Progetto OBD:
Implementazione di una rete neurale

Studente: Luca Sugamosto, matricola 0324613

Professore: Andrea Cristofari

Sommario

Istruzioni per eseguire il codice

Nella cartella “Progetto OBD” sono presenti due file MATLAB Code: “neuralNetwork.m” e “createNeuralNetwork.m”.

Il file “neuralNetwork.m” corrisponde alla funzione principale, infatti, in esso avviene il caricamento dei dati, la creazione ed inizializzazione dei data sets e la chiamata alle funzioni da eseguire. Mentre il file “createNeuralNetwork.m” è una classe in cui sono definite ed inizializzate tutte le variabili e tutte le funzioni utili ai fini del progetto.

Per eseguire il codice bisogna aprire il file “neuralNetwork.m” e cliccare “RUN”. Se si desidera effettuare prove differenti bisogna prima modificare il seme (quindi l’input della funzione “rng ()”) così da generare diversi numeri casuali nelle matrici dei pesi e cambiare l’ordine dei campioni nel data set.

Non bisogna fare nessun altro passaggio, poiché gli altri algoritmi / normalizzazioni / parametri di learning rate sono presenti e commentati.

Cosa si trova nel codice

Nel codice troverà tutte le variabili e le funzioni utili all’implementazioni di una serie di algoritmi provati e testati passo dopo passo e confrontati gli uni con gli altri:

- Metodo del gradiente stocastico con passo costante;
- Metodo del gradiente stocastico con Diminishing step-size;
- Metodo del gradiente batch con passo costante;
- Due metodi di normalizzazione diversi.

Alla fine, ho deciso di commentare tutto il resto e lasciare come variabili e funzioni attive quelle che implementano il METODO DEL GRADIENTE STOCASTICO CON PASSO COSTANTE e come normalizzazione la STANDARDIZZAZIONE.

Scelte progettuali

Scelta dell’algoritmo usato

L’algoritmo utilizzato per aggiornare i pesi e quindi realizzare una rete neurale ottima, si compone delle seguenti caratteristiche tecniche:

- La **direzione di discesa d_k** è uguale all’anti gradiente della funzione obiettivo; questa scelta garantisce il decremento della funzione obiettivo ed in particolare è anche la direzione di discesa più ripida.
- Il **passo di aggiornamento α_k** è scelto costante e nell’ordine dei millesimi; in questo modo ho un learning rate sufficientemente piccolo che permette di aggiornare le matrici dei pesi W e i vettori dei bias B senza causare underfitting e overfitting. Inoltre, si evita di far tendere i valori di W, B a $\pm\infty$ (cosa che succede con α nell’ordine dei decimi o centesimi).
- La **funzione di attivazione** scelta negli strati interni della rete è la “ReLu”, mentre per quanto riguarda l’ultimo strato nascosto si ha:

Per problemi di classificazione multi-classe utilizzo ancora la “ReLu”, mentre per problemi di classificazione binaria ho deciso di non usarne nessuna, questo perché ho notato che a parità di ACCURACY riesco ad ottenere una LOSS più bassa.

- Per quanto riguarda la **dimensione del batch m**, ho scelto il metodo stocastico. Questa scelta è dovuta al fatto che una dimensione più piccola del batch permette di ottenere maggiore stabilità (lo si nota nel comportamento a regime dell’accuratezza e della perdita) e un valore molto più alto di accuratezza (parallelamente un valore più basso di perdita). Unica nota positiva del metodo batch notata durante i test è che con essi si impiega la metà del tempo, rispetto al metodo stocastico.
- La **normalizzazione** scelta ed applicata al data set iniziale è la standardizzazione (normalizzazione con media e varianza); questa è stata preferita rispetto alla normalizzazione con massimo e minimo in quanto risolve alcuni casi in cui tornavano valori di accuratezza e di perdita costanti dall’inizio alla fine (dovuti probabilmente al fatto di essere in un punto di minimo o di flesso e quindi la derivata è nulla).

Scelta del valore dei parametri

Il data set utilizzato ha **8** features e **2** classi quindi si tratta di un problema di classificazione binaria. I campioni del data set sono in totale **216’958**.

La rete neurale creata è costituita da due strati nascosti (L = **2**), il primo strato nascosto ha **10** neuroni ed il secondo strato nascosto ne ha **6**. Questa scelta mi ha permesso di creare una rete che in tempi brevi riesce a restituire dei risultati garantendo allo stesso tempo ottimi valori di accuratezza e perdita.

Il numero di epoche scelto e salvato nel file “createNeuralNetwork.m” è pari a **25**, questo perché effettuando varie prove si evince che nella maggior parte dei casi l’andamento della rete sia già stabile entro questo numero di epoche.

La dimensione del batch considerata ad ogni iterazione è pari a **16**. Nonostante questo sia un valore molto piccolo che causa un rallentamento della rete neurale nel calcolo del risultato, è preferibile in quanto riesce a garantirmi un’ottima stabilità a regime e valori di accuratezza alti (valori di perdita bassi).

Il passo di aggiornamento (learning rate) α è preso costante e pari a **0.005**. Questa scelta mi ha permesso di ottenere aggiornamenti ottimali durante tutte le iterazioni, a differenza del Diminishing step-size; infatti, usando quest’ultimo in alcuni test (con numero di epoche alto) si è notato che, se si partisse male (bassa accuratezza) difficilmente si riuscirebbe a raggiungere un valore alto in quanto il passo di aggiornamento si riduce nel tempo e da poca importanza allo spostamento calcolato. Invece, per quanto riguarda il suo valore, questo permette di evitare casi estremi come, per esempio, poca differenza tra accuratezza iniziale e finale se α preso troppo piccolo e ritorno di errori dovuti a valori che tendevano a $\pm\infty$ se α preso troppo grande.

Test ed osservazioni finali

Tutti i test riportati di seguito sono stati eseguiti con il seguente dispositivo:

- Samsung Galaxy Book 3, processore i7-1355U 1,70 GHz, 16 GB di RAM DDR4.

rng	FIRST HIDDEN LAYER	SECOND HIDEN LAYER	EPOCHS	DIM. MINI-BATCH (m)	LEARNING RATE (α)	ANDAMENTO ACCURACY	VALORE ACCURACY (%)
1	10	8	25	32	0,0100	Costante	33,33%
2	10	8	25	32	0,0100	Crescente	92,42% -> 95,84%
3	10	8	25	32	0,0100	Crescente	94,07% -> 95,72%
4	10	8	25	32	0,0100	Costante	33,33%
5	10	8	25	32	0,0100	Crescente	65,93% -> 95,70%
1	10	8	25	32	0,0050	Costante	33,33%
2	10	8	25	32	0,0050	Crescente	89,99% -> 95,81%
3	10	8	25	32	0,0050	Crescente	92,82% -> 95,63%
4	10	8	25	32	0,0050	Costante	33,33%
5	10	8	25	32	0,0050	Crescente	65,20% -> 95,82%
1	10	8	25	32	0,0010	Costante	33,33%
2	10	8	25	32	0,0010	Crescente	80,27% -> 95,62%
3	10	8	25	32	0,0010	Crescente	83,93% -> 95,57%
4	10	8	25	32	0,0010	Costante	33,33%
5	10	8	25	32	0,0010	Crescente	61,99% -> 95,69%

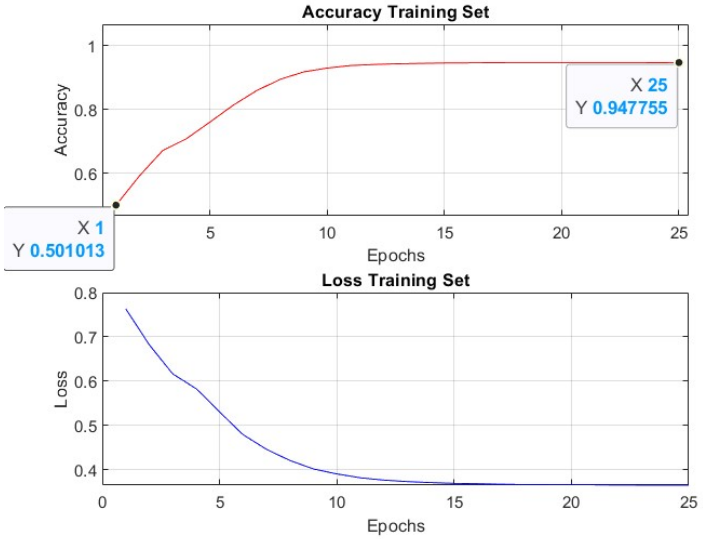
Nei test della tabella riportata sopra si è usata la NORMALIZZAZIONE MAX-MIN ed il METODO DEL GRADIENTE STOCASTICO CON α COSTANTE. Si nota in essa che all'aumentare del learning rate α si hanno range di accuratezza più alti.

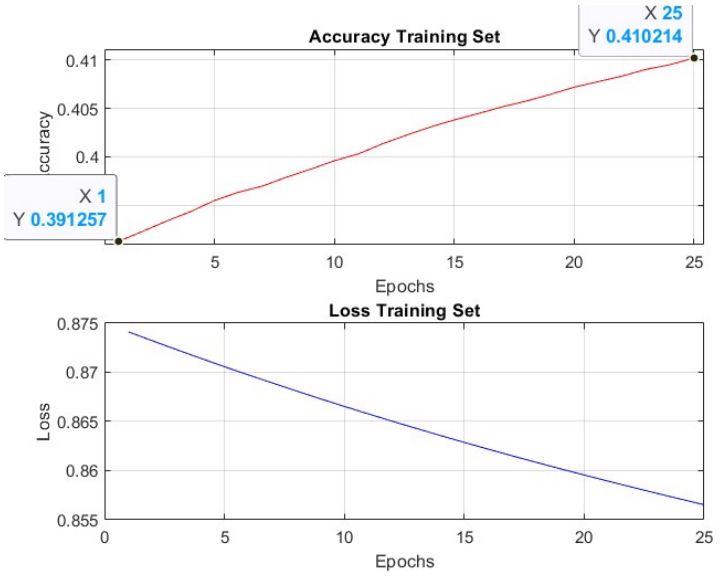
rng	FIRST HIDDEN LAYER	SECOND HIDEN LAYER	EPOCHS	DIM. MINI-BATCH (m)	LEARNING RATE (α)	ANDAMENTO ACCURACY	VALORE ACCURACY (%)
1	10	5	25	32	0,0050	Crescente	61,39% -> 65,60%
2	10	5	25	32	0,0050	Crescente	83,16% -> 95,32%
3	10	5	25	32	0,0050	Crescente	52,17% -> 59,43%
4	10	5	25	32	0,0050	Costante	66,67%
5	10	5	25	32	0,0050	Crescente	40,49% -> 95,74%
1	10	5	25	64	0,0050	Crescente	60,24% -> 65,45%
2	10	5	25	64	0,0050	Crescente	76,39% -> 95,32%
3	10	5	25	64	0,0050	Crescente	49,99% -> 58,15%
4	10	5	25	64	0,0050	Costante	66,67%
5	10	5	25	64	0,0050	Crescente	40,99% -> 95,72%
1	10	5	25	256	0,0050	Crescente	58,06% -> 64,27%
2	10	5	25	256	0,0050	Crescente	66,95% -> 95,47%
3	10	5	25	256	0,0050	Crescente	44,67% -> 57,81%
4	10	5	25	256	0,0050	Costante	66,67%
5	10	5	25	256	0,0050	Crescente	42,04% -> 84,01%

Nei test della tabella riportata sopra si è usata la NORMALIZZAZIONE MAX-MIN ed il METODO DEL GRADIENTE STOCASTICO CON α COSTANTE. Si nota in essa che aumentando la dimensione del batch da considerare ad ogni iterazione, il valore di accuratezza tende a diminuire. Inoltre, confrontando questa tabella con la precedente, si osserva che diminuendo il numero di neuroni nel secondo strato, il comportamento dell'accuratezza che rimane costante ha comunque valore doppio mentre nel primo test si trasforma in un incremento.

rng	FIRST HIDDEN LAYER	SECOND HIDEN LAYER	EPOCHS	DIM. MINI-BATCH (m)	LEARNING RATE (α)	ANDAMENTO ACCURACY	VALORE ACCURACY (%)
1	10	10	50	16	0,0050	Crescente	36,32% -> 84,94%
2	10	10	50	16	0,0050	Crescente	35,45% -> 64,72%
3	10	10	50	16	0,0050	Crescente	80,26% -> 94,15%
4	10	10	50	16	0,0050	Costante	32,24%
5	10	10	50	16	0,0050	Crescente	65,72% -> 94,29%
1	10	6	25	16	0,0050	Crescente	76,91% -> 95,83%
2	10	6	25	16	0,0050	Crescente	51,73% -> 91,72%
3	10	6	25	16	0,0050	Crescente	83,81% -> 94,94%
4	10	6	25	16	0,0050	Crescente	65,12 -> 95,33%
5	10	6	25	16	0,0050	Crescente	87,67% -> 94,19%
1	10	4	75	16	0,0050	Crescente	87,49% -> 95,77%
2	10	4	75	16	0,0050	Crescente	81,37% -> 95,34%
3	10	4	75	16	0,0050	Crescente (fino all'epoca 39 poi va in overfitting)	94,65% -> 95,75%
4	10	4	75	16	0,0050	Crescente con sovraelongazione iniziale	76,72%-> 82,24%
5	10	4	75	16	0,0050	Decresce e poi cresce	46,03% -> 44,12%

Nei test della tabella riportata sopra si è usata la STANDARDIZZAZIONE ed il METODO DEL GRADIENTE STOCASTICO CON α COSTANTE. Si può osservare che variando il numero di neuroni si ottengono dei comportamenti diversi tra loro, infatti, se si hanno pochi neuroni (4) la rete può bloccarsi o può decrescere l'accuratezza. Usandone troppi (10) invece la rete neurale restituisce valori di accuratezza non sempre ottimali o potrebbe trovare un punto di minimo/flesso e rimanere costante.





I due andamenti di accuratezza e di perdita riportati mostrano come usando un batch composto da soli pochi campioni (16) si ottengono dei valori migliori ed in minor tempo (GRAFICO DI SINISTRA).

Mentre usando un metodo batch, e quindi tutti i campioni del training set insieme, si hanno dei valori peggiori e che tendono al valore ideale ma hanno bisogno di più epoche per farlo (GRAFICO DI DESTRA).

Tutti i test finali eseguiti sulla rete con i valori ottimali scelti sono presenti nella cartella "Grafici delle prove con parametri finali"; quelli che seguono sono i primi cinque test riportati in tabella.

rng	FIRST HIDDEN LAYER	SECOND HIDEN LAYER	EPOCHS	DIM. MINI-BATCH (m)	LEARNING RATE (α)	ANDAMENTO ACCURACY	VALORE ACCURACY (%)
1	10	6	25	16	0,0050	Crescente	76,91% -> 95,83%
2	10	6	25	16	0,0050	Crescente	51,73% -> 91,72%
3	10	6	25	16	0,0050	Crescente	83,81% -> 94,94%
4	10	6	25	16	0,0050	Crescente	65,12 -> 95,33%
5	10	6	25	16	0,0050	Crescente	87,67% -> 94,19%