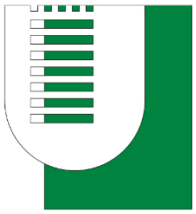

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

UNIVERSITÀ DEGLI STUDI DI ROMA "TOR VERGATA"

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

PROGETTO PER METODI DI OTTIMIZZAZIONE PER BIG DATA

"IMPLEMENTAZIONE DI UNA RETE NEURALE IN MATLAB"

PROFESSORE

Prof. Andrea Cristofari

Prof. Andrea Pacifici

STUDENTI

Auriemma Valerio

Gerardi Emanuele

Regine Mario

ANNO ACCADEMICO 2022/2023

INDICE

1. PROGETTO

1.1. OBIETTIVO

Lo scopo del progetto è stato quello di implementare un modello di machine learning fra quelli visti durante il corso, ovvero tra i seguenti modelli:

- Minimi quadrati
- Regressione logistica
- Rete neurale

Si è deciso di implementare una Rete Neurale sia per comprenderne a pieno il funzionamento sia per la sua diffusione più che attuale. A livello di linguaggio di programmazione si è scelto di utilizzare Matlab, poiché noto a tutti i membri del gruppo.

1.2. ISTRUZIONI PER L'ESECUZIONE

Per poter eseguire il codice, è necessario avere installato Matlab sul proprio pc con installato il Toolbox "Statistics and Machine Learning Toolbox". Nella cartella del progetto ci sono due file principali e tutte le funzioni ausiliarie utilizzate nel progetto. Il file "ReteNeurale.m" è il main del programma in cui vengono definiti tutti i parametri necessari al funzionamento della rete, ovvero il caricamento del dataset, la scelta degli iperparametri. Per semplicità di esecuzione, è possibile caricare direttamente i workspace calcolati in precedenza, così da ottenere immediatamente i risultati.

1.3. ARCHITETTURA DELLA RETE NEURALE

Come rete neurale, si è deciso di realizzare un'architettura il più generica possibile così da poter affrontare diversi tipi di problemi con un unico programma. La rete offre numerosi parametri di configurazione così da poter effettuare test in condizioni sempre diverse così da poter trovare la miglior configurazione per un dato problema. Di seguito vengono anche riportate le variabili, presenti all'interno del codice, che permettono all'utente di modificare gli iperparametri.

1.3.1. SCELTA DEL NUMERO DI LAYER E NUMERO DI NEURONI

Una rete neurale è composta da un layer di ingresso, un layer di uscita e un numero variabile di layer nascosti. Quest'ultimo valore può essere impostato dall'utente andando a modificare il valore della variabile "hiddenLayers", così da settare il numero di layer nascosti che si desiderano. Ogni layer nascosto può avere un numero di neuroni diverso; quindi, bisogna andare ad inserire il numero desiderato nel vettore "layersSz", in cui basta inserire il numero di neuroni desiderato secondo la struttura della rete neurale che si vuole ottenere. Ad esempio, se si vuole avere una rete a 2 strati nascosti con, rispettivamente, 32 e 16 neuroni basta modificare:

```
hiddenLayers = 2;

layerSz = [dimInput  32  16  dimOutput];
```

dove dimInput e dimOutput sono le dimensioni dei campioni del Training Set.

1.3.2. Algoritmi

In questa rete neurale viene utilizzato lo Stochastic Gradient Descent (SGD).

1.3.3. Inizializzazione dei pesi

A livello strutturale, la matrice dei pesi è stata implementata attraverso un tensore. L'inizializzazione dei pesi viene affidata alla funzione "he_init", che permette di inizializzare i pesi della rete in maniera randomica secondo una distribuzione Gaussiana i cui parametri sono dipendenti dalla dimensione dell'input. Per quanto riguarda il bias, inizialmente viene settato a zero. All'interno del codice i pesi e i bias sono contenuti nella stessa matrice.

1.3.4. Regolarizzazione

La rete neurale implementata permette di scegliere fra due tipi di regolarizzazione:

- ι_1 - regularization
- ι_2 – regularization

È possibile utilizzare una o l'altra andando a modificare il valore della variabile "reg". Settando il valore ad 1, si attiverà la regolarizzazione L1, mentre, settando il valore della regolarizzazione a 2, si attiverà la regolarizzazione L2.

1.3.5. Learning Rate

Il **Learning Rate αk** è scelto costante e nell'ordine dei centesimi o inferiore. È possibile modificarlo andando a modificare la variabile "step". Questa scelta permette di avere un learning rate sufficientemente piccolo così da 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 della matrice dei pesi (quindi pesi e bias) a ±∞ (cosa che succede solitamente con α nell'ordine dei decimi).

1.3.6. Funzione attivazione

Le funzioni di attivazione implementante sono la funzione sigmoidea e la ReLu. È possibile cambiare tale funzione anche se, con diverse prove, si è visto che i risultati migliori si ottengono con la ReLu. Quindi la scelta la può comunque effettuare l'utente in base al problema che deve risolvere.

1.3.7. Cross Validation

All'interno del codice è stata utilizzata una procedura di Cross Validation. Quindi, ogni Training Set utilizzato viene suddiviso in 3:

- Training Set;
- Validation Set;
- Test Set;

Con le seguenti proporzioni:

- 70% dei dati per il Training Set;
- 15% caduno dei dati per Validation Set e Test Set;

Ruolo importante nella fase di Cross Validation, viene svolta dall'iperparametro λ. Nel codice è possibile andare a modificare i valori di tale parametro operando sulla variabile "lambda". È un vettore, quindi, permette l'uso di molteplici λ. Dopo la verifica dell'accuratezza sul Validation Set, viene scelto il parametro λ che offre performance migliori.

1.3.8. Minibatch

Per ottenere performance migliori si è deciso di utilizzare un minibatch che permette di prendere un sottoinsieme di campioni del Training Set così da migliorare le predizioni e quindi l'accuratezza della rete. La dimensione di tale batch è selezionabile dall'utente andando a modificare il valore della variabile "mini". Nella successiva fase di test, la dimensione del minibatch è stata scelta secondo la seguente regola:

mini = 1/32 del Training Set

2. Test della rete neurale

Per effettuare i test della rete, sono stati presi online diversi dataset così da poter effettuare test per problemi di classificazione sia nel caso multiclasse sia nel caso binario. L'accuratezza è calcolata come media dei punti classificati correttamente sul totale. Viene riportata solamente l'accuracy calcolata nel test set.

2.1. Dataset utilizzati e Pre-processamento

Ogni Dataset, prima di esser utilizzato, subisce una fase di pre-processamento che consiste nell'andare ad eseguire una normalizzazione e uno shuffle dei dati. Questo processo viene eseguito solo per i dati del Training Set.

2.1.1. Classificazione

La rete da noi realizzata è in grado di fare previsioni su dataset di classificazione sia binaria sia multiclasse. Ovviamente, nel caso della multiclasse, i tempi di esecuzione si allungano passando da qualche decimo a qualche minuto.

2.1.1.1. Binaria

Per la classificazione binaria, sono stati testati due differenti dataset:

- Tumors
- Acoustic extinguisher fire

Il primo dataset è composto da 563 campioni e 31 features.
Il secondo dataset è composto da 17 442 campioni e 6 features.
Di seguito, vengono riportati, in forma tabellare, i risultati ottenuti con la rete neurale implementata per entrambi i dataset visti in precedenza. Sono state effettuate prove con parametri differenti. Sono state notate quelle più significative.

Tumors						
Hidden Layers	Number of units	Learning Rate	Regularization	Lambda	Epoch	Accuracy
1	4	0.0001	ι_2	0.01	5	92,8%
1	10	0.0001	ι_2	0.01	5	76,1%
2	[16, 8]	0.001	ι_2	0.001	5	83,3%
3	[16, 8]	0.0001	ι_1	0.02	10	88,1%

Acoustic extinguisher fire						
Hidden Layers	Number of units	Learning Rate	Regularization	Lambda	Epoch	Accuracy
1	3	0.00001	ι_2	0	8	85,6%
1	15	0.00001	ι_2	0	10	85,2%
1	16	0.0001	ι_1	0.01	10	85,5%

2.1.1.2. Multiclasse

Per la classificazione multiclasse, è stato scelto come dataset di test il ben noto MNIST, cioè un dataset molto conosciuto nell'ambito del Machine Learning. Esso è composto da circa 42 000 campioni, 785 features e 10 classi. Di seguito è possibile vedere il risultato ottenuto con la rete.

MNIST						
Hidden Layers	Number of units	Learning Rate	Regularization	Lambda	Epoch	Accuracy
2	[128, 256]	0.0001	ι_2	0	5	92,3%