Assignment I

1920-2-F1801Q151, Advance Machine Learning, 2019-2020

Università degli studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione

Villa Giacomo `807462`

21 Ottobre 2019

Indice

1 Dataset e Operazioni preliminari

Il **dataset** a nostra disposizione contiene informazioni riguardanti i pagamenti, fattori demografici, dati sul credito, cronologia dei pagamenti ed estratti conto dei clienti di carte di credito a Taiwan da Aprile 2005 a Settembre 2005.

Sono stati effettuati degli **studi riguardanti la correlazione** (*Pearson*) tra gli attributi e il target, in seguito agli stessi ho evidenziato una **bassa correlazione di alcuni** di questi; ho deciso di non procedere all'eliminazione di quelli con correlazione bassa (valori inferiori allo `0.0x`) in quanto la rete, durante la fase di apprendimento provvederà ad attribuire un peso minore ai collegamenti di quei attributi non utili alla classificazione. L'unica ragione per cui avrebbe avuto senso eliminarli sarebbe stato per ragioni di tempi di calcolo i quali in ogni caso non subiscono rallentamenti. **La matrice di correlazione è presente nel notebook**.

Sono state **tentate** anche altre strategie quale la **combinazione degli attributi** (creazione di `amountPay` e `amountBill`) e la **binarizzazione di alcuni attributi**. **Non sono state osservate particolari migliorie** e pertanto non sono state incluse, sono comunque presenti commentate per eventuali supervisioni.

Tutti i dati risultano essere **normalizzati** mediante la funzione `StandardScaler()` della libreria `Keras`.

Dato interessante riguarda la situazione sul **bilanciamento del dataset**, è possibile notare come i record etichettati con la classe `1` rappresentino il 22% dei record totali:

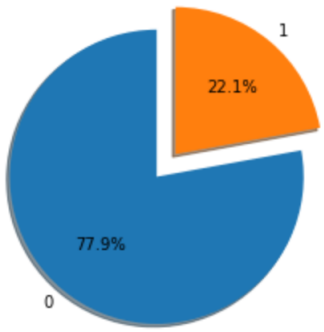

Figura 1: Divisione record in funzione della classe

Risulta fin da qui possibile affermare che **inferire correttamente sulla classe** `1` **potrebbe essere problematico**.

2 Struttura Rete

Per **implementare la rete neurale** è stata utilizzata la libreria `Keras`, il metodo utilizzato è `Sequential` in modo tale da poter collegare sequenzialmente i vari `Dense` (fully connected) layers. Il numero degli **hidden layers** è pari `2` con, rispettivamente **64 neuroni e 32 neuroni**. In ingresso ho un numero di neuroni pari alle features del problema e in **output un solo neurone** (siamo in un contesto binario). In generale:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_505 (Dense)            (None, 64)                1472
_________________________________________________________________
dropout_304 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_506 (Dense)            (None, 32)                2080
_________________________________________________________________
dropout_305 (Dropout)        (None, 32)                0
_________________________________________________________________
dense_507 (Dense)            (None, 1)                 33
_________________________________________________________________
activation_152 (Activation)  (None, 1)                 0
=================================================================
Total params: 3,585
Trainable params: 3,585
Non-trainable params: 0
```

Figura 2: Model Summary

La **scelta dei livelli e dei rispettivi neuroni** è giustificata dal fatto che siamo dotati di un **buon numero di record** (il dataset ne conta 27000) e dunque sarà **possibile apprendere** i `3585` parametri che la rete richiede. In generale ho **provato diverse configurazioni** e ho potuto notare come le performance (che verranno trattate nella sezione `4 Performance` di questa documentazione) non subiscono in generale un grosso miglioramento.

Utilizzo inoltre la **tecnica del** `Dropout` per disattivare determinati neuroni durante la fase di training al fine di **evitare l'overfitting** (i pesi non vengono aggiornati per una certa percentuale di neuroni dato ogni layer).

3 Training della Rete

La **funzione di attivazione** per gli **hidden layer** è ricaduta sulla `ReLu` la scelta è stata guidata dallo stato dell'arte e dalla documentazione presso varie fonti le quali la identificavano come migliore funzione di attivazione per gli hidden layers. Di fatto permette di non avere tutti i neuroni attivi contemporaneamente rendendo di fatto la rete più efficiente.

Per la **funzione di output** ho deciso di utilizzare una `Sigmoid` in quando questa mappa i valori in un range compreso tra `0` ed `1` (di fatto le classi da classificare). In fase di effettiva predizione si procederà ad un assegnamento in funzione del superamento della soglia di `0.5` del valore in output (se maggiore assegno `1` altrimenti `0`).

Per la **dimensione dei batch** ho impostato questa a `256` in quanto, dato il grande numero di record, penso possa essere una quantità sufficiente per permettere un buon movimento del gradiente. Ho **effettuato vari test** con diversi valori, ho potuto osservare come i risultati non tendano a cambiare molto (questo penso sia dovuto al dataset); reputo comunque più corretto utilizzare il valore impostato.

Il **numero di epoche** utilizzato è pari a 20, ho potuto osservare come dopo la ventesima epoca la loss e le metriche utilizzate non vedevano significativi miglioramenti.

Data la natura sbilanciata del dataset ho deciso di utilizzare la funzione `compute_class_weight` di `sklearn` al fine di avvantaggiare la classe di minoranza.

In ultimo, per la **funzione di loss**, ho deciso di minimizzare la `binary_crossentropy` sostanzialmente una cross entropy (quindi una quantificazione della differenza tra due distribuzioni di probabilità) applicata a una realtà binaria dato il problema.

4 Performance

Ai fini di **analisi di performance** del modello ho deciso di utilizzare la **10 CV**; il numero di record mi permette di adottare questo approccio, fornendomi inoltre stime sicuramente più solide per quanto riguarda le performance stesse. Ottengo, per tanto, le seguenti performance sui 10 fold:

Performance			
	Precision	**Recall**	**F1-score**
Label 0	*0.877*	*0.81*	*0.842*
Label 1	*0.473*	*0.601*	*0.529*

Fornisco inoltre un **andamento grafico** delle misure di performance quali **Precision e Recall** sui 10 fold date le **due classi**:

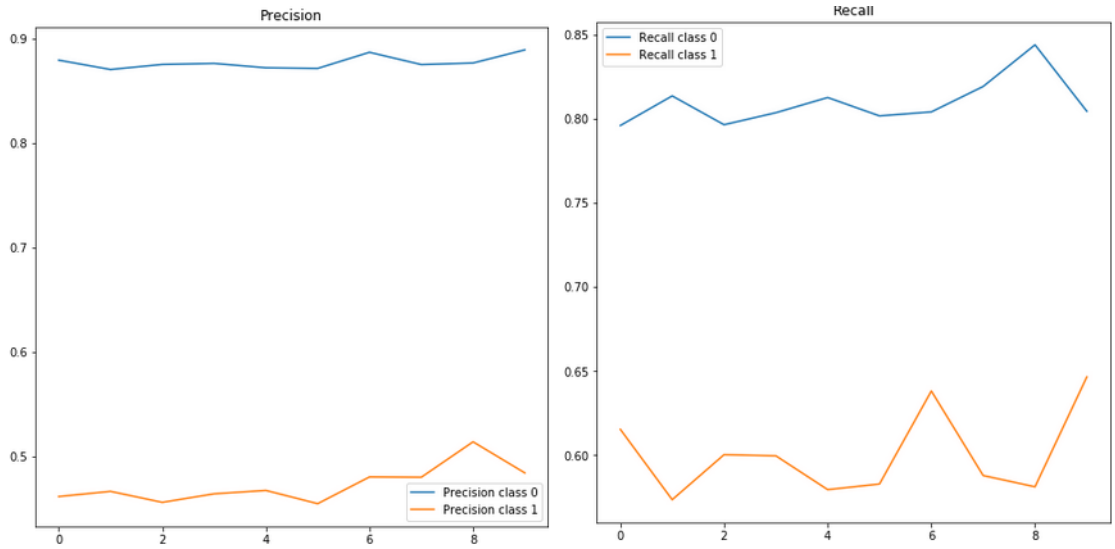

Figura 3: Valori Precision e Recall sui 10 fold

La **classe 1** risulta essere la **più problematica**. Il valore della **Precision** indica una tendenza a etichettare record della classe 0 con la classe 1, il valore della **Recall** invece indica che comunque buona parte delle istanze della classe 1 viene classificata correttamente. Per quanto riguarda invece i valori di performance legati alla **classe 0** questi risultano essere **nel complesso accettabili**.

Fornisco inoltre l'**andamento grafico** per quanto riguarda l'**accuracy** e la **funzione di loss**; vengono forniti i grafici per le fold numero `1`, `5` e `10` in quanto

l'andamento risulta essere molto simile in tutte le fold.

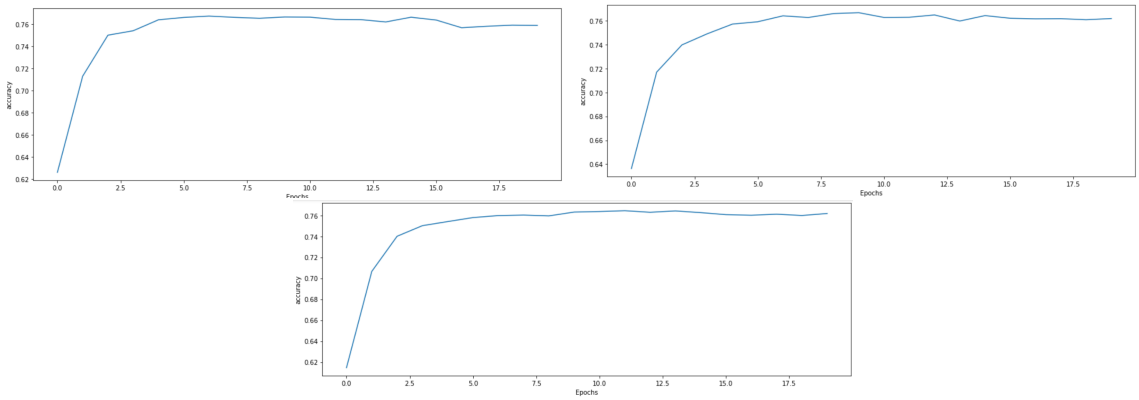

Figura 4: Andamento Accuracy fold `1`, `5` e `10`

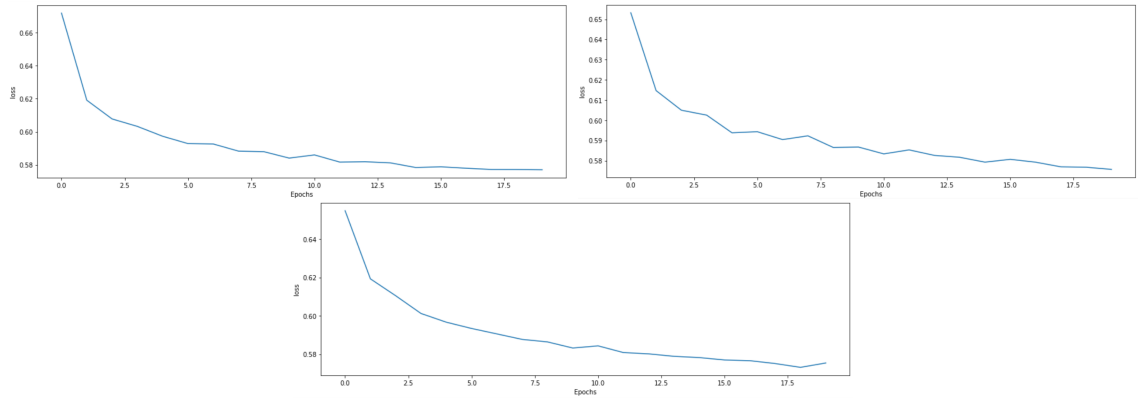

Figura 5: Andamento Loss Function fold `1`, `5` e `10`

Come è possibile notare, in modo particolare per ciò che riguarda i grafici della loss function, **il numero di epoche** è giustificato dalla decrescita della loss appunto, che si stabilizza intorno alla 18°/19° epoca (l'indice parte da 0, l'epoca 0 è pertanto da intendersi come epoca 1).

5 Conclusioni

In generale mi reputo **relativamente soddisfatto** delle performance ottenute, sarebbe molto interessante effettuare **ulteriori attività di pre processing** sul dataset

al fine di tentare un aumento di performance. In particolare sarebbe interessante effettuare un **undersampling** al fine di ottenere più bilanciamento e permettere al modello, si spera, di apprendere meglio.