Report dell'Assignment 1 di Advanced Machine Learning

Paolo Mariani 800307

Ottobre 2019

1 Introduzione

L'assignment verte sulla creazione di una rete neurale per la previsione di inadempimenti di pagamenti bancari da parte dei clienti di una banca Taiwanese, mediante la classificazione dei clienti che sono soliti pagare o meno nel mese successivo il debito contratto.
La costruzione del modello ha inizio con una analisi del dataset "Default of Credit Card Clients Dataset", contenente molteplici informazioni utili per determinare se il cliente salderà o meno il debito.
L'applicazione reale del modello è legata al riconoscimento di casi di inadempimento prima che si verifichino, per permettere ad esempio di attivare meccanismi di prevenzione sulla contrazione di debiti o di etichettamento dei clienti come "inadempienti".

In questo report verranno descrite le procedure e le considerazioni che sono state effettuate per realizzare il modello, oltre ad una breve analisi critica nei confronti delle performance che vengono raggiunte da esso: si inizia con una breve overview dei dati per passare alla successiva fase di preprocessing di questi, seguendo poi con la descrizione del modello implementato e raggiungendo per ultime le conclusioni.

2 Dati e Preprocessing

Il Dataset "Default of Credit Card Clients Dataset", ottenuto tramite il file *train.csv*, contiene dati relativi a 27000 clienti di una banca di Taiwan per il periodo lungo sei mesi tra Aprile e Settembre 2005; le colonne del forniscono le seguenti informazioni:

- **LIMIT_BAL**: importo del credito concesso in dollari Taiwanesi
- **SEX**: genere (1=maschile, 2=femminile)
- **EDUCATION**: titolo di studio-educazione del cliente (1=diploma, 2=laurea, 3=high school, 4=altri, 5=sconosciuto, 6=sconosciuto)
- **MARRIAGE**: stato civile (matrimonio) (1=sposato, 2=single, 3=altro)
- **AGE**: età
- **PAY_0 (PAY_2-3-4-5-6)**: stato del rimborso del debito a Settembre 2005 (per PAY_0, fino a PAY_6 per Aprile), con indicazione del ritardo del pagamento da 1 a 9 mesi (e oltre), un pagamento nei tempi previsti è indicato con -1
- **BILL_AMT1 (BILL_AMT2-3-4-5-6)**: importo dell'estratto conto mensile da Settembre 2005
- **PAY_AMT1 (PAY_AMT2-3-4-5-6)**: importo del pagamento del mese precedente rispetto al mese considerato
- **default.payment.next.month**: variabile target che indica se il pagamento sarà ritardato o no (1=sì, 0=no)

2.1 Preprocessing

Per garantire al modello una buona attività di apprendimento dei dati è stato ritenuto opportuno

procedere con l'analisi delle variabili del dataset col fine di utilizzare solo le informazioni utili per classificare correttamente la variabile target.
Una overview dei valori assunti dalle variabili e delle relazioni esistenti tra esse è stata effettuata tramite la libreria python **pandas-profiling**, la quale permette di creare un report dettagliato del dataset con un semplice comando.
Il risultato ottenuto ha permesso di effettuare una serie di considerazioni utili, tra le quali risultano fondamentali per l'intero preprocessing:

- La matrice della correlazione di Pearson utilizzato per valutare la correlazione tra le variabili evidenzia una forte correlazione tra le variabili **BILL_AMT1-2-3-4-5-6**: è stato deciso di rappresentare le informazioni fornite con un'unica variabile ed evitare di fornire al modello una dati correlati che possono generare un bias in fase di interpretazione delle osservazioni; inoltre assumono valori concentrati perlopiù entro 200000, è necessario scalare il range di valori che può assumere ogni variabile.
  È stato deciso di ridurre la complessità di calcolo applicando ad ognuna una trasformazione logaritmica e generando una singola variabile **AVG_BILL_AMT** rappresentante una media delle variabili.
- Anche alle variabili **PAY_AMT** è stata applicata la trasformazione logaritmica, senza però generare una variabile unica dalla loro media come nel caso precedente.
- Gli istogrammi della variabile **EDUCATION** evidenzia che solo pochi valori di quelli assumibili sono molto frequenti, i rimanenti sono sporadici nel dataset ed indicano situazioni sconosciute (unknown) o rare: è stato pensato di raggruppare i valori più rari in un unica categoria (valore 0) ed ora assume valori 1 per diploma, 2 per laurea, 3 per high school e 4 per altri; Successivamente è stata binarizzata in 4 diverse variabili "**0**", "**1**","**2**","**3**".
- La variabile **MARRIAGE** può assumere 4 valori diversi, è stato deciso di applicare una binarizzazione alla variabile e creare 4 nuove variabili "**SPOSATO0**", "**SPOSATO1**", "**SPOSATO2**", "**SPOSATO3**".
- La variabile "**AGE** assume molteplici valori nel dataset, rappresentando un range di età comprese tra 21 e 79: per ridurre la complessità della variabile è stata considerato un raggruppamento dei valori in diversi bin (["0-20", "21-30", "31-40", "41-50", "51-70", "71+"]) ed una successiva binarizzazione della variabile rappresentante il binning che ha generato 6 nuove variabili intitolate come gli intervalli imposti.
- La variabile "**LIMIT_BAL** è una variabile categorica che può assumere 80 valori diversi, per ridurre la numerosità della variabile è stato applicato un raccoglimento dei valori in 10 bin diversi ([0, 80000, 150000, 230000, 310000, 390000, 470000, 550000, 630000, 710000, 1000001]) ed una successiva binarizzazione della variabile rappresentante il binning creando 10 nuove variabili "**LIMITE_i**".

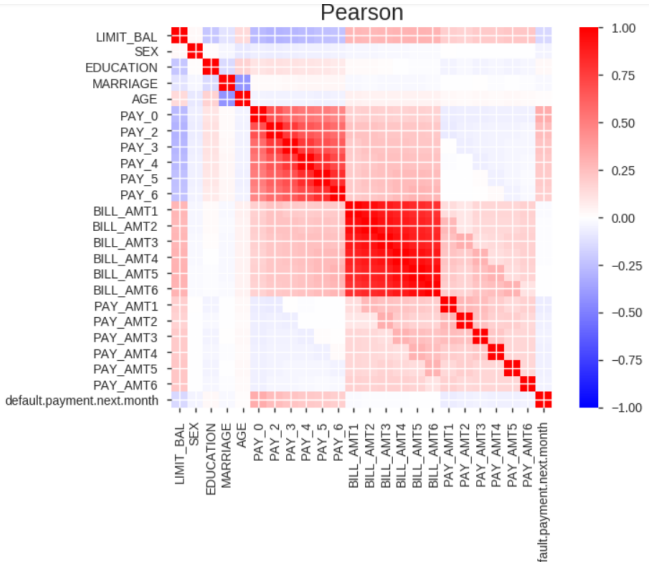

Figura 1: Matrice di Correlazione delle variabili

Tutte le colonne che sono state binarizzate sono state rimosse dal Dataset, il risultato dell'applicazione del Preprocessing ai dati è il seguente:

- **SEX**.
- **PAY_0** fino a **PAY_6**.
- **PAY_AMT1** fino a **PAY_AMT6**.
- ['**0-20**', '**21-30**', '**31-40**', '**41-50**', '**51-70**', '**71+**'] rappresentanti gli intervalli binarizzati della variabile AGE.
- ['**0**', '**1**', '**2**', '**3**'] rappresentanti gli intervalli

binarizzati della variabile EDUCATION.

- **SPOSATO0** fino a sposato **SPOSATO3** generati a partire dalla variabile MARRIAGE.
- **['LIMITE_1', 'LIMITE_2', 'LIMITE_3', 'LIMITE_4', 'LIMITE_5','LIMITE_6', 'LIMITE_7', 'LIMITE_8', 'LIMITE_9', 'LIMITE_10']** partendo dalla variabile LIMIT_BAL.
- **AVG_BILL_AMT** generata dalla media delle singole variabil BILL_AMT trasformate con trasformazione logaritmica.
- **default.payment.next.month**.

2.2 Split dei dati e Downsampling

A partire dal Dataset ottenuto dopo la fase di Preprocessing sui dati è stata effettuata la partizione delle osservazioni tramite la procedura di split dei dati "train_test_split" associando alla variabile *datitrain* l'80% dei dati, e il restante 20% alla variabile *datitest*: si tratta di una ulteriore partizione rispetto a quella già rappresentata dalla divisione dei file *train.csv* e *test.csv* per poter garantire tramite valutazione del modello sulla porzione di dati *datitest* una verifica del corretto apprendimento dei parametri della rete neurale.
A seguito della partizione dei dati è stata analizzata la composizione della variabile **default.payment.next.month**, che in tutto il dataset risulta essere sbilanciata per il 78% per la classe 0 e solo il 22% rimanente per la classe 1.
È stato deciso di implementare un meccanismo di ***Downsampling*** del dataset per decrementare il numero di osservazioni con variabile target etichettata come 0 per pareggiarlo con il numero di osservazioni più rare etichettate come 1, ottenendo una proporzione del 50% ciascuna.
Prima di procedere con l'implementazione del modello della rete neurale, le variabili contenenti i dataframe *datitrain* e *datitest* sono state separate ulteriormente nel formato di input richiesto dalla rete (matrice) per la fase di apprendimento e test della rete, generando le strutture che saranno fornite alla rete neurale in input:

- X_train
- y_train
- X_test
- y_test

3 Implementazione Rete Neurale

La rete neurale presentata è il frutto di una continua ricerca della migliore configurazione dei layer e dei parametri, la quale è stata effettuata tramite la continua elaborazione dei feedback ottenuti da molteplici prove; il risultato ottimale raggiunto è il seguente:

- **Strato Iniziale Dense 1**: 32 neuroni, dimensione dell'input pari al numero di colonne (38), funzione di attivazione "relu"
- **Strato Dropout 1**: il 30% di unità sono spente casualmente in fase di training
- **Strato Dense 2**: 64 neuroni, funzione di attivazione "relu"
- **Strato Dropout 2**: il 30% di unità sono spente casualmente in fase di training
- **Strato Dense 3**: 32 neuroni, funzione di attivazione "relu"
- **Strato Dropout 3**: il 60% di unità sono spente casualmente in fase di training
- **Strato Finale Dense 4**: 1 neurone, funzione di attivazione "sigmoid"

Per poter arrivare alla analisi della rete appena mostrata si effettua una breve descrizione dei modelli implementati precedentemente, in modo da comprendere quali siano le ragioni che hanno portato all'implementazione finale.

3.1 Primo modello

La configurazione iniziale della rete è molto simile a quella finale, con la presenza di *3 layer Dense* per svolgere facilmente un compito abbastanza complesso; differiva però per il numero di neuroni per ogni strato (sempre 32) e per la presenza di *un solo layer*

Dropout situato tra il secondo ed il terzo strato;
L'utilizzo di un layer di ***Dropout*** è giustificato dall'intenzione di "proteggere" la rete da un eventuale caso di ***overfitting***, in cui il modello comincia a specializzarsi sulle osservazioni del training set e non riesce più a fornire buoni risultati per la classificazione sulla porzione di dati dedicata al test set: implementando questo layer è possibile simulare il training della rete neurale sui dati come se fossero disponibili più versioni differenti della stessa rete, in quanto il layer è specializzato nello spegnimento ranodmico di una determinata percentuale di unità (nel modello in analisi il 30%) che favoriscono un apprendimento migliore ed evita di creare una specializzazione della rete.

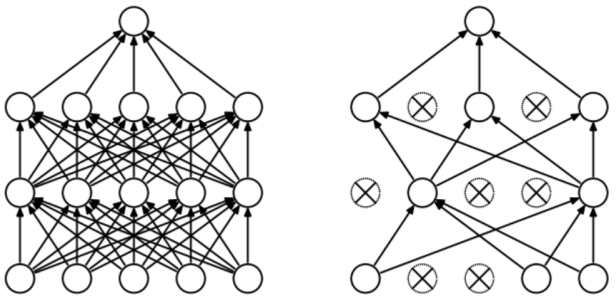

Figura 2: Esempio di struttura della rete con e senza dropout

3.1.1 Funzioni di attivazione

Tutte le funzioni di attivazione che sono state associate a ciascun layer (ad esclusione di quello finale) sono di tipo "***ReLU***", cioè una funzione di attivazione "Rectified Linear Unit" che restituisce, fornito un valore in input, un risultato equivalente se il valore è positivo (cioè identità) o un valore pari a 0 se invece l'input risulta essere negativo; È una delle funzioni di attivazione più utilizzate e consigliate dalla letteratura poichè:

- se i valori di input sono negativi la funzione non si attiva, quindi non tutti i neuroni sono attivati e la rete diventa sparsa; ciò garantisce una buona velocità di computazione del modello.
- il gradiente della funzione è zero per i valori negativi, quindi durante il meccanismo di backpropagation non avviene un aggiornamento di quei pesi associati.

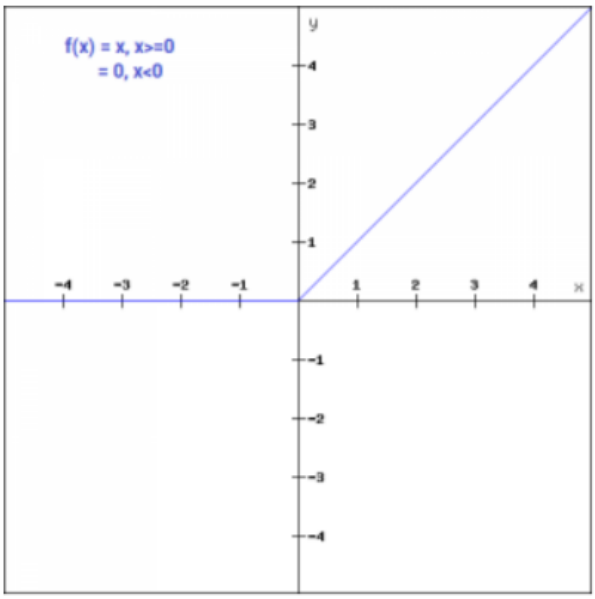

Figura 3: Funzione di attivazione ReLU

Un'altra peculiarità del modello iniziale riguarda la presenza della funzione di attivazione ***Sigmoid*** nell'ultimo layer, una funzione di trasferimento che permette in casi di classificazione di produrre un risultato soddisfacente mappando l'output su di un intervallo di valori [0,1]; è stata scelta in quanto indica una probabilità di quanto una osservazione appartenga alla classe 0 (o 1 alternativamente) come risultato della sua computazione. Inoltre la funzione è differenziabile, ma come contro ha che il suo gradiente diventa molto piccolo nell'intervallo [-3, 3] e la rete può non apprendere a causa di ciò, quindi è stata scelta come tradeoff accettabile a favore della sua capacità di classificare tra due classi.

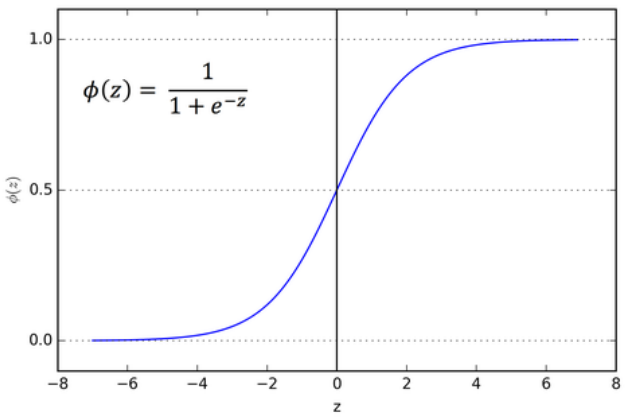

Figura 4: Funzione di attivazione Sigmoid

3.1.2 Optimizer

Per quanto riguarda l'ottimizzatore, è stato scelto ***Adam*** (da Adaptive Moment Estimation), esso rappresenta un ottimizzatore combinazione di RMSProp e di AdaGrad con momentum: permette rispetto al metodo SGD di ridurre i tempi di calcolo per raggiungere l'ottimo, anche se non sempre è migliore rispetto a quello fornito dal più lento SGD. Il vantaggio che si trae dal suo utilizzo è legato alla sua capacità di adattare il "*learning rate*", cioè un iperparametro che permette di controllare, in funzione dell'errore commesso, di quanto modificare i pesi della rete ogni volta che avviene un aggiornamento degli stessi. Inizialmente l'ottimizzatore della rete era RMSProp, successivamente durante la fase di test dei vari modelli è risultato più performante Adam e quindi RMSProp è stato sostituito.

3.1.3 Loss Function

Infine l'ultima componente da considerare nel modello è la Loss Function: la scelta è ricaduta sulla "***Binary Crossentropy***", una funzione di perdita consigliata dalla letteratura per i casi di classificazione binaria e per via del fatto che il suo valore incrementa (per ogni osservazione analizzata) sempre più quando la probabilità osservata è lontano dal valore reale dell'osservazione.

3.2 Modello finale

In ultimo si passa ad analizzare la rete proposta per la risoluzione del problema di classificazione.
La rete possiede la stessa struttura del primo modello (appena descritto sopra), ma possiede alcune modifiche relative al numero di neuroni e di layer Dropout; Il numero di neuroni è stato incrementato da 32 a 64 per il layer centrale (*Dense2*) in quanto ha fornito migliori risultati in termini di accuracy su più esecuzioni del modello rispetto alla versione a 32 unità.

In relazione invece al numero di layer Dropout è importante sottolineare che sono stati aggiunti altri due layer intermedi dopo il layer Dense 1 (con parametro 0.3) e prima dell'ultimo layer Finale Dense 4 (con parametro 0.6): una modifica sostanziale considerando che le performance sono incrementate notevolmente in termini di F-measure, a causa del cambiamento dinamico della struttura della rete in fase di training; l'ultimo layer Dropout possiede una percentuale randomica di spegnimento delle unità parecchio alto, pari a 0.6, ma la scelta è risultata corretta in quanto i risultati sono migliorati diversificando la struttura dei neuroni prima del layer Dense Finale 4.

Le altre componenti, quali ottimizzatore, funzioni di attivazione, Loss function e dimensione del batch (128) sono identiche a quelle descritte nel primo modello.
In fase di training la rete è stata impostata per essere eseguita per 50 epoche valutando con le metriche di *accuracy* e *f-measure* (la cui implementazione può essere visualizzata nel notebook), applicando il meccanismo di split dei dati per effettuare la validazione del modello ad ogni epoca pari al 30% dei dati di training.

4 Risultati e Conclusioni

4.1 Risultati

Durante la fase di training e validation su 50 epoche il modello ha raggiunto:

- un valore di **Loss** pari a 0.57 nel training, 0,58 nella fase di validation. I due valori sono molto vicini, graficamente permettono di osservare che non si verifica una situazione di overfitting.

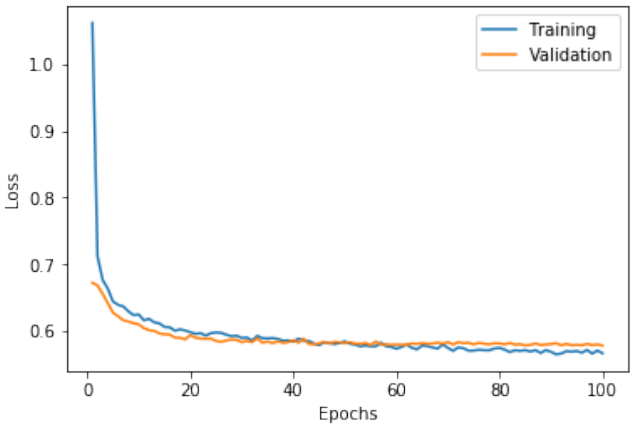

Figura 5: Valori di Loss per Training e Validation su 100 epoche

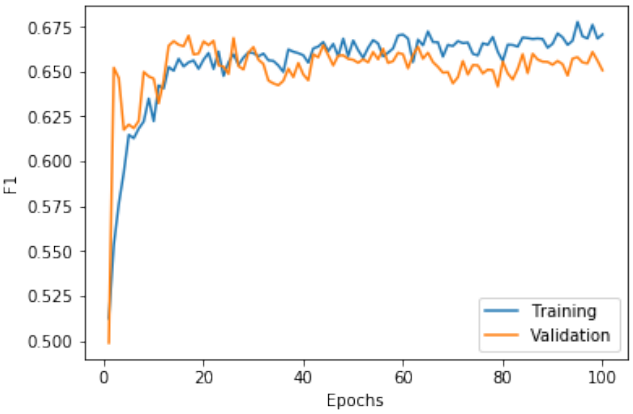

Figura 7: Valori di F-measure per Training e Validation su 100 epoche

- un valore di **Accuracy** pari a 0.70 nel training, 0,69 nella fase di validation. Anche per questa metrica i due valori risultano molto vicini, rimangono costanti nel tempo e non si verifica una condizione di overfitting dove invece solitamente l'accuracy del training diverge raggiungendo valori di molto migliori della validation.

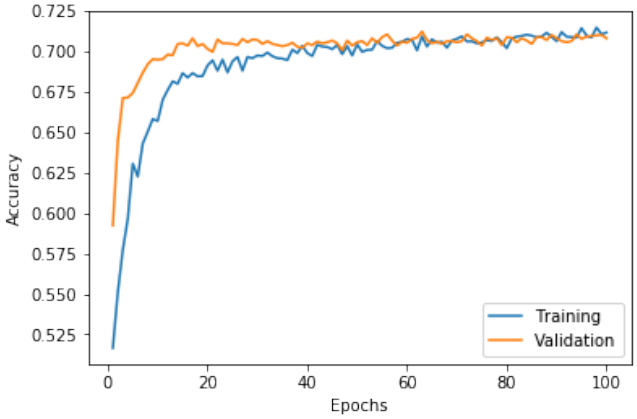

Figura 6: Valori di Accuracy per Training e Validation su 100 epoche

- un valore di **F-measure** pari a 0.67 nel training, 0,66 nella fase di validation.

A seguito della fase di fitting del modello, è stato proposto alla rete un insieme di osservazioni pari al 20% dell'intero file (contenute nelle due componenti *X_test* e *y_test*) *train.csv* su cui è stato effettuato un test ulteriore per interpretare la bontà del modello su osservazioni mai viste prima.
I risultati ottenuti sono i seguenti:

- **Test Loss: 0.573**
- **Test Accuracy: 0.776**
- **Test F-Measure: 0.524**

Si osserva nella seguente immagine i valori che sono stati ottenuti di misure pesate e misure specifiche per ogni valore della classe binaria.

	precision	recall	f1-score	support
0	0.87	0.83	0.85	4205
1	0.49	0.57	0.53	1195
accuracy			0.78	5400
macro avg	0.68	0.70	0.69	5400
weighted avg	0.79	0.78	0.78	5400

Figura 8: Valori pesati delle metriche

Si osserva facilmente che le metriche pesate assumono buoni valori addirittura più elevati delle aspettative, ma nello specifico si osserva che le osservazioni etichettate con il valore 1 sono ancora difficilmente classificabili dal modello in maniera corretta (comunque migliori rispetto alle reti neurali precedenti).

4.2 Test

Per terminare, è stato effettuato il Preprocessing (descritto precedentemente) anche sui dati contenuti nel file *test.csv*, così da poter replicare la stessa condizione dei dati del training; a seguito di ciò i dati sono stati forniti in input alla rete neurale che, tramite una procedura *predict_classes(test)*, ha classificato le osservazioni del file *test.csv*. I risultati della classificazione sono poi stati scritti nel file *Paolo_Mariani_800307_score1.txt* per la consegna.