Results Report

Homework 3

Stefano Cortellessa – 254260

Per lo sviluppo di questo Homework sono stati considerati diversi dataset su cui sono stati effettuate le operazioni di seguito riportate (per eseguire i diversi dataset basta decommentare la variabile 'breast' desiderata).

- Breast Cancer (9 *Attributes,* 286 *Istances*),
- Hayes Roth (5 *Attributes, 160 Istances*),
- Car Evaluation (6 *Attributes, 1728 Istances*)

Le principali librerie utilizzate per l'implementazione sono *pandas*, *numpy & sklearn.*
I risultati ottenuti dal testing sono riportati nei notebook che sono in allegato a questo file (Homework3.ipynb) nella cella 'In[40]' perché meglio visibili e organizzabili.

I file *.html* contenenti i risultati ottenuti sono così organizzati:

- ***Breast_A*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 2
- ***Breast_B*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2
- ***Breast_C*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 3
- ***Breast_D*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 3
- ***Breast_?*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2, inseriti anche nel trainingSet

- ***Hayes_A*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 2
- ***Hayes_B*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2
- ***Hayes_C*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 3
- ***Hayes_D*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 3
- ***Hayes _?*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2, inseriti anche nel trainingSet

- ***tic-tac-toe_A*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 2
- ***tic-tac-toe_B*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2
- ***tic-tac-toe_C*** = rete neurale con 1 hidden layer da 100 nodi e massimo numero di '?' 3
- ***tic-tac-toe_D*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 3
- ***tic-tac-toe _?*** = rete neurale con 2 hidden layer da 100 nodi e massimo numero di '?' 2, inseriti anche nel trainingSet

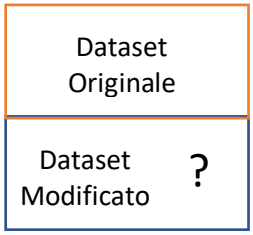
Lo scopo di questo homework è stato quello di creare e allenare una o più reti (nel mio caso una rete per ogni combinazione di valori mancanti) a predire determinati output (multi-label classification) su un dataset che presenta valori mancanti, identificati dal carattere *'?'.*

Inizialmente è stato aperto il dataset attraverso la funzione *openDataset()* e creata una copia per mantenere le istanze reali. Successivamente, *randomicamente*, sono stati inseriti all'interno del primo vari caratteri *'?',* identificanti il valore mancante, di un numero massimo pari alla variabile *upbound* che può essere facilmente modificata (il valore 'standard' da me assegnato è 3 in quanto,

per il metodo di splitting descritto successivament, la separazione tra Training Set e Test Set risulterà rispettivamente di circa 80% / 20 %).
Il passo successivo è stato quello di codificare il dataset risultante e la copia dell'originale attraverso la tecnica *One-Hot Encoding* (con l'utilizzo della funzione *get_dummies()* messa a disposizione da *pandas)*. I due dataset sono poi stati 'appesi' l'uno all'altro in modo tale da avere

- Dataset originale senza '?',
- Dataset modificato con '?'

In un unico dataset grande il doppio come visibile dalla figura seguente.



Per fare in modo da riuscire ad eseguire un training che sia il più accurato possibile, sono stati creati i due set di training e di testing.
Per scelta, questi ultimi sono stati così suddivisi:

- Per ogni sample del dataset, gli stessi sono stati divisi in due (*xTrain & yTrain*)
- Controllando poi sul dataset totale (originale e modificato insieme) le entry che hanno '?' nella variabile target:
    - Nel Training Set sono messe tutte quelle entry che NON hanno il valore '?',
    - Nel Test Set, invece, sono messe quelle entry che presentano '?' nella variabile target.

Questa scelta è stata effettuata in quanto altrimenti, facendo training su dati/valori in cui sono presenti caratteri uguali a '?', il quale per noi rappresenta un valore mancante, avrebbe portato alla predizione di tale carattere, che non rappresenta lo scopo.

Prima di passare alla fase di allenamento e testing, è stata sviluppata l'architettura che crea una rete neurale per ogni possibile combinazione di valori mancanti all'interno del dataset, predicendo in output tali valori attraverso una classificazione multi-lable e multi-categorical.
Quindi la rete viene allenata passandogli il trainingSet e testata successivamente classificando quelle che sono le label mancanti per quella specifica rete.

Quindi è stata implementata la funzione *probPredict*, la quale è responsabile di:

- Splitting dell'input/output *(preprocessing() method),*
- Splitting di Test Set e Training Set *(splitTrainAndTest() method),*
- Allenamento delle reti neurali *(NeuralNets() method)*,
- Classificazione *(trainAndPredictProba() method)*,
- Calcolo del vettore probabilistico rappresentante la probabilità che ogni colonna contenga il valore 1 in quella 'cella' del dataset (*trainAndPredict() method*),
- Percentuale di splitting di TrainingSet e TestSet,
- Zero/One Loss,
- Log Loss,
- Percentuale '?' nell'intero dataset.

Esempio vettore probabilistico*:*

```
['Class_?', 'Class_no-recurrence-events', 'Class_recurrence-events']
[2.40147157e-05,          9.99956119e-01,          3.33944523e-05]

La probabilità che quel valore sia uguale a:
-   '?' = 2.40147157e-05
-   'no-recurrence-events' = 9.99956119e-01
-   'recurrence-events' = 3.33944523e-05
```

Per scelta, come già indicato precedentemente, i risultati non sono stati salvati in file separati in quanto questi sono stati fatti separatamente per ogni combinazione possibile di *missing values*, qui ndi su dataset molto grandi sarebbero stati creati migliaia di file separati e quindi di difficile lettura (esiste nel codice un metodo *writing()*, non utilizzato, che esegue la scrittura dei risultati sui file csv).

Dalle diverse esecuzioni effettuate sui diversi dataset precedentemente indicati, ho tratto le seguenti conclusioni:

Ciò che è stato notato, in quanto aspettato a priori, e verificato dall'esecuzione della mia architettura è che:

- Il valore di accuracy per questa tipologia di dataset (categorici) è più bassa rispetto a dataset con valori reali.
- La classificazione multi-lable è meno efficace di una classificazione single lable. Questo può essere notato facilmente comparando i valori di 0/1 Loss quando si stanno predicendo 2 classi e quando se ne sta predicendo solamente una.
- Reti con più strati sono più accurate (valore di accuracy più alto) rispetto a quelli con meno strati e con meno neuroni in esse.

Comparando i diversi dataset invece:

Per il dataset Breast Cancer, inserendo un numero massimo di '?' (su una riga) pari a 2 con una percentuale totale sull'intero dataset pari al 8% si nota che una perdita 0/1 per la classificazione di classi singole buona (una media di 0,2) e che aumentando il numero di label predette (classificazione multi-lable) la perdita rimane sugli stessi valori. Inoltre, provando ad aumentare il numero di livelli della rete/numero di nodi si nota che l'architettura inaspettatamente dà in output una perdita molto simile alla classificazione single lable precedentemente descritta, rimanendo su una media di 0.2.
Inserendo invece un numero massimo di '?' (su una riga) pari a 3 con una percentuale totale sull'intero dataset pari al 18% si nota che anche in questo caso la perdita 0/1 già per la classificazione di classi singole è alta (una media di 0,4), ma che aumentando il numero di label predette (classificazione multi-lable) la perdita aumenta ad una media di 0,6.
Inserendo invece nel Training Set anche i sample che presentano missing values si nota che, come aspettato, la perdita 0/1 aumenta, anche se di poco raggiungendo un valore medio di 0,3 per classificazione a single label e 0,2 per multi lable.

Per il dataset Hayes Roth, inserendo un numero massimo di '?' (su una riga) pari a 2 con una percentuale totale sull'intero dataset pari al 14% si nota che la perdita 0/1 la classificazione di classi singole è buona (una media di 0,1), ma che aumentando il numero di label predette (classificazione multi-lable) la perdita aumenta fino a 0,4.

Provando ad aumentare il numero di livelli della rete/numero di nodi si nota che l'output dell'architettura migliora di poco.
Inserendo invece un numero massimo di '?' (su una riga) pari a 3 con una percentuale totale sull'intero dataset pari al 28% si nota anche in questo caso la perdita 0/1 già per la classificazione di classi singole è alta (una media di 0,2) e che aumentando il numero di label predette (classificazione multi-lable) la perdita aumenta ad un valore molto vicino a 0,5.
Inserendo invece nel Training Set anche i sample che presentano missing values in questo caso si nota che la perdita 0/1 rimane su un valori medi di 0,2 per classificazione a single label e aumenta fino a 0,6 per multi lable.

Infine, per il dataset tic-tac-toe può essere osservato un comportamento molto simile a quello dei precedenti dataset, con valori di output simili.
Inserendo un numero massimo di '?' (su una riga) pari a 2 con una percentuale totale sull'intero dataset pari al 14% si nota che la perdita 0/1 già per la classificazione di classi singole è molto alta (una media di 0,1), ma che aumentando il numero di label predette (classificazione multi-lable) la perdita aumenta fino quasi ad 0,3.
Anche in questo caso, provando ad aumentare il numero di livelli della rete/numero di nodi si nota che l'output dell'architettura migliora di poco.
Inserendo invece un numero massimo di '?' (su una riga) pari a 3 con una percentuale totale sull'intero dataset pari al 30% si nota anche in questo caso la perdita 0/1 già per la classificazione di classi singole è alta (una media di 0,2) e che aumentando il numero di label predette (classificazione multi-lable) la perdita aumenta ad un valore molto vicino a 0,4.
Inserendo invece nel Training Set anche i sample che presentano missing values si nota che, come aspettato, la perdita 0/1 aumenta, anche se di poco raggiungendo un valore medio di 0,3 per classificazione a single label e 0,4 per multi lable.

Il dataset 'migliore' in termine di prestazioni generali tra quelli testati è quindi *'Breast Cancer'.*
Una considerazione fondamentale da fare è che avendo il dataset modificato (quello contenente i '?') concatenato al dataset originale e creando il testSet inserendo al suo interno solo quei sample che presentano il valore '?' nella variabile di target, c'è la possibilità che venga creato un testSet con una sola istanza. Ciò sta a significare che in quest'ultimo caso, se la rete sbaglia la predizione, la perdita sarà pari a 1, aumentando così la media descritta precedentemente.
Pertanto, i valori espressi in base al parametro 0/1 loss è indicativa e dipendente sia dalla grandezza del testSet che, ovviamente, dal numero di '?' inseriti randomicamente. Infatti, come visibile dalle esecuzioni eseguite su training set contenenti missing values, orientativamente, i valori di perdita aumentano, quindi hanno un'accuratezza più bassa.
Quello che posso concludere dalla mia analisi è che questo tipo di problema, per come è stato pensato, è abbastanza efficiente, ma per dataset con molte istanze, in quanto più è grande il dataset più è alto il numero di '?' presenti in esso, il livello di accuracy tende a peggiorare.