** $\sim \mathbf{0.831}$ con crollo a valori negativi all'aumentare dei nodi della rete.
- L'utilizzo dell'**early stopping** permette di terminare l'addestramento quando l'errore sul validation set non migliora e selezionare l'epoca ottimale. In questo modo, si evita overfitting e si tengono i tempi di training contenuti promuovendo anche il risparmio di risorse.

ILS per il Broad Learning System

Terzo esperimento: tabelle di confronto.

Task: addestrare una rete BLS sul dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, per prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr_end	MSE_te	RMSE_tr_end	RMSE_te	MAE_tr_end	MAE_te	t_tr [s]
50	100	0.32161	0.29534	0.56711	0.54346	0.44238	0.42492	0.47608
50	500	0.30930	0.27090	0.55615	0.52048	0.42554	0.39477	6.96970
100	100	0.28364	0.24826	0.53258	0.49826	0.40225	0.37480	0.55406
100	500	0.27035	0.23345	0.51995	0.48317	0.38991	0.35883	7.51420
150	100	0.27030	0.23647	0.51990	0.48628	0.39027	0.36537	0.74217
150	500	0.24707	0.21026	0.49706	0.45854	0.36236	0.33085	8.33770
750	600	0.20757	0.16206	0.45561	0.40258	0.31375	0.27758	31.244

Tabella: ILS. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 RTX 4060 — 32GB RAM.

feat	enha	MSE_tr_end	MSE_te	RMSE_tr_end	RMSE_te	MAE_tr_end	MAE_te	t_tr [s]
50	100	0.31756	0.29545	0.56352	0.54356	0.44095	0.42504	98.587
50	500	0.30493	0.27088	0.55221	0.52046	0.42409	0.39474	678.72
100	100	0.27980	0.24828	0.52896	0.49827	0.40112	0.37481	127.21
100	500	0.26529	0.23350	0.51506	0.48322	0.38824	0.35880	786.44
150	100	0.26450	0.23631	0.51430	0.48611	0.38867	0.36536	182.07
150	500	0.24086	0.21027	0.49078	0.45855	0.36032	0.33099	933.54

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 RTX 4060 — 32GB RAM.

ILS per il Broad Learning System

Terzo esperimento: tabelle di confronto.

Task: addestrare una rete BLS sul dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, per prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr_end	MSE_te	RMSE_tr_end	RMSE_te	MAE_tr_end	MAE_te	t_tr [s]
50	100	0.32161	0.29534	0.56711	0.54346	0.44238	0.42492	0.47608
50	500	0.30930	0.27090	0.55615	0.52048	0.42554	0.39477	6.96970
100	100	0.28364	0.24826	0.53258	0.49826	0.40225	0.37480	0.55406
100	500	0.27035	0.23345	0.51995	0.48317	0.38991	0.35883	7.51420
150	100	0.27030	0.23647	0.51990	0.48628	0.39027	0.36537	0.74217
150	500	0.24707	0.21026	0.49706	0.45854	0.36236	0.33085	8.33770
750	600	0.20757	0.16206	0.45561	0.40258	0.31375	0.27758	31.244

Tabella: ILS. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr_end	MSE_te	RMSE_tr_end	RMSE_te	MAE_tr_end	MAE_te	t_tr [s]
50	100	0.31756	0.29545	0.56352	0.54356	0.44095	0.42504	98.587
50	500	0.30493	0.27088	0.55221	0.52046	0.42409	0.39474	678.72
100	100	0.27980	0.24828	0.52896	0.49827	0.40112	0.37481	127.21
100	500	0.26529	0.23350	0.51506	0.48322	0.38824	0.35880	786.44
150	100	0.26450	0.23631	0.51430	0.48611	0.38867	0.36536	182.07
150	500	0.24086	0.21027	0.49078	0.45855	0.36032	0.33099	933.54

Tabella: pseudoinversa. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

ILS per il Broad Learning System

Terzo esperimento: considerazioni.

- **Accuratezza**: il miglior valore dell'**MSE** in fase di test è $\sim \mathbf{0.21}$ per entrambi i modelli nel caso $N_e = 150$, $N_f = 500$; tuttavia l'approccio **ILS** risulta nettamente più efficiente.
  Nelle altre combinazioni di nodi, prestazioni comparabili ma tempi di training molto diversi, a favore di **ILS**.
- **Tempo di training**: a parità di **MSE**, il metodo **diretto** richiede **933.54 s**, mentre **ILS** solamente **8 s**.
- **Altre metriche**: anche **MAE** e **RMSE** risultano sovrapponibili.
- **Commenti**: l'ILS consente di aumentare la dimensione del modello senza penalizzare i tempi di training, rendendolo adatto a applicazioni reali on-line e su larga scala. Consente inoltre, all'arrivo di nuovi campioni, l'aggiornamento efficiente di un modello già addestrato.

Incremental learning via mini-batch

Quarto esperimento: tabelle di confronto.

Task: partendo dal dataset `WEC_Perth_Sydney.csv`, formato da circa 55000 campioni con 179 features, addestrare una rete in grado di prevedere la quantità totale di potenza prodotta dalle fabbriche di Perth e Sydney in base alle features.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094809	0.0098081	0.09737	0.099036	0.068701	0.069520	0.99014	11.96
200	300	0.0063861	0.0069698	0.079913	0.083485	0.055527	0.056666	0.99300	16.159
200	500	0.0067974	0.0072709	0.082446	0.085270	0.055713	0.057675	0.99269	24.440
300	200	0.0044335	0.0047447	0.066584	0.068882	0.045672	0.046710	0.99523	17.609
300	300	0.0034750	0.0041476	0.058950	0.064402	0.039822	0.041383	0.99583	20.361
300	500	0.0030551	0.0035103	0.055273	0.059248	0.037069	0.038063	0.99647	30.891
500	200	0.0020707	0.0024193	0.045505	0.049186	0.030003	0.030950	0.99757	25.144
500	300	0.0018186	0.0021151	0.042645	0.045991	0.027553	0.028802	0.99787	31.098
500	500	0.0016535	0.0021496	0.040663	0.046363	0.026471	0.028200	0.99784	44.724

Tabella: ILS mini-batch, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094652	0.0097666	0.097289	0.098826	0.068686	0.069438	0.99019	0.97347
200	300	0.0063790	0.0069885	0.079868	0.083597	0.055515	0.056725	0.99298	1.2229
200	500	0.0067543	0.0072802	0.082184	0.085324	0.055597	0.057656	0.99268	2.5335
300	200	0.0044230	0.0047226	0.066506	0.068721	0.045557	0.046545	0.99525	1.2391
300	300	0.0034560	0.0041355	0.058787	0.064308	0.039726	0.041260	0.99584	1.8939
300	500	0.0030188	0.0034748	0.054944	0.058947	0.036822	0.037843	0.99651	3.1631
500	200	0.0020564	0.0024102	0.045348	0.049094	0.029878	0.030859	0.99758	2.4649
500	300	0.0018043	0.0021052	0.042477	0.045882	0.027427	0.028673	0.99788	3.1427
500	500	0.0016214	0.0021268	0.040267	0.046118	0.026161	0.027918	0.99786	4.5406

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini-batch

Quarto esperimento: tabelle di confronto.

Task: partendo dal dataset `WEC_Perth_Sydney.csv`, formato da circa 55000 campioni con 179 features, addestrare una rete in grado di prevedere la quantità totale di potenza prodotta dalle fabbriche di Perth e Sydney in base alle features.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094809	0.0098081	0.09737	0.099036	0.068701	0.069520	0.99014	11.96
200	300	0.0063861	0.0069698	0.079913	0.083485	0.055527	0.056666	0.99300	16.159
200	500	0.0067974	0.0072709	0.082446	0.085270	0.055713	0.057675	0.99269	24.440
300	200	0.0044335	0.0047447	0.066584	0.068882	0.045672	0.046710	0.99523	17.609
300	300	0.0034750	0.0041476	0.058950	0.064402	0.039822	0.041383	0.99583	20.361
300	500	0.0030551	0.0035103	0.055273	0.059248	0.037069	0.038063	0.99647	30.891
500	200	0.0020707	0.0024193	0.045505	0.049186	0.030003	0.030950	0.99757	25.144
500	300	0.0018186	0.0021151	0.042645	0.045991	0.027553	0.028802	0.99787	31.098
500	500	0.0016535	0.0021496	0.040663	0.046363	0.026471	0.028200	0.99784	44.724

Tabella: ILS mini-batch, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr	MSE_te	RMSE_tr	RMSE_te	MAE_tr	MAE_te	R2_te	t_tr [s]
200	200	0.0094652	0.0097666	0.097289	0.098826	0.068686	0.069438	0.99019	0.97347
200	300	0.0063790	0.0069885	0.079868	0.083597	0.055515	0.056725	0.99298	1.2229
200	500	0.0067543	0.0072802	0.082184	0.085324	0.055597	0.057656	0.99268	2.5335
300	200	0.0044230	0.0047226	0.066506	0.068721	0.045557	0.046545	0.99525	1.2391
300	300	0.0034560	0.0041355	0.058787	0.064308	0.039726	0.041260	0.99584	1.8939
300	500	0.0030188	0.0034748	0.054944	0.058947	0.036822	0.037843	0.99651	3.1631
500	200	0.0020564	0.0024102	0.045348	0.049094	0.029878	0.030859	0.99758	2.4649
500	300	0.0018043	0.0021052	0.042477	0.045882	0.027427	0.028673	0.99788	3.1427
500	500	0.0016214	0.0021268	0.040267	0.046118	0.026161	0.027918	0.99786	4.5406

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini–batch

Quarto esperimento: considerazioni.

- **Accuratezza**: il miglior valore dell'**MSE_test** è $\sim \mathbf{0.0021}$ per entrambi i modelli nel caso $N_f = 500$, $N_e = 300$.
- **Tempo di training**: a parità di **MSE**, il modello con ILS richiede fino a **45 s**, mentre con l'algoritmo ILMB non si superano i **5 s**.
- **Altre metriche**: anche **MAE** e **RMSE** risultano sovrapponibili.
- **Commenti**: entrambi i modelli sono applicabili con profitto in tempi sostenibili. L'ILMB tuttavia potrebbe essere applicato in contesti reali on–line e a larga scala con maggior efficienza.

Incremental learning via mini–batch

Quinto esperimento: tabelle di confronto.

Task: partendo dal dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, addestrare una rete BLS in grado di prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21353	0.20719	0.46209	0.45518	0.33762	0.33002	4.6215
200	300	0.21582	0.20717	0.46456	0.45516	0.34185	0.33213	6.4389
200	500	0.20414	0.19718	0.45182	0.44405	0.32565	0.31861	9.7225
500	200	0.18114	0.17750	0.42561	0.42131	0.30278	0.29761	10.151
500	300	0.17851	0.17531	0.42250	0.41870	0.29973	0.29528	12.609
500	500	0.17372	0.17082	0.41679	0.41330	0.29356	0.28976	18.272
750	650	0.15927	0.16068	0.39909	0.40085	0.27635	0.27626	34.259
750	700	0.15965	0.16339	0.39956	0.40421	0.27636	0.27783	37.701
750	1200	0.15284	0.15626	0.39095	0.39530	0.26752	0.26868	73.340
1000	650	0.15193	0.15777	0.38978	0.39721	0.26685	0.27084	51.032
1000	700	0.15043	0.15506	0.38785	0.39378	0.26472	0.26793	54.584
1000	1200	0.14580	0.15072	0.38183	0.38822	0.25951	0.26408	103.920

Tabella: ILS mini–batch, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21366	0.20722	0.46223	0.45522	0.33793	0.33018	0.35412
200	300	0.21583	0.20717	0.46458	0.45516	0.34199	0.33223	0.49514
200	500	0.20416	0.19717	0.45184	0.44403	0.32571	0.31866	0.98949
500	200	0.18127	0.17754	0.42576	0.42135	0.30297	0.29758	0.96841
500	300	0.17846	0.17521	0.42244	0.41858	0.29954	0.29514	1.2507
500	500	0.17361	0.17054	0.41666	0.41297	0.29349	0.28955	1.7692
750	650	0.15926	0.16033	0.39907	0.40041	0.27626	0.27601	3.6874
750	700	0.15956	0.16285	0.39945	0.40354	0.27605	0.27717	3.7549
750	1200	0.15273	0.15601	0.39081	0.39498	0.26731	0.26827	6.5469
1000	650	0.15188	0.15774	0.38971	0.39717	0.26687	0.27087	4.6598
1000	700	0.15038	0.15462	0.38779	0.39321	0.26461	0.26753	4.8194
1000	1200	0.14582	0.15080	0.38187	0.38833	0.25926	0.26409	8.8745

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini–batch

Quinto esperimento: tabelle di confronto.

Task: partendo dal dataset `superconductivity_data.csv`, formato da circa 22000 campioni e 91 features, addestrare una rete BLS in grado di prevedere la superconduttività di alcuni materiali metallici.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21353	0.20719	0.46209	0.45518	0.33762	0.33002	4.6215
200	300	0.21582	0.20717	0.46456	0.45516	0.34185	0.33213	6.4389
200	500	0.20414	0.19718	0.45182	0.44405	0.32565	0.31861	9.7225
500	200	0.18114	0.17750	0.42561	0.42131	0.30278	0.29761	10.151
500	300	0.17851	0.17531	0.42250	0.41870	0.29973	0.29528	12.609
500	500	0.17372	0.17082	0.41679	0.41330	0.29356	0.28976	18.272
750	650	0.15927	0.16068	0.39909	0.40085	0.27635	0.27626	34.259
750	700	0.15965	0.16339	0.39956	0.40421	0.27636	0.27783	37.701
750	1200	0.15284	0.15626	0.39095	0.39530	0.26752	0.26868	73.340
1000	650	0.15193	0.15777	0.38978	0.39721	0.26685	0.27084	51.032
1000	700	0.15043	0.15506	0.38785	0.39378	0.26472	0.26793	54.584
1000	1200	0.14580	0.15072	0.38183	0.38822	0.25951	0.26408	103.920

Tabella: ILS mini–batch, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

feat	enha	MSE_tr_full	MSE_te	RMSE_tr_full	RMSE_te	MAE_tr_full	MAE_te	tr_time [s]
200	200	0.21366	0.20722	0.46223	0.45522	0.33793	0.33018	0.35412
200	300	0.21583	0.20717	0.46458	0.45516	0.34199	0.33223	0.49514
200	500	0.20416	0.19717	0.45184	0.44403	0.32571	0.31866	0.98949
500	200	0.18127	0.17754	0.42576	0.42135	0.30297	0.29758	0.96841
500	300	0.17846	0.17521	0.42244	0.41858	0.29954	0.29514	1.2507
500	500	0.17361	0.17054	0.41666	0.41297	0.29349	0.28955	1.7692
750	650	0.15926	0.16033	0.39907	0.40041	0.27626	0.27601	3.6874
750	700	0.15956	0.16285	0.39945	0.40354	0.27605	0.27717	3.7549
750	1200	0.15273	0.15601	0.39081	0.39498	0.26731	0.26827	6.5469
1000	650	0.15188	0.15774	0.38971	0.39717	0.26687	0.27087	4.6598
1000	700	0.15038	0.15462	0.38779	0.39321	0.26461	0.26753	4.8194
1000	1200	0.14582	0.15080	0.38187	0.38833	0.25926	0.26409	8.8745

Tabella: ILMB, $m_b = 25$. Intel Core i7-13650HX, NVIDIA GeForce RTX 4060 - 32GB RAM.

Incremental learning via mini-batch

Quinto esperimento: considerazioni.

- **Accuratezza**: il miglior **MSE**, pari a $\sim$ **0.151**, si ottiene mediante una rete con $N_f = 1000$, $N_e = 1200$. A parità di nodi, gli MSE dei due modelli sono sovrapponibili.
- **Tempo di training**: a parità di nodi, il modello con ILS richiede dai **6 s** ai **100 s**. Con l'algoritmo ILMB il tempo si riduce notevolmente: non si superano infatti i **9 s**.
- **Altre metriche**: **MAE** e **RMSE** risultano sostanzialmente sovrapponibili lungo tutta la griglia.
- **Commenti**: ILMB permette di aumentare le dimensioni della rete, con conseguente miglioramento delle prestazioni, diventando potenzialmente applicabile in contesti reali di apprendimento on–line o large scale, contesti in cui, il metodo con pseudoinversa non è una via percorribile per motivi intrinseci.