TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Relazione progetto
Metodi Di Ottimizzazione Per Big Data
A.A. 2023/2024

Lupini Martina 0344256

Indice

1 Introduzione

Il progetto prevede l'implementazione di un modello di machine learning. Si è scelto di implementare una **rete neurale** per un problema di **classificazione binaria**.

Il codice può essere trovato a questo link.

Il linguaggio di programmazione utilizzato è Python e per poter eseguire il modello è necessario scaricare le seguenti librerie:

- numpy
- matplotlib
- sklearn
- pandas

2 Dataset

Sono stati scelti tre dataset per testare le performance del modello:

- **Dataset Alzheimer**: questo dataset contiene informazioni sanitarie dettagliate per 2.149 pazienti. Il dataset include dettagli demografici, fattori legati allo stile di vita, anamnesi medica, misurazioni cliniche, valutazioni cognitive e funzionali, sintomi e diagnosi della malattia di Alzheimer.
  - **numero di campioni**: 2.149
  - **numero di features**: 33
- **Dataset Mushrooms**: questo dataset contiene 8 caratteristiche principali dei funghi (come ad esempio stagione, altezza del gambo, colore del cappello, ...) insieme ad un'indicazione se il fungo è velenoso o commestibile.
  - **numero di campioni**: 54.035
  - **numero di features**: 8
- **Dataset Fraud detection**: questo dataset include oltre 550.000 transazioni con carte di credito effettuate da titolari europei nel 2023. I dati sono stati anonimizzati per garantire la privacy dei titolari. Per ogni transazione c'è un'indicazione se è fraudolenta o meno.
  - **numero di campioni**: 568.631
  - **numero di features**: 29

3 Preprocessing dei dati

Prima di passare all'addestramento della rete neurale si svolgono una serie di operazioni sul dataset volte a migliorare la qualità dei dati. Infatti, dati più puliti comportano una maggiore accuratezza e affidabilità nei risultati. Nel corso della

trattazione indicheremo come *istanze positive* quelle che hanno il label pari a 1 e *istanze negative* quelle che lo hanno pari a 0. Le operazioni preliminari svolte sono:

1. **Rimozione valori nulli**: vengono rimosse tutte le istanze che presentano valori nulli.

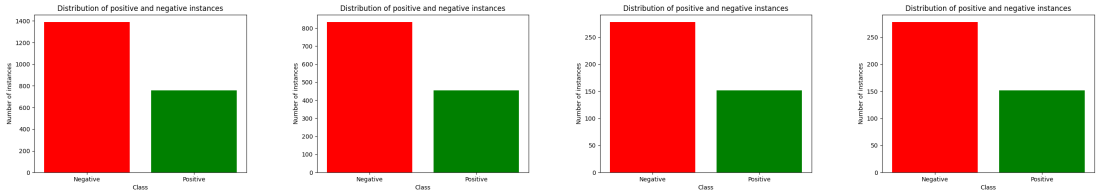
2. **Standardizzazione dataset**: si manipolano i dati in modo che le variabili abbiano una media pari a 0 e una deviazione standard pari a 1. Serve a ridurre la scala dei dati in modo che siano comparabili tra loro e facciano performare la rete neurale in modo migliore.

3. **Riduzione della dimensione dataset**: questa fase viene effettuata solo sui dataset *Mushrooms* e *Fraud detection*. Vista l'assenza di hardware specializzato e l'esecuzione della rete neurale su comuni personal computer, verrà considerata solo una sottoparte delle istanze degli ultimi due dataset (facendo attenzione che la percentuale di istanze positive e negative rimanga la stessa del dataset completo). In particolare, si considerano 6.000 istanze per *Mushrooms* e 20.000 per *Fraud detection*

4. **Divisione in training, validation e testing set**: la proporzione utilizzata è **60%** per training set, **20%** per testing set e **20%** per validation set. Nel suddividere il dataset originale si utilizza la **stratificazione**. Lo scopo di questa tecnica è garantire che la distribuzione delle classi nei diversi set rimanga proporzionata rispetto a quella presente nel dataset originale. Se i dati venissero divisi casualmente, si potrebbero ottenere set di dati in cui una classe è sovra o sottorappresentata rispetto alla distribuzione originale. Questo potrebbe portare il modello a imparare meno efficacemente o in modo errato, poiché i set di training, validation e testing non sarebbero rappresentativi della realtà.

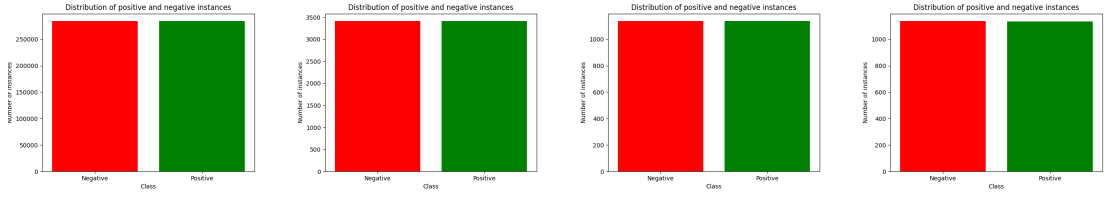
Distribuzioni delle istanze positive (verde) e negative (rosso) nei vari dataset:

Dataset Alzheimer



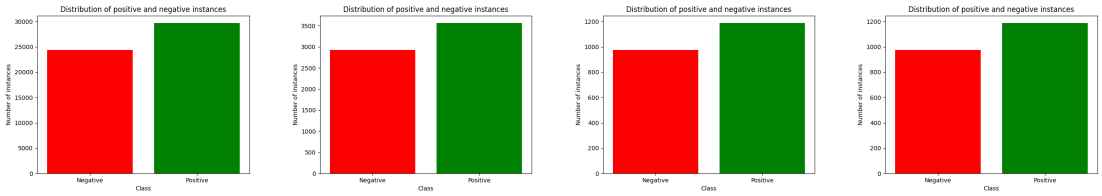
Dataset Completo | Training set | Validation set | Test set

Dataset Fraud detection



Dataset Completo | Training set | Validation set | Test set

Dataset Mushrooms



Dataset Completo | Training set | Validation set | Test set

Come possiamo vedere dai grafici la stratificazione garantisce che la proporzione di esempi appartenenti a ciascuna classe sia mantenuta nel campione di training, validation e testing.

4 Rete Neurale

Si è scelto di implementare una rete con **due strati nascosti**. La scelta di due strati nascosti rappresenta un buon compromesso tra capacità di apprendere relazioni non lineari e complessità computazionale.

4.1 Funzioni di attivazione

Come funzione di attivazione è possibile scegliere tra **ReLu** (Rectified Linear Unit) e **tanh** (tangente iperbolica). Lo strato di output ha sempre come funzione di attivazione **sigmoid** in quanto risulta essere la più appropriata per problemi di classificazione [1].

4.2 Inizializzazione pesi

Per quanto riguarda l'inizializzazione dei pesi è stata scelta **He initializaion** [2] in quanto ha delle ottime performance soprattutto se si utilizza **ReLu** come funzione di attivazione [3].

4.3 Addestramento

Nella fase di addestramento si utilizza il **metodo del gradiente stocastico con momentum** con stepsize costante di 0.1. La variante con momentun aiuta a migliorare la convergenza e accelerare l'addestramento.
Il minibatch selezionato ha una dimensione pari a 64, poiché l'SGD è noto per funzionare bene con batch di dimensioni ridotte [4]. Tuttavia, 64 è stato considerato un valore non troppo piccolo, in modo da evitare l'introduzione di instabilità eccessiva.
La scelta del learning rate è stata guidata dalle raccomandazioni presentate in [5], in cui si evidenzia che, diminuendo il learning rate, si riduce l'errore di generalizzazione, ma al contempo aumenta significativamente il numero di epoche necessarie per raggiungere la massima accuratezza. Per trovare un compromesso tra tempo e accuratezza, seguendo anche quanto riportato in [4], dove si suggerisce che l'SGD

funziona bene con learning rate non troppo bassi, è stato scelto un valore di 0.1. Come funzione di perdita si utilizza la **binary cross-entropy**.

4.4 Cross-validation

La tecnica della cross validation è stata utilizzata per decidere la configurazione ottimale della rete neurale. In particolare è stata eseguita facendo variare:

- **La struttura della rete neurale**: sono state prese in esame le seguenti configurazioni (consideriamo solo il numero di neuroni degli strati nascosti):
  - (32, 32)
  - (64, 32)
  - (64, 64)
  - (128, 64)
  - (128, 128)
  - (256, 128)
  - (256, 256)
- **Tipo di regolarizzazione**: per ciascuna delle configurazioni precedenti si è eseguito il training con regolarizzazione L1, con regolarizzazione L2 e senza regolarizzazione.
- **Parametro di regolarizzazione** λ: sono stati considerati i seguenti parametri:
  - regolarizzazione L1: 0.001, 0.005, 0.5, 1
  - regolarizzazione L2: 0.01, 0.1, 0.5, 4.5

4.5 Valutazione del modello

Le metriche considerate per valutare il modello finale sono:

- **Accuracy**: indica la percentuale di previsioni corrette sul totale delle previsioni.
- **Recall**: misura la capacità del modello di trovare tutti i campioni positivi.
- **Precision**: 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.

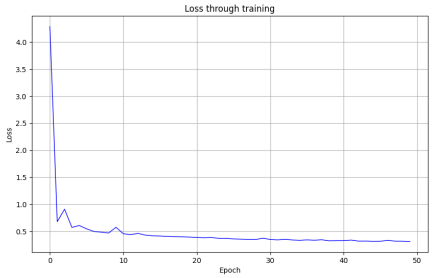
5 Risultati

In questa sezione verranno analizzati i risultati ottenuti sui diversi dataset al variare della funzione di attivazione.

5.1 Dataset Alzheimer (2.000 istanze)

5.1.1 ReLu

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(32, 32)* utilizzando *regolarizzazione L1* con $\lambda = 0.5$.
Tempo impiegato per la cross-validation: 1:16 min.



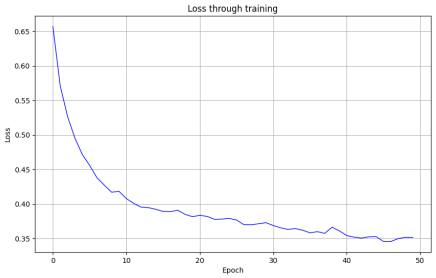
Loss durante l'addestramento

Metrica	Valore
Accuracy	86.51%
Precision	81.33%
Recall	80.26%
F1 Score	0.8079

Risultati del modello usando il testing set

5.1.2 Tanh

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(32, 32)* utilizzando *regolarizzazione L1* con $\lambda = 0.5$.
Tempo impiegato per la cross-validation: 1:28 min.



Loss durante l'addestramento

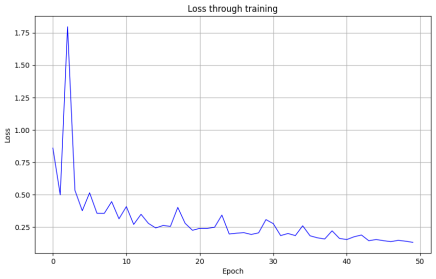
Metrica	Valore
Accuracy	84.88%
Precision	83.21%
Recall	71.71%
F1 Score	0.7703

Risultati del modello usando il testing set

5.2 Dataset Mushrooms (6.000 istanze)

5.2.1 ReLu

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(256, 128)* utilizzando *regolarizzazione L1* con $\lambda = 0.001$.
Tempo impiegato per la cross-validation: 7:38 min.



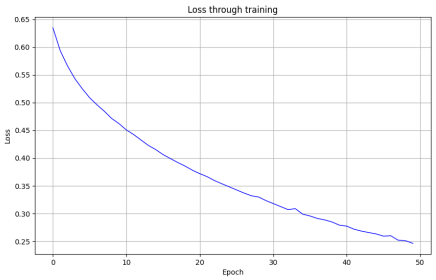
Loss durante l'addestramento

Metrica	Valore
Accuracy	96.30%
Precision	97.27%
Recall	95.96%
F1 Score	0.9660

Risultati del modello usando il testing set

5.2.2 Tanh

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(256, 128) senza regolarizzazione.*
Tempo impiegato per la cross-validation: 9:11 min.



Loss durante l'addestramento

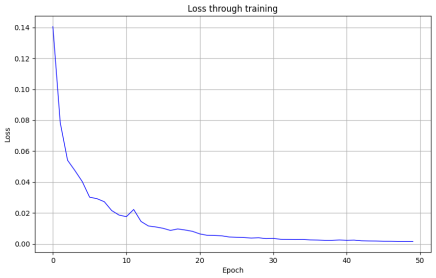
Metrica	Valore
Accuracy	96.67%
Precision	97.13%
Recall	96.80%
F1 Score	0.9696

Risultati del modello usando il testing set

5.3 Dataset Fraud detection (20.000 istanze)

5.3.1 ReLu

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(256, 128) senza regolarizzazione.*
Tempo impiegato per la cross-validation: 9:09 min.



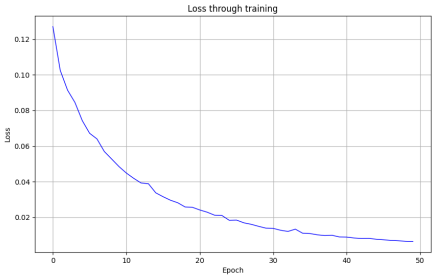
Loss durante l'addestramento

Metrica	Valore
Accuracy	98.78%
Precision	98.43%
Recall	99.12%
F1 Score	0.9877

Risultati del modello usando il testing set

5.3.2 Tanh

Eseguendo la cross-validation si ottiene che la configurazione ottimale è *(128, 128)* utilizzando *regolarizzazione L1* con $\lambda = 0.001$.
Tempo impiegato per la cross-validation: 8:43 min.



Loss durante l'addestramento

Metrica	Valore
Accuracy	98.73%
Precision	98.85%
Recall	98.59%
F1 Score	0.9872

Risultati del modello usando il testing set

5.4 Osservazioni finali

La rete neurale sul dataset *Alzheimer* mostra prestazioni inferiori rispetto agli altri due dataset. Questo risultato potrebbe essere attribuito al numero ridotto di istanze del dataset, nonché allo squilibrio tra istanze positive e negative.
Nel dataset *Alzheimer* la funzione di attivazione ReLu presenta delle performance migliori rispetto a Tanh. Questo avviene anche per il dataset *Fraud detection* seppur la differenza sia minima. Nel dataset *Mushrooms* è Tanh a performare meglio.
La regolarizzazione L1 risulta essere quella più utilizzata, soprattutto nel dataset *Mushrooms*, che è quello con più features.

È importante sottolineare che, in diverse esecuzioni del programma, sia il modello ottimale che le sue performance potrebbero variare. Ciò è dovuto al fatto che, ad ogni esecuzione, i dataset vengono suddivisi in maniera differente nei set di training, validation e testing.

Riferimenti bibliografici

[1] Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. Towards Data Sci, 6(12), 310-316.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV), 1026-1034.

[3] Datta, L. (2020). A survey on activation functions and their relation with xavier and he normal initialization. arXiv preprint arXiv:2004.06632.

[4] Xing, C., Arpit, D., Tsirigotis, C., & Bengio, Y. (2018). A walk with sgd. arXiv preprint arXiv:1802.08770.

[5] Wilson, D. R., & Martinez, T. R. (2001, July). The need for small learning rates on large problems. In IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222) (Vol. 1, pp. 115-119). IEEE.