TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Relazione progetto

Metodi di Ottimizzazione per Big Data

A.A. 2024/2025

Cacace Elisa 0365355

Callegari Luca

Jin Valentina 0365400

Indice

1 Introduzione

Il progetto prevede la realizzazione di un modello machine learning classico: una **rete neurale** di tipo feedforward. Si è scelto, inoltre, di risolvere un problema di **classificazione binaria**.

Tale modello è stato implementato interamente in linguaggio Python e non è stato previsto l'utilizzo di librerie già implementate da terzi per l'addestramento della rete. Si è però deciso di usufruire, e quindi di scaricare, di librerie per la realizzazione di grafici, per l'esecuzione di calcoli matematici e per il preprocessing dei dati.

In particolare sono state utilizzate le seguenti librerie:

- numpy
- matplotlib
- pandas
- scikit-learn
- seaborn

Il codice sorgente lo si può scaricare da questo link.

2 Dataset

Per valutare le prestazioni del modello sono stati scelti tre dataset:

- ***Glioma Grading Clinical and Mutation Features***
- ***Darwin***
- ***Predict Students' Dropout and Academic Success***

2.1 Glioma Grading Clinical and Mutation Features

I gliomi sono i tumori primari più comuni del cervello. Possono essere classificati come LGG (glioma di basso grado) o GBM (glioblastoma multiforme) in base a criteri istologici e/o di imaging. Anche i fattori clinici e molecolari/mutazionali sono molto importanti per il processo di classificazione. I test molecolari sono costosi, ma aiutano a diagnosticare con precisione i pazienti affetti da glioma.

In questo dataset, sono stati considerati i 20 geni con mutazioni più frequenti e 3 caratteristiche cliniche, provenienti dai progetti TCGA-LGG e TCGA-GBM sui gliomi cerebrali.

Il compito di previsione consiste nel determinare se un paziente ha un LGG o un GBM, a partire dalle sue caratteristiche cliniche e molecolari/mutazionali. L'obiettivo principale è individuare il sottoinsieme ottimale di geni mutati e caratteristiche cliniche per il processo di classificazione del glioma, al fine di migliorare le prestazioni e ridurre i costi.

→ Campioni: 839

→ Features: 23

→ Label: Grade (0 = "LGG"; 1 = "GBM")

2.2 Darwin

Il dataset DARWIN è stato appositamente concepito per offrire alla comunità scientifica un'opportunità avanzata di perfezionare le attuali metodologie di apprendimento automatico, finalizzate alla previsione della malattia di Alzheimer attraverso l'analisi della scrittura manuale.

→ Campioni: 174

→ Features: 451

→ Label: class (0 = "P"; 1 = "H")

2.3 Predict Students' Dropout and Academic Success

Il dataset è stato creato nell'ambito di un progetto che mira a contribuire alla riduzione dell'abbandono e dell'insuccesso accademico nell'istruzione superiore, attraverso l'utilizzo di tecniche di apprendimento automatico per identificare precocemente gli studenti a rischio, in modo da poter attuare strategie di supporto mirate.

Il dataset include informazioni disponibili al momento dell'immatricolazione dello studente – percorso accademico, dati demografici e fattori socio-economici.

Il problema è formulato come un compito di classificazione in tre categorie (Dropout, Enrolled, e Graduate) al termine della durata normale del corso.

Per adattarlo ad un problema di classificazione binaria si è deciso di unire le classi Dropout e Enrolled in un'unica classe.

→ Campioni: 4424

→ Features: 36

→ Label: Categorical (0 = "Dropout/Enrolled"; 1 = "Graduate")

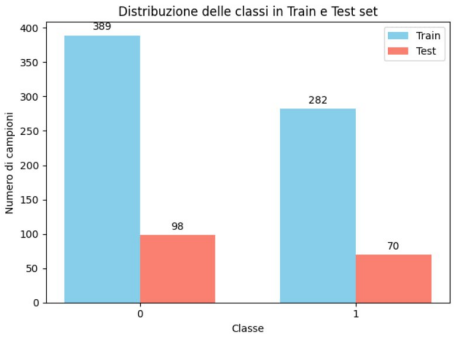
3 Preprocessing dei dati

Una volta caricato il dataset, il quale viene sottoposto ad una fase di preprocessamento dei dati, *pulendoli* per poter addestrare correttamente il modello.

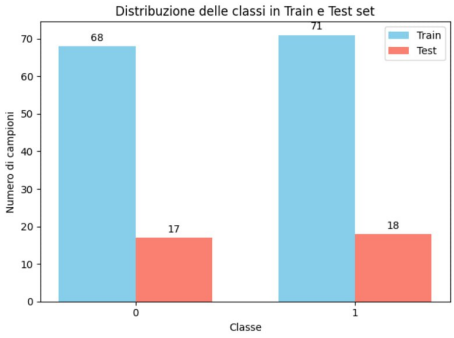
Le tecniche di preprocessamento adottate sono state le seguenti:

1. **Eliminazione label incoerenti**: questa modifica è stata apportata solo per il dataset *Darwin* togliendo la colonna ID in quanto non rilevante per l'addestramento del modello
2. **Codifica label testuali**
3. **Eliminazione righe incomplete**: sono state rimosse righe che presentavano valori mancanti
4. **Shuffle delle righe**: per ottenere istanze di entrambe le classi sia sul training set sia sul test set, dal momento che i dataset presentavano un ordinamento basato sui label è stato necessario riordinare randomicamente le righe (i campioni). In questo modo si migliora la distribuzione dei dati, migliorando quindi anche le performance del modello
5. **Separazione features e label**
6. **Standardizzazione feature numeriche**: è stata eseguita per rendere confrontabili le feature che si trovano su scale diverse, così da evitare che quelle con valori numericamente più alti possano influenzare in modo sproporzionato l'addestramento del modello

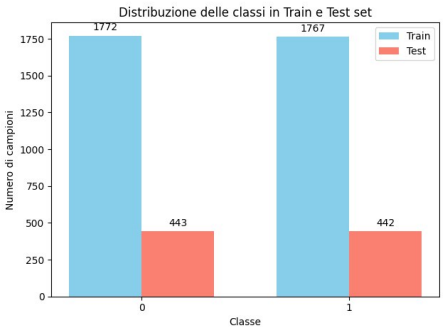
7. **Divisione dataset**: la divisione in training set e test set è stata eseguita con **stratificazione** così da mantenere la stessa proporzione delle classi del dataset originale



Dataset Glioma



Dataset Darwin



Dataset Academic Dropout

4 Rete Neurale

Si è deciso di implementare una rete neurale con un numero di strati nascosti variabile, così come il numero di neuroni per ogni strato nascosto. In particolare il numero di strati nascosti è definito nel file `config.py` e può essere facilmente cambiato prima dell'esecuzione del programma. Di default è stato impostato a:

```
# NUMERO DI LIVELLI NASCOSTI
L = 3
```

Per scegliere il modello migliore per ogni dataset è stata usata nell'addestramento la **k-fold cross validation** per trovare il miglior numero di neuroni per ciascun livello nascosto e il relativo λ per la regolarizzazione. A questo scopo è stato definito nel file `config.py` una lista di neuroni possibili:

```
HIDDEN_LAYER_NEURONS_LIST = [10, 12, 16, 32, 64]
```

Analogamente, sempre nel file `config.py`, si è definita una lista di possibili scelte del **fattore di regolarizzazione**:

```
LAMBDA_VALUES_LIST = [0,1e-4,1e-3,1e-2,0.1]
```

4.1 Inizializzazione dei pesi e funzioni di attivazione

Le funzioni di attivazione sono state definite nel file `activationFunctions.py` e all'avvio del programma si lascia la scelta all'utente tra le funzioni **ReLu** e **Tanh**. Inoltre in base alla scelta della funzione di attivazione viene impostata anche la migliore inizializzazione dei pesi corrispondente.

In particolare le due inizializzazioni implementate sono state: ***He Initialization*** e ***Xavier Initialization***, rispettivamente per ReLu e Tanh.

4.2 Addestramento

Nella fase di addestramento del modello si è utilizzato il **metodo del gradiente stocastico con momentum**, decidendo di adottare un **learning rate dinamico**. Nelle prime iterazioni dell'algoritmo si usa un passo α costante ma poi, per avere una convergenza stocastica, si cambia usando un *diminishing stepsize*.

All'inizio, viene calcolato il numero totale di mini-batch necessari in base alla dimensione del dataset `m` e alla dimensione scelta per ogni batch (`MINI_BATCH_SIZE`). Per ogni **epoca** si rimescolano casualmente gli indici dei dati per evitare che il modello impari dipendenze dall'ordine.

Poi, in ogni iterazione del ciclo interno, si estraggono gli indici del batch corrente (`batch_idx`) e si selezionano i corrispondenti esempi (`X_batch`) e le etichette (`Y_batch`) da usare per aggiornare il modello.

Una volta fissati `X_batch` e `Y_batch` si procede alla fase del calcolo delle uscite ϕ tramite la procedura di **feedforward** per poi calcolare il gradiente della loss function usando la procedura di **backpropagation**.

Dal momento che per l'addestramento si è utilizzata la k-fold cross validation per trovare il modello migliore, inevitabilmente sono state provate tutte le possibili combinazioni di parametri e quindi anche alcune non molto buone. Per questo motivo, a volte, durante la cross validation alcuni modelli restituiscono precision pari a 0 e il programma genera un warning di divisione per 0.

5 Valutazione del modello

Le metriche di valutazione utilizzate sono:

- **Accuracy**: indica la percentuale di previsioni corrette sul totale delle previsioni.
- **Precision**: misura la capacità del modello di trovare tutti i campioni positivi.
- **Recall**: misura la capacità del modello di classificare correttamente i veri positivi rispetto a tutti i campioni classificati come positivi.
- **F1 score**: è una metrica che combina precision e recall in un'unica misura, utilizzando la loro media armonica. È utile quando si cerca un equilibrio tra precision e recall. Un F1 score alto è ottenibile solo se entrambe le metriche sono alte.

Queste metriche sono indicative sulla performance del modello quando lo testiamo su nuovi campioni: i campioni appartenenti al test set.

6 Risultati e prestazioni

Per valutare le prestazioni dei nostri modelli si considera, oltre la *Accuracy*, *Precision*, *Recall* e *F1 score*, anche:

- un grafico che misura l'andamento dell'accuracy durante il training
- un grafico che misura l'andamento della loss function durante il training
- matrice di confusione

Questi risultati sono ottenuti effettuando preliminarmente la procedura di **cross validation** per ogni combinazione di funzioni di attivazione e tipi di regolarizzazione, prendendo come *best model* il modello che ha avuto le migliori prestazioni.

Per maggiore chiarezza si definisce il *best model* come: $([a, b, c, d, e], \lambda)$ dove $[a, b, c, d, e]$ è il vettore dei neuroni per ogni strato (compresi input e output), mentre λ è il fattore di regolarizzazione ottimo.

6.1 Glioma Grading Clinical and Mutation Features

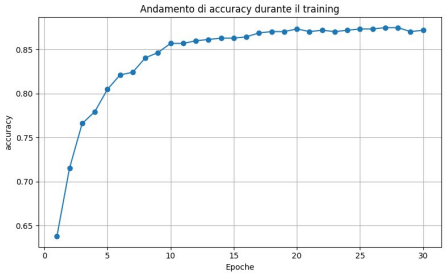
Tramite la **cross validation** è stato determinato come *best model*:

$$([23, 12, 32, 64, 2], 0.01)$$

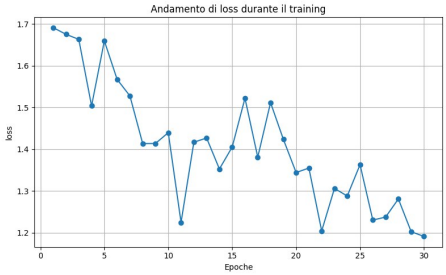
usando la Tanh con regolarizzazione L2.

Si sono, inoltre, ottenute le seguenti prestazioni:

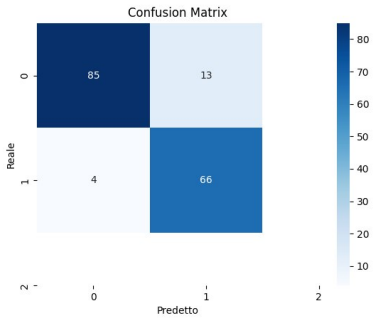
	Accuracy	Precision	Recall	$F1-\text{score}$
Classe 0	0.90	0.96	0.87	0.91
Classe 1		0.84	0.94	0.89



Accuratezza durante l'addestramento



Loss durante l'addestramento



Matrice di confusione

6.2 Darwin

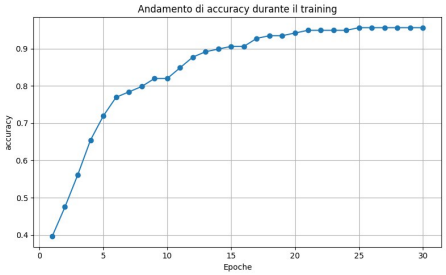
Tramite la **cross validation** è stato determinato come *best model*:

$$([450, 12, 16, 64, 2], 0.1)$$

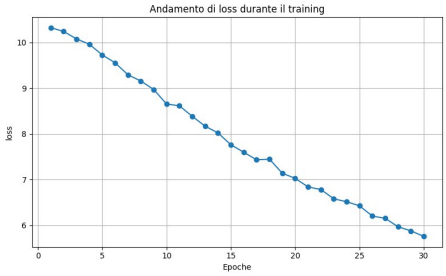
usando la Tanh con regolarizzazione L2.

Si sono, inoltre, ottenute le seguenti prestazioni:

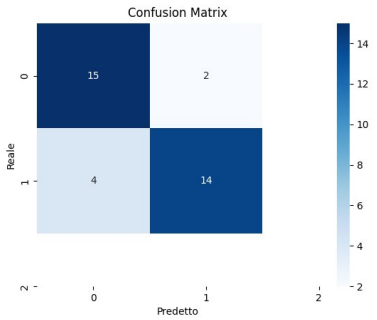
	Accuracy	Precision	Recall	$F1-\text{score}$
Classe 0	0.83	0.79	0.88	0.83
Classe 1		0.88	0.78	0.82



Accuratezza durante l'addestramento



Loss durante l'addestramento



Matrice di confusione

6.3 Predict Students' Dropout and Academic Success

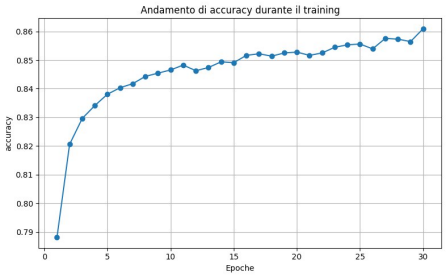
Tramite la **cross validation** è stato determinato come *best model*:

$$([36, 10, 16, 32, 2], 0.0001)$$

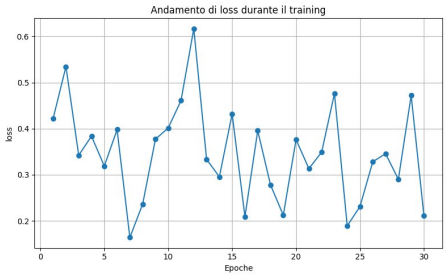
usando la Tanh con regolarizzazione L1.

Si sono, inoltre, ottenute le seguenti prestazioni:

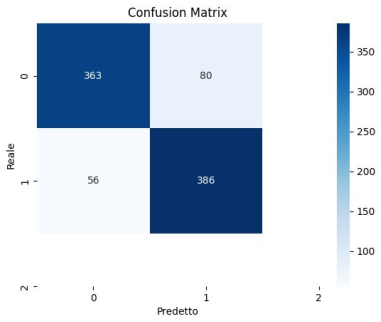
	Accuracy	Precision	Recall	$F1-\text{score}$
Classe 0	0.85	0.87	0.82	0.84
Classe 1		0.83	0.87	0.85



Accuratezza durante l'addestramento



Loss durante l'addestramento



Matrice di confusione

7 Conclusioni

Dai risultati ottenuti si può affermare che la funzione di attivazione Tanh è risultata essere la funzione di attivazione migliore per tutti e tre i dataset scelti.

Si può osservare che per il secondo dataset (Darwin) abbiamo un'accuratezza di circa 97% durante il training. Questo è un chiaro segno di possibile overfitting, infatti le prestazioni sul test set non sono state altrettanto soddisfacenti. Tale risultato era atteso data la scarsità di istanze presenti in quello specifico dataset.

Infine, per tutti i dataset, **l'andamento della loss** durante l'addestramento è chiaramente decrescente (globalmente), anche se a volte presenta delle oscillazioni che si verificano perché ad ogni passo l'aggiornamento dei pesi è basato su un sottoinsieme diverso del dataset, che può avere una distribuzione leggermente diversa rispetto al totale. Questo porta a variazioni nei gradienti calcolati, e quindi a fluttuazioni nei valori della loss.

8 Istruzioni d'uso

Per poter avviare il programma bisogna preliminarmente scaricare l'interprete Python versione 3.10.4, o successive. Successivamente aprire il prompt riga di comando e in ordine digitare i seguenti comandi.

- Uso MAC OS
    1. `python Start.py`
    2. `python Main.py`
- Uso Windows
    1. `python Start.py`
    2. `python Main.py`

Nel caso in cui si utilizzi Windows come sistema operativo principale e Ubuntu venga eseguito su una macchina virtuale, è necessario digitare `python3` anziché `python` per avviare l'interprete Python all'interno dell'ambiente Ubuntu.

Una volta avviato il programma `Main.py` selezionare per prima cosa il dataset, successivamente la funzione di attivazione desiderata, ed infine il tipo di regolarizzazione.

9 Bibliografia

I dataset utilizzati per questo progetto sono scaricabili qui:

- Glioma
- Darwin
- Academic Dropout

E per i riferimenti teorici sono state utilizzate le slide del corso: `Modelli e Algoritmi di Machine Learning A.A. 2024/2025`