ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA

in
Data Mining

Indice di Pericolosità Offensiva: un metodo di predizione per le partite di Calcio

CANDIDATO
Alessio Demeli

RELATORE:
Chiar.mo Prof.
Claudio Sartori

CORRELATORE:
Andrea Carmè

Sessione III

Anno Accademico 2018/2019

Sommario

Introduzione

La nascita di nuove tecnologie in grado di generare massive quantità di dati, insieme ad uno sviluppo parallelo di componenti, metodologie e paradigmi innovativi, hanno portato ad una sostanziale rivoluzione nell'ambito della raccolta, elaborazione ed estrazione di conoscenza dai dati.

Proprio per questa l'ambito del Data Mining acquisisce una sempre più grande importanza al giorno d'oggi, nella maggior parte degli ambiti della conoscenza, che sia per acquisire un vantaggio in ottica aziendale o in generale di business o che sia per ricerca. I campi di applicazione spaziano dall'ambito medico, con macchinari in grado di fare diagnosi con un altissimo grado di accuratezza e precisione, a quello della moda, pubblica amministrazione e per finire anche in ambito sportivo.

Fra le tecnologie, non propriamente dell'ambito del Data Mining, più in voga in questo momento troviamo le reti neurali, in particolare quelle che si chiamano Deep Network, con risultati notevoli riscontrati in tanti campi come il riconoscimento di immagini, la predizione di malattie dai dati dei pazienti, la traduzione automatica di testi, il gioco.

Per questo motivo, considerando che in ambito sportivo alcune tecnologie di Machine Learning ed Analytics sono già utilizzate con successo, ci si è chiesti se e come si potesse sfruttare la potenzialità di questi strumenti anche in ambito calcistico. Da qui, grazie anche alla collaborazione di una importante società italiana, di nome SiCS, che si occupa di raccolta, elaborazione ed analisi dati in ambito calcistico, si è deciso di proseguire ed esplorare la strada dell'utilizzo delle reti neurali per predire le partite di calcio, con buoni risultati.

La tesi dunque si diramerà in quattro capitoli, nel capitolo 1 si introdurrà in generale il Data Mining, cioè l'attività in cui ricadono tutte quelle tecnologie e tecniche per l'estrazione di conoscenza da dati grezzi, successivamente nel capitolo 2 si tratteranno in maniera

approfondita le reti neurali, strumento con il quale si è deciso di predire le partite e se ne parlerà facendo un'introduzione sulle reti più famose e di successo, insieme ad una più formale descrizione matematica e di tipologie di strumenti che le reti mettono a disposizione. Il capitolo 3 tratterà brevemente l'indice sul quale si basa il lavoro di tesi e si parlerà della società fornitrice dei dati e degli ideatori di questo indice, infine nel capitolo 4 si introdurranno i principali strumenti di sviluppo delle reti, insieme ai principali parametri di configurazione ed algoritmi usati, con una parte finale di presentazione di risultati di performance di predizione dei match. Le conclusioni presenteranno i possibili scenari futuri ed un confronto fra le reti sviluppate, così da portare alla luce i miglioramenti apportati dalle reti neurali in materia di predizione.

1. Il Data Mining

Storicamente il termine Data Mining è nato negli anni ’90, dove il termine era una sorta di abbreviazione per “analisi matematica su un database di grandi dimensioni”[1]. Lo scopo principale del Data Mining è dunque quello di effettuare estrazione di conoscenza e rivelazione di schemi non visibili o impliciti, dove l’obiettivo della scoperta di tali legami o informazioni, dipendono nella sua totalità dagli obiettivi degli Enti che la effettuano.

Il Data Mining è multidisciplinare, visto che per permettere l’analisi, l’estrazione e l’individuazione di schemi si avvale di tecniche, algoritmi, metodologie provenienti da vari settori della conoscenza:

- Intelligenza artificiale;
- Machine Learning;
- Statistica e matematica avanzata;
- Tecniche e tecnologie per i DBMS.

La materia dunque è applicata nei più disparati settori dell’industria, dalla sanità ai trasporti, passando per sicurezza, vendite, manutenzione, analisi di mercato avanzate e il suo utilizzo non fa altro che aumentare anche a causa dell’avanzamento di nuove tecnologie[2].

E’ abbastanza evidente come la diffusione di Social Network, sensoristica avanzata, facilità di reperire e memorizzare dati, presenza di connettività Internet capillare abbiano permesso e “costretto” il proliferare di nuove tecniche, algoritmi, tecnologie di elaborazione dati, dove i Big Data ne rappresentano un caso emblematico.

1.1. Attività del Data Mining

Il Data Mining quindi può essere visto come un processo, composto di varie fasi, dove in ognuna di queste fasi si eseguono determinate attività (task) ben definite e conosciute. Ogni attività ha le sue peculiarità in base al problema che si occupa di risolvere e sul come lo risolve, dunque un processo può facilmente richiedere l’esecuzione di più “task” diversi. Fra le attività conosciute e diffuse del Data Mining ci sono:

- Regressione, dove si stima il valore (numerico) di un determinato attributo, partendo dagli altri già presenti per la popolazione sotto esame;
- Classificazione e stima della probabilità di una classe, dove si stabilisce l'appartenenza di un individuo ad una determinata "categoria" nel primo caso e nel secondo si stima anche la probabilità di appartenere a quella categoria (ad esempio nel primo caso cerco di stabilire, data la cartella clinica di un paziente, che malattia ha e nel secondo caso stimo anche con che probabilità, per ogni possibile malattia);
- Calcolo della similarità, che si occupa di dare un punteggio su quanto due individui si "assomigliano", definita una certa misura di similarità;
- Clustering, in cui si "raggruppano" individui di una popolazione con caratteristiche simili;
- Association Rule Discovery, in cui si cerca di trovare le associazioni non banali fra individui, dato un insieme di transazioni in cui compaiono insieme (ad esempio scoprire che l'acquisto di una tipologia di crostacei porta a comprare anche un determinato gruppo di dvd);
- Data Reduction, in cui si cerca di ridurre la quantità di dati disponibili appunto, preservandone il più possibile significatività ed importanza (si ha un compromesso fra perdita di informazione utile contro maggior "visione d'insieme" e facilità di elaborazione guadagnata);
- Modellazione causale, si stima l'impatto di certe azioni su altre (per esempio quanto una campagna pubblicitaria possa realmente influenzare la vendita di un prodotto);
- Profiling, si cerca di individuare un comportamento tipico in una popolazione o comunque quelli (se più di uno) che la caratterizzano (identificare buoni, discreti, cattivi correntisti per una banca, rilevare i comportamenti tipici di anomalie in una determinata strumentazione);

- Predizione e link analysis, dato un grafo si cerca di stabilire le connessioni mancanti fra nodi (se i nodi sono utenti e le connessioni rapporti di amicizia, equivale a trovare o predire amicizie mancanti, partendo da quelle già esistenti).

Le attività di Data Mining che saranno maggiormente oggetto della tesi sono in particolare classificazione e data reduction.

1.2. La Classificazione

La classificazione è l'attività del Data Mining che si occupa di stabilire le classi di individui a cui una popolazione appartiene, dove per popolazione si può intendere un qualunque gruppo o fenomeno di interesse. Possiamo immaginare la popolazione come un gruppo di pazienti di un ospedale, un insieme di segnali provenienti dallo stesso sensore oppure un gruppo di squadre di calcio, per i quali abbiamo informazioni raccolte in un qualche formato e supporto elaborabile digitale.

Identificato cosa si intende per popolazione, parliamo delle classi. Per classi, in genere, si intendono un insieme di caratteristiche che vogliamo identificare negli individui di una popolazione e possono essere esiti di partite nel caso si abbiano squadre piuttosto che prognosi di determinate malattie a pazienti.

In base alla natura del problema e della popolazione, si hanno due gruppi in cui possiamo dividere le tecniche e algoritmi di Data Mining:

- Tecniche per apprendimento supervisionato;
- Tecniche per apprendimento non supervisionato.

L'apprendimento in se per se, soprattutto in ambiente di Intelligenza Artificiale, prevede una terza tipologia, chiamata apprendimento per rinforzo (Reinforcement Learning [fig.1]).

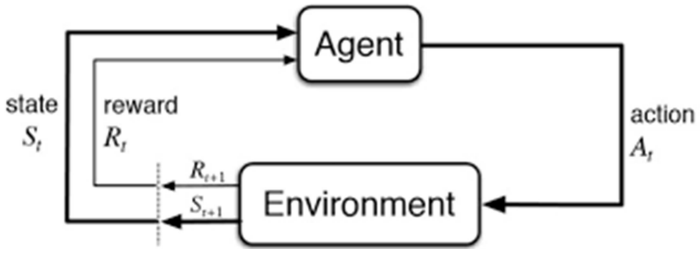


Figura 1

In questo tipo particolare di apprendimento c'è un agente che esegue delle azioni in un ambiente, dato uno stato iniziale, cercando di ottenere la più alta ricompensa possibile. Utile ad esempio nel caso di sviluppo di un agente che deve imparare a giocare, con lo scopo di imparare una strategia vincente (avere la massima ricompensa da ogni azione ed osservare lo stato in cui si viene a trovare in risposta all'azione stessa).

Per quanto riguarda l'apprendimento supervisionato e non, la differenza sostanziale è nella presenza di una specifica classe o categoria da individuare o meno nella popolazione. Se infatti siamo nel caso supervisionato, la popolazione avrà un determinato numero di categorie in cui sarà classificabile mentre nel caso non supervisionato saranno assenti.

In questo caso, l'apprendimento utilizzato sarà quello supervisionato, in quanto la natura stessa della predizione di partite porta l'identificazione di tre categorie da predire, cioè vittoria, pareggio o sconfitta.

1.2.1. Apprendimento supervisionato

Questo tipo di apprendimento prevede l'esistenza di categorie nella popolazione in esame, come già discusso, e ci sono due modi per ottenerle:

- Informazioni da esperti, in cui le categorie ci vengono indicate da esperti del dominio, per ogni individuo, su cui costruiremo il modello di classificazione per tutti gli individui che invece non sono stati già classificati;
- Dalla storia precedente, ovvero inizialmente non si hanno le categorie e non vengono fornite ma, passato del tempo, una volta conosciute le classi reali, possono essere

classificate a posteriori ed essere utilizzate per costruire modelli di classificazione per i nuovi futuri dati.

Ecco un esempio visuale di classificazione ed apprendimento supervisionato (fig.2).

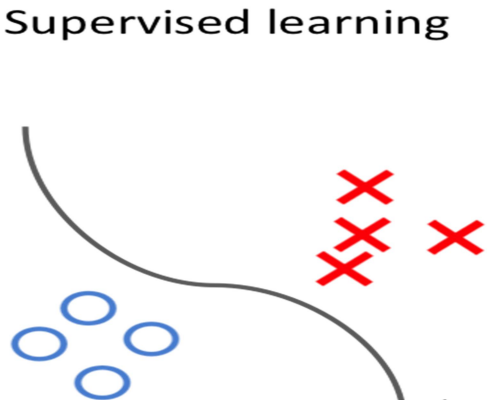


Figura 2

L'immagine sopra mostra in breve come un modello di classificazione, una volta addestrato, riconosca le categorie dei vari nuovi individui in ingresso, separandoli nelle classi di appartenenza.

Fra le tecniche di apprendimento supervisionato ci sono gli alberi decisionali, le reti neurali, classificatori Bayesiani. Ecco un esempio di albero decisionale (fig.3):

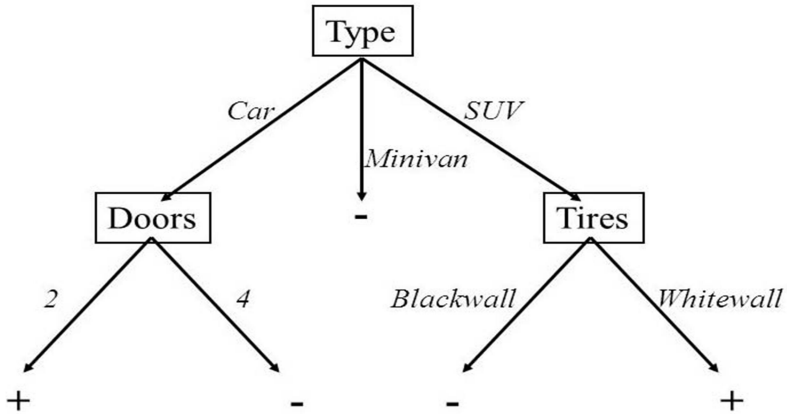


Figura 3

Ogni algoritmo utilizza le sue metriche, così da decidere che attributi usare per classificare al meglio le istanze sottoposte. Nel caso dell'algoritmo C4.5 (algoritmo per alberi decisionali, fra

i principali di sempre per il Data Mining [3]) si utilizza l'information gain (fig.4) massimo, ovvero la differenza in entropia fra le istanze attuali e quella condizionale data dall'uso di uno specifico attributo, per scegliere come classificare ogni nuovo individuo. Al crescere dell'information gain, si hanno maggiori possibilità di classificare correttamente le nuove istanze minimizzando l'errore di predizione della classe (anche se in realtà bisogna tener conto del fenomeno noto come adattamento eccessivo od overfitting, non trattato per C4.5).

$$IG(Y|X) = H(Y) - H(Y|X)$$

Figura 4

Algoritmi e tecniche supervisionate ce ne sono in abbondanza, ai giorni d'oggi spiccano le reti neurali (trattate nel Capitolo 3).

Introdotte le tecniche supervisionate, ora è il momento di parlare dell'attività di Data Mining nota come riduzione della dimensionalità.

1.3 Riduzione della dimensionalità

L'attività di riduzione della dimensionalità non riguarda solo il Data Mining, ma è da tenere in considerazione ogni qualvolta si fa uso di spazi multidimensionali per rappresentare i dati. Ipotizzando che ogni individuo sia un vettore X_i abbiamo che $X_i \in R^d$, dove d rappresenta il numero di dimensioni (attributi) della popolazione, i l'i-esimo individuo e R^d lo spazio vettoriale contenente la popolazione; in questo scenario esiste un fenomeno noto come maledizione della dimensionalità, che causa una forte sparsità nello spazio vettoriale R^d, rendendo necessario reperire un numero di individui esponenziale nel numero di dimensioni per ottenere buone performance di classificazione[4].

Fra i problemi introdotti dalla maledizione della dimensionalità abbiamo anche l'inefficacia che induce nelle attività basate sulla distanza (ad esempio nel clustering), in quanto all'aumentare del numero di dimensioni, il rapporto fra la distanza dei due punti più lontani

e i due punti più vicini si avvicina a zero (rendendo di fatto la distanza non significativa nel considerare due elementi simili o diversi fra loro).

Le considerazioni di cui sopra rendono quindi necessario utilizzare tecniche di riduzione della dimensionalità (riduzione del numero di attributi). Possiamo suddividere tale in altre due sotto-attività:

- Selezione delle caratteristiche;
- Estrazione delle caratteristiche.

Nel caso della selezione, si cerca di ridurre il numero di attributi scegliendo quelli considerati più significativi ed eliminando quelli ritenuti superflui mentre nel caso di estrazione si ottengono nuovi attributi come composizione degli originali, tramite qualche tipo di operazione lineare o anche non lineare (il metodo è utile solo se riduce il numero totale di attributi).

Per quanto riguarda la selezione delle caratteristiche, abbiamo quattro possibili approcci [5]:

- Forza Bruta, in cui si usano, uno alla volta, tutte le possibili combinazioni di attributi, misurandone ogni volta le performance rispetto il modello sviluppato e si sceglie la migliore;
- Wrapper, in cui si sceglie la combinazione di attributi che restituisce le migliori performance rispetto la tecnica di Data Mining usata, senza provare però tutte le possibili come nel caso Forza Bruta;
- Embedded, in cui la selezione degli attributi è parte integrante della tecnica di Data Mining (algoritmo C4.5);
- Filter. la selezione degli attributi avviene prima dell'esecuzione del modello di Data Mining.

Altre tecniche, locali, di selezione delle caratteristiche sono l'eliminazione degli attributi ridondanti e di quelli inutili. Gli attributi ridondanti sono quelli che sono in buona parte, se non completamente, contenuti in altri e dunque servono solo a duplicarne, o quasi, il

contenuto (ad esempio se si ha un attributo nome, uno cognome ed uno nome completo, i primi due possono essere eliminati in quanto contenuti nel terzo). Attributi inutili sono invece quelli ininfluenti a scopi di classificazione, come per esempio un attributo numero di carta di credito per stabilire se una persona soffra di una determinata malattia.

Per quanto riguarda l'estrazione di caratteristiche, fra le principali tecniche c'è la PCA (Principal Component Analysis).

1.3.1. Principal Component Analysis

E' una delle tecniche più usate per la riduzione della dimensionalità dei dati. Considerando infatti gli n individui di una popolazione (appartenenti allo spazio vettoriale R^d) ed i d attributi da cui sono descritti, questo tipo di procedura statistica ci permette di ottenere quelli che si chiamano i componenti principali.

In dettaglio l'estrazione dei componenti principali avviene eseguendo trasformazioni ortogonali sullo spazio originale R^d, ottenendo così un insieme ridotto di attributi linearmente non correlati fra loro. Per ottenere quindi il nuovo spazio dimensionale ridotto è necessario calcolare gli autovettori ed autovalori della matrice di covarianza (fig.5), così da avere in prima battuta vettori tutti linearmente non correlati fra loro, per i quali l'autovalore massimo sarà il primo componente (quello che "preserva" la maggior quantità di varianza)[6].

$$\mathbf{V} = \begin{pmatrix} \sigma_1^2 & \operatorname{Cov}(X_1, X_2) & \cdots & \operatorname{Var}(X_1, X_n) \\ \operatorname{Cov}(X_1, X_2) & \sigma_2^2 & \cdots & \operatorname{Var}(X_2, X_n) \\ \cdots & \cdots & \cdots & \cdots \\ \operatorname{Var}(X_1, X_n) & \operatorname{Var}(X_2, X_n) & \cdots & \sigma_n^2 \end{pmatrix}$$

Figura 5

Alla fine della procedura quindi si avranno le componenti, tutte ortogonali fra loro, che preservano la maggior parte della varianza e ciò che si ottiene equivale ad un nuovo spazio vettoriale R^p, in cui sono state eseguite le proiezioni dei vettori dallo spazio originale R^d, che

avrà dimensioni ridotte rispetto l'originale ma dove il contenuto informativo perso sarà il minore possibile [7].

Per avere i nuovi valori associati allo spazio vettoriale originale R^d e non nelle nuove coordinate date da R^p, è necessario moltiplicare il nuovo vettore per il proprio autovettore. Essendo la PCA sensibile ad intervalli di valori differenti per i diversi attributi, è necessario prima di tutto calcolare lo Z-score[8] dello spazio R^d originale, ottenendo così, per ogni attributo, che la media dei valori sia zero e la varianza uno (dato il d-esimo attributo, la media dei valori assunti dagli n individui per quell'attributo sarà zero e la varianza uno).

Di seguito un'immagine riassuntiva di ciò che si ottiene con la PCA(fig.6).

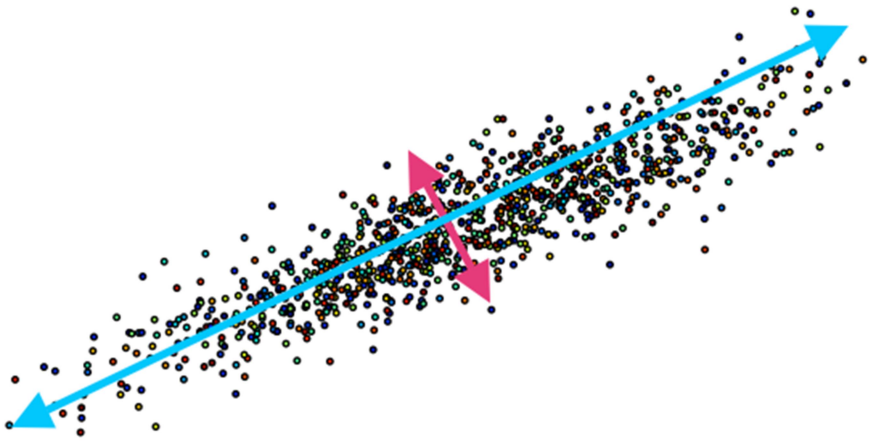


Figura 6

Nella figura sopra, possiamo vedere come siano indicati rispettivamente il primo componente (lungo l'asse disegnato in azzurro) ed il secondo, dove si può notare come il primo conservi la maggior parte della varianza ed il secondo la rimanente (dunque il primo avrà un autovalore di valore assoluto maggiore del secondo).

1.3.2. Altre tecniche di riduzione della dimensionalità

Ulteriori tecniche statistiche per ridurre la dimensionalità del dato esistono, fra quelle che prenderemo in considerazione ci sono:

- Campionamento dei dati;

- Aggregazione dei dati.

Nel caso dell'aggregazione dei dati, lo scopo principale è ridurre la variabilità, il rumore che caratterizza i dati e la dimensionalità, in quanto questo può permettere l'emergere di schemi più generali, meno affetti da rumore (anche dovuto semplicemente da errori, di qualunque tipo).

Il campionamento dei dati invece riguarda maggiormente la riduzione della quantità di dati analizzata e/o elaborata, visto che elaborare quantità di dati troppo grandi può richiedere, a volte, troppo tempo per un algoritmo di Data Mining oppure essere troppo costoso. Il campionamento può mantenere quasi inalterate le caratteristiche della popolazione da analizzare, se il campione è abbastanza rappresentativo, e le principali tipologie sono:

- Campionamento Casuale, in cui si estrae un singolo elemento dalla popolazione, data una certa funzione di distribuzione di probabilità nota(fig.7);
- Con ripetizione, in cui si ripetono estrazioni a campionamento casuale e gli elementi estratti vengono reinseriti (lasciando così inalterata la distribuzione di probabilità nella popolazione);
- Senza ripetizione, in cui si ripetono estrazioni a campionamento casuale ma senza reinserimento degli elementi estratti (dunque la funzione di distribuzione di probabilità cambierà ad ogni iterazione);
- Stratificato, si divide la popolazione in gruppi, secondo criteri specifici ed infine da ognuno di questi gruppi si esegue un campionamento casuale.

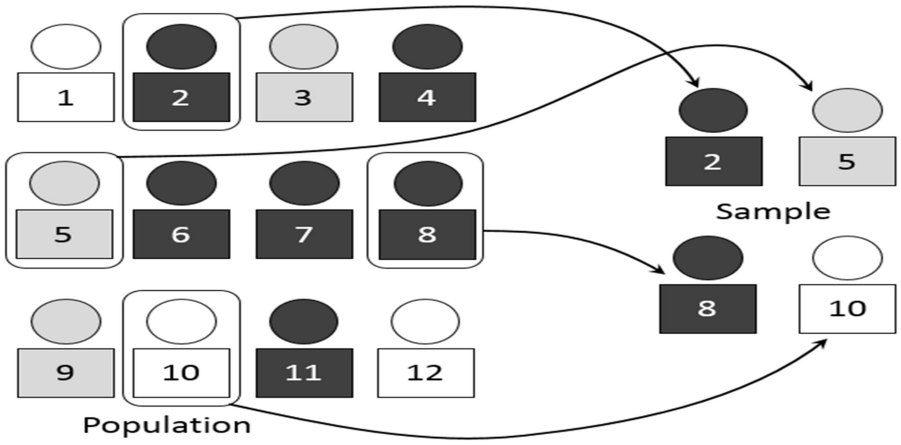


Figura 7

In statistica esistono tecniche per scegliere dimensione e numero ottimale di campioni, così da preservarne allo stesso tempo le caratteristiche principali ma elaborando i dati in un tempo minore. In generale, il campionamento ha un'importanza che va oltre la semplice riduzione della dimensionalità, in quanto un cattivo campionamento può portare a pessimi risultati di predizione, come ad esempio nel caso di scarso bilanciamento delle classi di interesse nella popolazione iniziale[9]. L'impatto misurabile nell'errore di predizione dello sbilanciamento, porta allo sviluppo di tecniche, anche di campionamento, mirate a mitigare ed alleviare questo problema (sarà trattato nel Capitolo 4).

1.4 Trasformazione di attributi

Come accennato precedentemente per la PCA(1.4.1), molti algoritmi ed in generale molti modelli di classificazione sono sensibili ai diversi intervalli dei singoli attributi, basta pensare alle attività di Data Mining che fanno uso di funzioni di distanza, dove attributi con range più ampi diventano più importanti ed influenzano dunque maggiormente il risultato della classificazione. Un altro esempio di sensibilità ai valori degli attributi è dato dalle reti neurali, dove valori a range molto diversi, possono portare ad un rallentamento della convergenza della rete.

Le tecniche più utilizzate per la trasformazione degli attributi sono:

- Standardizzazione;

- Normalizzazione.

Nel caso della standardizzazione, dato un individuo $X \in R^m$, il nuovo valore standardizzato sarà dato da $Z \rightarrow \frac{X-\mu}{\sigma}$, dove se la distribuzione di probabilità di X è Gaussiana, allora i valori Z seguiranno una distribuzione a "campana" di valor medio zero e deviazione standard uno ($\mu = 0,\ \sigma = 1$).

Nel caso della normalizzazione invece portiamo i valori in intervalli noti, ad esempio l'espressione $X \rightarrow \frac{X-X_{min}}{X_{max}-X_{min}}$ darà individui con intervalli di valori compresi fra zero ed uno.

Entrambe le tecniche, seppur causando traslazione, contrazione ed allungamento dello spazio dei valori, non ne cambiano la distribuzione, dunque si hanno i vantaggi in termini di correttezza e gestibilità per gli algoritmi di Data Mining, senza gli svantaggi dovuti dal cambio del modello probabilistico dei dati stessi.

1.5 Fasi di Estrazione, trasformazione e caricamento

Per quanto riguarda le sorgenti dati per gli algoritmi di Data Mining, possono esserci molteplici tipologie di fonti da cui possono essere recuperati:

- Database classici (RDBMS), con una qualche struttura concettuale, logica e fisica, implementato tramite un determinato strumento;
- Database non relazionali (NoSQL DB), adatti all'organizzazione ed immagazzinamento di grandi moli di dati differenti e non strutturati (come i Big Data);
- File semi-strutturati, con una qualche idea di gerarchia e struttura, automaticamente elaborabili (ad esempio file xml, file csv);
- File di testo semplici.

La diversità delle fonti, dei formati e dei paradigmi su cui sono costruiti questi strumenti, fanno si che sia necessario gestire i dati in modo uniforme ed appropriato, secondo un approccio metodologico ben definito. La qualità dei dati forniti in ingresso alle tecniche ed

algoritmi del Data Mining è fondamentale per ottenere buoni risultati, infatti una buona parte del lavoro è un'attenta analisi, pulizia, comprensione e trasformazione degli stessi, per evitare di cadere in quello che in gergo tecnico viene definito "garbage in, garbage out" (se diamo dati di scarsa qualità in ingresso a questi algoritmi, in uscita non possiamo aspettarci che la stessa cosa[10]). Le tre fasi, in inglese, sono note come ETL (Extraction, Transformation and Loading).

1.5.1. Estrazione

Durante questa fase ci si occupa in primis di reperire le fonti dati necessarie, in seconda battuta invece si procede al vero e proprio caricamento e salvataggio dei dati estratti in un'unica area. Questa fase è spesso accompagnata da una fase di pulizia (Cleansing), che ci restituisce i dati nel formato più appropriato, da essere poi elaborati nella fase successiva (quella di Trasformazione), ed è caratterizzata dalle seguenti attività:

- Eliminazione dei dati duplicati;
- Trattamento dei dati mancanti;
- Correzione di valori errati o impossibili in determinati campi o attributi (campo prezzo impossibile, ad esempio assume un valore negativo);
- Valori inseriti nei campi sbagliati (inserire nome e cognome dove c'è il campo data);
- Correzione di dati sbagliati (campo città con scritto "Bolognia" invece di "Bologna");
- Trattamento di valori nulli, molto importante per evitare errori di calcolo ed aggregazione impredicibili e difficili da individuare.

Finita la fase di pulizia, abbiamo la sorgente dei dati pronta per essere sottoposta a Trasformazione. Ecco un esempio di ciò che abbiamo terminata questa fase(fig.8):

Figura 8

1.5.2. Trasformazione

Durante questa fase ci si occupa di migliorare ulteriormente la qualità dei dati, effettuando una serie di procedure:

- Conversione dei vari formati sorgente in uno unico, secondo le esigenze di elaborazione dati dei vari algoritmi di Data Mining (per formato si intende sia quello di memorizzazione fisica, sia a livello di come è strutturato il dato a livello logico, come ad esempio la trasformazione di attributi categorici discreti in vettori binari);
- Normalizzazione delle informazioni, in cui si cerca di ottenere uniformità dei dati, come ad esempio cambiando le unità di misura di specifici campi o attributi se ne contengono diverse per lo stesso oggetto o individuo;
- Selezione degli attributi e record di interesse, in base al problema che dobbiamo affrontare (riduzione del numero dei dati, per velocizzarne l'elaborazione e selezione degli attributi di interesse per il problema);
- Integrazione delle informazioni, ad esempio unendo informazioni di uno stesso dato provenienti da fonti diverse, così da avere un dato finale corretto ed unico;
- Aggregazione dei dati secondo le metriche e misure che ci interessa recuperare.

Al termine delle procedure sopra descritte, abbiamo i dati pronti per essere memorizzati nella fase finale del Caricamento.

1.5.3. Caricamento

Durante questa fase, avviene il caricamento vero e proprio dei dati nel formato scelto. Le modalità con cui i dati vengono memorizzati sono di due tipi:

- Aggiornamento, in cui vengono aggiunti, ai dati attuali, solo le ultime modifiche effettuate;
- Ricaricamento, in cui vengono eliminati i dati precedenti e si scrivono tutte le nuove informazioni.

Nel caso dell'aggiornamento quindi si tiene traccia dei dati già in nostro possesso e vengono integrati i vecchi con i nuovi mentre nel caso del ricaricamento si hanno nuovi dati dopo ogni cambiamento.

Tutti il processo di Data Mining segue, in realtà, un determinato standard aperto, fra i più diffusi anche a livello industriale, chiamato CRISP-DM. Nel prossimo paragrafo verrà introdotto sia per importanza e sia per la generalità che lo rende fra i più usati nel campo del Data Mining (compreso nella tesi in questione) [11].

1.6 Cross Industry Standard Process for Data Mining (CRISP-DM)

Questa metodologia, come precedentemente accennato, è diventata di successo in quanto è agnostica agli strumenti, industria ed applicazioni di utilizzo, il che ne permesso un uso diffuso per chi implementa soluzioni di Data Mining. Un po' più in dettaglio la metodologia si divide in:

- Business Understanding, in cui si formula il problema a livello concettuale e si pensa allo scenario di applicazione;
- Data Understanding, in cui si valutano le fonti di dati disponibili e se ne valuta il costo per ottenerle;

- Data Preparation, dove si effettuano le operazioni di ETL per preparare, sistemare, aumentare la qualità dei dati grezzi iniziali e portarli nel formato di nostro interesse, in base ai modelli di Data Mining usati;
- Modeling, in cui si sviluppa il modello di Data Mining vero e proprio, in base ai dati preparati e raccolti;
- Evaluation, nella quale valutiamo la qualità del modello o dei modelli sviluppati, in base all'impatto stimato sul business, sulla loro utilità, quantità ed importanza degli errori effettuati;
- Deployment, dove i modelli sviluppati e valutati vengono realizzati a livello software, per ottenerne un qualche vantaggio o comunque per utilizzarli con gli scopi formulati all'inizio nel business understanding.

Le varie fasi sono cicliche, in particolare Business Understanding e Data Understanding possono susseguirsi più volte, prima di partire con la preparazione dei dati, così come Data Preparation e Modeling, in quanto se il modello sviluppato risponde male o non funziona, può essere necessario rivedere la fase di elaborazione dei dati a disposizione ed Evaluation con Business Understanding, in quanto se il modello ha pessime performance, può essere necessario raffinare il problema o riformularlo completamente. L'insieme dei passaggi stessi, possono essere svolti di nuovo da capo, per scopi di espansione delle funzionalità esistenti. Di seguito un'immagine esemplificativa del processo (fig.9).

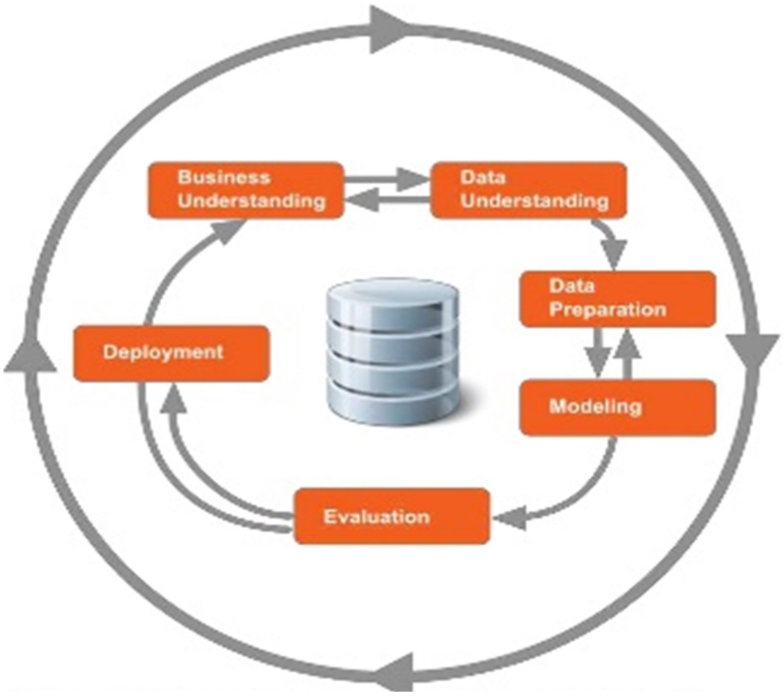


Figura 9

L'ultima parte di questo capitolo riguarderà proprio la parte di valutazione delle performance di un modello, soprattutto nel caso di apprendimento supervisionato con scopi di classificazione e predizione.

1.7 Valutazione delle performance

La fase di valutazione delle performance di un modello è chiaramente molto importante nell'attività del Data Mining e le seguenti tecniche sono indipendenti dagli strumenti ed algoritmi usati.

Nella fase di costruzione di un modello, in particolare se prendiamo in considerazione l'apprendimento supervisionato (essendo quello che caratterizza la predizione delle partite), si usa in genere quello che è chiamato training set, costituito da istanze di una popolazione già classificate, con lo scopo di addestrare il modello alla classificazione e/o predizione di nuovi individui, che costituiranno invece il test set (dalla quale dunque mancherà l'attributo che rappresenta la classe). Fatte queste premesse, avremo ovviamente un errore di classificazione, sia per il training sia per il test set, che dipende dalle metriche e funzioni utilizzate per valutarlo.

Nel caso del training set dunque, prima si addestra il modello con i dati di training, successivamente si verificano le performance di classificazione sugli stessi dati in termini di differenza fra classe predetta e reale, valutando la percentuale di errore (training set error). Lo stesso procedimento viene eseguito sul test set, calcolando dunque l'errore di predizione (test set error).

Ciò che può succedere è che il training set error sia sensibilmente minore del test set error, problema non da poco considerando come in genere sia proprio il tentativo di ridurre il primo a guidare la creazione ed evoluzione del modello di classificazione. Il fenomeno appena descritto prende il nome di overfitting, causato dall'eccessivo addestramento del modello stesso a riconoscere quella particolare popolazione, motivo per cui, alla minima variazione degli individui che la compongono, il modello fallisce nel classificarli correttamente (scarsa capacità di generalizzazione). La questione si può anche porre come complessità del modello, in quanto la troppa complessità che il modello assume rispetto a quella del problema, porta lo stesso a diventare molto sensibile al rumore e al minimo cambiamento casuale nella popolazione (fig.10).

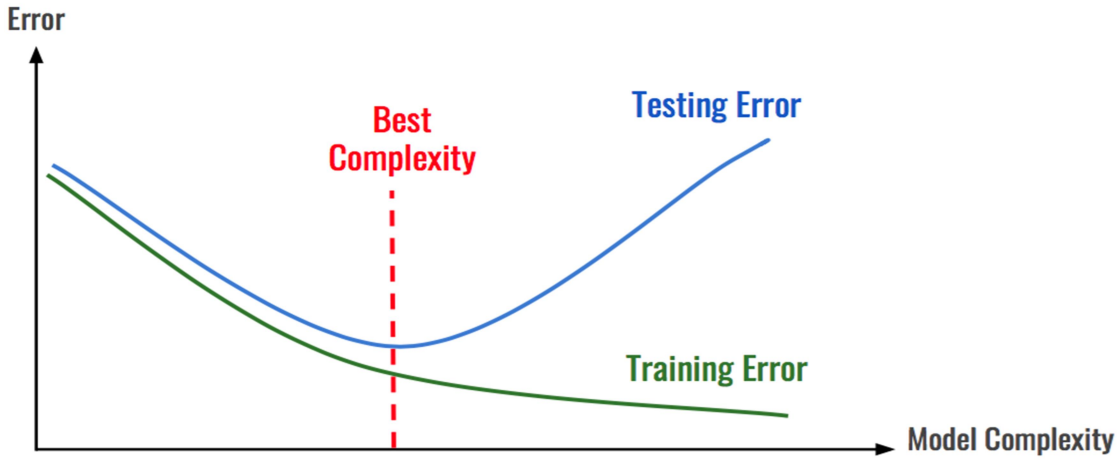


Figura 10

Le tecniche principali per prevenire l'overfitting dipendono anche dallo strumento che si usa, per citarne alcune:

- Dropout(fig.11), tecnica di regolarizzazione nelle reti neurali che si occupa di disattivare un certo numero di neuroni, scelti casualmente, in modo tale da introdurre del rumore nella fase di addestramento ed introdurre la variabilità necessaria a rendere la rete capace di generalizzare (le performance non peggiorano sensibilmente al variare dell'input);
- Minimum Description Lenght (MDL), formalizzazione del Rasoio di Occam alla teoria dell'informazione, in cui si afferma che la migliore ipotesi (in questo caso rappresentata da un modello ed i suoi parametri) per un insieme di dati è quella che porta alla maggior compressione dei dati stessi[12] (usato nel C4.5 per fare pruning dell'albero decisionale, così da prevenire overfitting).

Fatta dunque una breve carrellata delle tecniche per prevenire overfitting, ora passiamo alle tecniche invece utilizzate per effettuare una stima corretta delle varie tipologie di errore. Una cosa importante da tenere in considerazione, quando si valutano le performance dei modelli sviluppati, è sicuramente la necessità di indipendenza delle prove di valutazione da effettuare ed anche la necessità di stabilire una valutazione in base a più osservazioni, così da ottenere un indicatore maggiormente affidabile e preciso sull'errore reale.

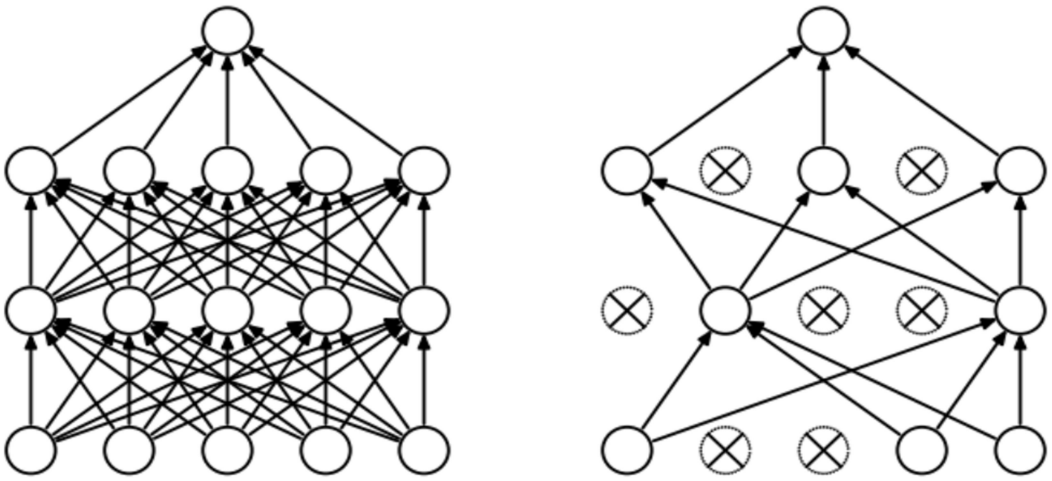


Figura 11

Tecniche che servono a garantire una migliore stima dell'errore sono:

- Cross-Validation (fig.12) o k-fold, in cui si divide il training set casualmente in k parti e se ne usa una come validation set e le altre k-1 come training set, ripetendo il procedimento k volte. In ognuna delle iterazioni, si prende una parte diversa come validation set ed infine si valuta l'errore medio di predizione ottenuto come media dell'errore nelle k iterazioni compiuto su ogni k-esimo validation set;
- Holdout (fig.13), dove dividiamo il dataset supervisionato originale in due parti, casualmente, di cui una per il training e l'altra per il test, così da riuscire a stimare le performance del modello prima ancora di avere un test set. Una suddivisione casuale potrebbe alterare la proporzione delle classi dal dataset originale a quelli di training e test, per evitare questo rischio si può ricorrere alla tecnica di campionamento stratificato;
- Leave one out, caso estremo di k-fold in cui il valore di k è pari ad N (con N numero di elementi della popolazione).

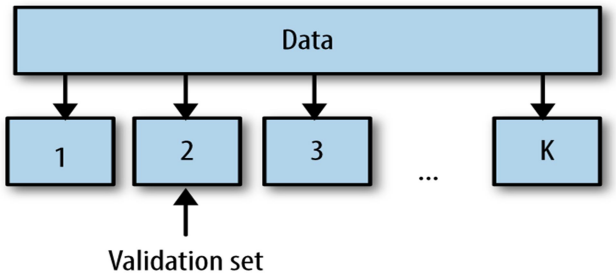


Figura 12

In figura 12 abbiamo un esempio di k-fold applicato ad una delle k parti, in una generica iterazione.

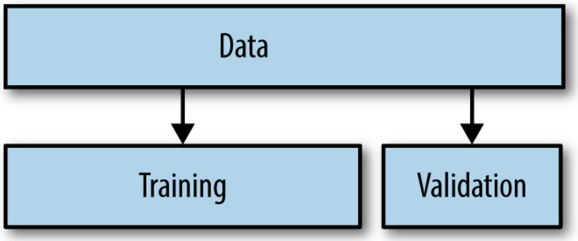


Figura 13

Nella figura 13 vediamo un semplice caso di holdout, in cui parte dei dati viene usata con scopi di validazione (dunque come validation set).

L'introduzione del validation set, permette di avere una sorta di anteprima delle performance sul test set, così da indurre ad una revisione anche immediata del modello in caso che si abbia errore elevato.

I rimanenti indicatori di prestazioni riguardano più la valutazione corretta della bontà dei modelli, rispetto alla corretta stima in se per se dell'errore o della giusta complessità del modello, fornendo degli indicatori qualitativi di capacità di predizione corretta.

L'impatto dell'errore sullo scenario di utilizzo dell'applicazione può variare notevolmente, per questo è molto importante usare le metriche giuste, così da guidare l'algoritmo di Data Mining nella direzione corretta anche in base al contesto di applicazione. Uno degli strumenti utili ad avere una sorta di sommario sulle performance dei classificatori è la matrice di confusione(fig.14), utile a capire le performance del modello nella predizione di ogni singola classe. La matrice infatti permette di calcolare tutti gli indicatori necessari ad una completa valutazione delle performance, tenendo traccia dell'errore compiuto dal modello di predizione rispetto alle classi reali di appartenenza.

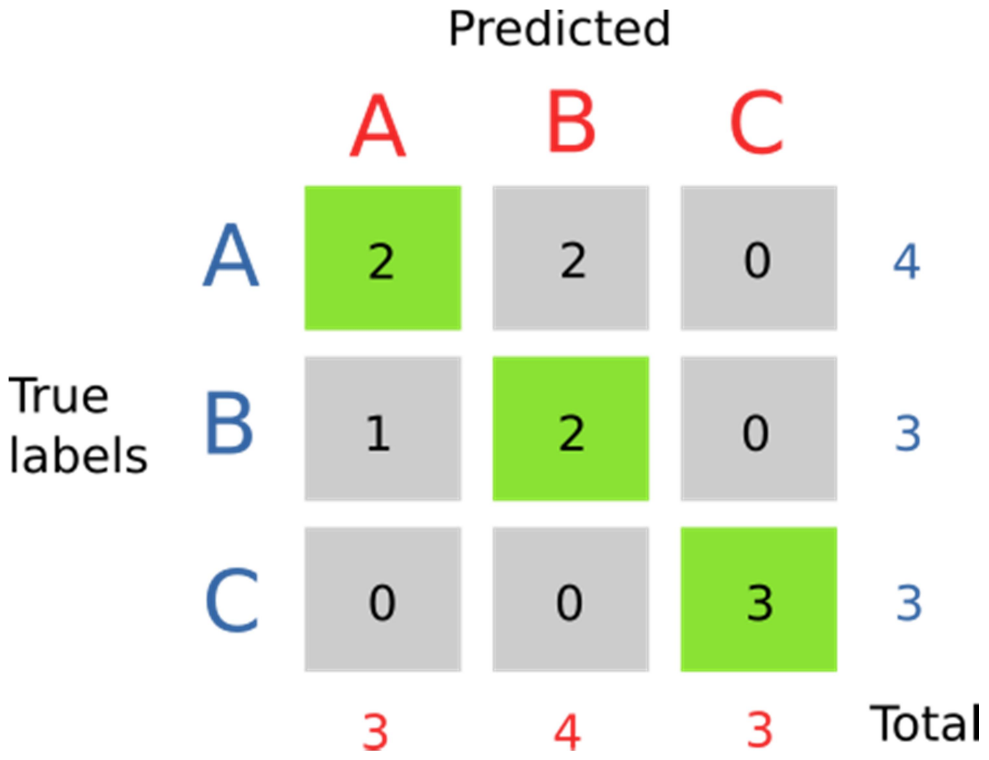


Figura 14

Come si può notare in figura 14, la matrice permette di confrontare il numero di classi predette dal modello rispetto a quelle reali, in più per ogni singola classe abbiamo anche il numero di predizione corrette (sono nella diagonale, TP_i) e tutte le scorrette in colonna, con il dettaglio di quale classe abbiamo scambiato con un'altra durante la predizione. Ciò permette di essere estremamente precisi nei calcoli delle performance e garantisce l'uso di una vasta gamma di indicatori diversi. I valori a destra rappresentano il totale dei valori reali per quella classe (T_i) mentre in basso abbiamo il totale dei valori predetti dal nostro modello (P_i).

Le principali tecniche di misurazione delle performance dunque sono:

- Precision, con cui misuriamo, per ogni classe, la frazione di predizioni corrette sul totale delle predizioni effettuate dal modello (formula $\frac{TP_i}{P_i}$, per esempio 2/3 per la classe A);

- Recall, con la quale misuriamo la frazione di predizioni corrette sul totale delle classi totali reali (formula $\frac{TP_i}{T_i}$, 2/4 sempre per la classe A);
- Accuracy, ovvero la quantità di predizioni totali corrette (formula $\sum_1^{n\_classes} \frac{TP_i}{N}$, dove "n_classes" è il numero totale di classi da predire, 7/10 in questo caso);
- F1_score, cioè la media armonica di precision e recall per una data classe (formula $2 * \frac{recall_i * precision_i}{recall_i + precision_i}$, 0,57 in questo caso per la classe A);
- Kappa statistic, che misura la concordanza fra due classificazioni, in questo caso il rapporto di quanto il nostro classificatore eccede in concordanza il classificatore casuale, rispetto l'eccedenza del classificatore perfetto con il casuale (formula $\frac{\Pr(c) - \Pr(r)}{1 - \Pr(r)}$, dove Pr(c) è l'accuracy mentre $\Pr(r) = \frac{T_A * P_A + T_B * P_B + T_C * P_C}{N^2}$).

Questo insieme di statistiche si rende necessario per l'importanza della valutazione stessa dei modelli di classificazione, in quanto metodi come la precisione, rischiano di restituire risultati fuorvianti in caso di classi fortemente sbilanciate nel test set.

Le reti neurali sono fra i modelli di classificazione, in particolare per l'apprendimento supervisionato, di maggior successo della storia recente, motivo per il quale saranno l'argomento del capitolo successivo.

2 Machine Learning e Reti Neurali

Le reti neurali sono fra gli strumenti attualmente di maggior successo e stanno vivendo una forte diffusione in ogni campo della conoscenza.

I campi di applicazione delle reti neurali vanno dal riconoscimento visuale di oggetti, fino alla produzione di opere d'arte (seppur non capolavori, ma ottime imitazioni), riconoscimento vocale e molto altro. I principali motivi di successo si possono identificare sia nella diffusione di quantità di dati elevatissime (Big Data), che permettono di costruire algoritmi e meccanismi di apprendimento sempre più sofisticati, completi e precisi, sia nella sempre maggiore quantità di risorse computazionali disponibili, con le nuove architetture hardware, calcolo parallelo e distribuito, abilitanti a nuove frontiere dell'apprendimento automatico (Machine Learning appunto).

Reti neurali di successo, per i campi sopra descritti, sono:

- AlexNet, che nel 2012 riuscì ad abbassare il tasso di errore nella competizione ILSVRC per il riconoscimento e classificazione di immagini di più del 10,8% rispetto al miglior concorrente dell'anno precedente, raggiungendo un tasso di errore del 15,3% [13];
- DeepDream, progetto nato in realtà non con lo scopo specifico di produrre arte e nonostante ciò ha mostrato capacità di riprodurre uno stile artistico, data un'immagine in input creata casualmente come del rumore bianco, simile a quello delle immagini con cui è stata addestrata per il riconoscimento di oggetti[14].

Nonostante ci siano profonde differenze fra le reti citate e ciò che è stato implementato nella tesi, soprattutto in termini di complessità e requisiti computazionali, questo ci aiuta a capire come le potenzialità delle stesse siano enormi.

Di seguito verranno introdotte formalmente le reti neurali.

2.1 Concetti generali e descrizione formale

Una rete neurale è, per certi aspetti, un'approssimazione del cervello animale in quanto è composto da nodi che simulano i neuroni (fig.15), da connessioni che simulano le sinapsi e da segnali, in questo caso digitali, che attraversano le connessioni e determinano la risposta del sistema a determinati stimoli[15]. Con approssimazione del cervello animale si intende che l'obiettivo, almeno originale, era quello di ottenere intelligenza nel comportamento simulando i meccanismi di elaborazione e di comunicazione tipici dei cervelli negli organismi biologici.

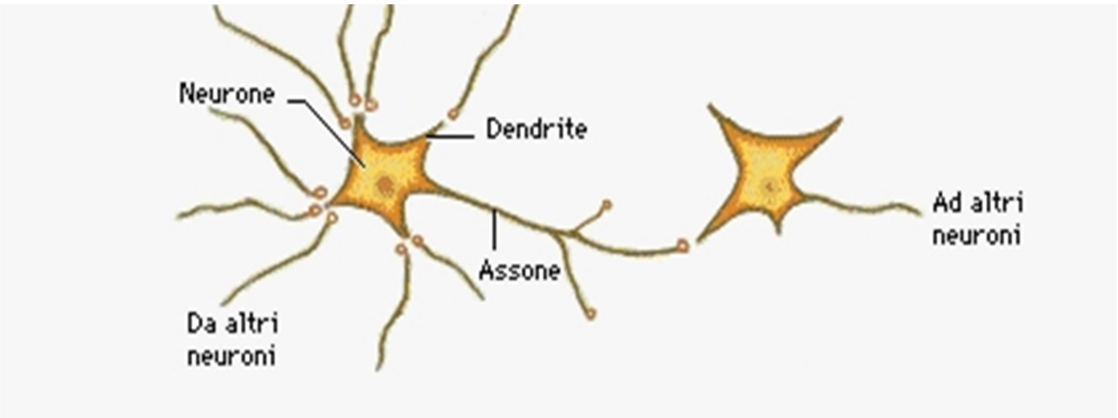


Figura 15

Le reti, di per se, non sono infatti un algoritmo di risoluzione ma più un ambiente che permette di essere utilizzato, da vari algoritmi, per risolvere problemi di Machine Learning e nello specifico anche di Data Mining. Si dice per questo che le reti neurali sono un sistema che segue un approccio "connessionista", dove dunque sono le connessioni, i pesi e l'esperienza acquisita (intesa come ingressi e risposte corrispondenti) a determinarne le capacità di apprendimento[16].

I componenti di una rete neurale dunque sono:

- Neuroni, ovvero le singole unità computazionale alla base di una qualunque rete (ogni neurone sarà indicato da $N_{i,j}$, dove i è il neurone e j il livello in cui si trova). L'input di un neurone sarà dato dalla somma pesata dei neuroni a lui connessi da un livello precedente;

- Connessioni, rappresentate da tante frecce che collegano un neurone ad altri neuroni, dove ogni connessione è identificata da un peso. Ogni peso moltiplicherà l'uscita del neurone da cui si origina verso il neurone destinazione (peso identificato da $W_{k,z}$, dove k è il neurone di origine e z il neurone destinazione);
- Funzioni di attivazione, cioè funzioni non lineari che moltiplicano il valore totale di uscita di un determinato neurone, trasformandolo in un numero reale compreso in un determinato intervallo ([0, 1], [0, +inf], eccetera);
- Livello, ovvero la posizione di un gruppo di neuroni rispetto ai dati in ingresso. I neuroni che prelevano in ingresso l'input grezzo, rappresentano il livello di input, quelli che restituiscono la risposta finale (nell'ultimo livello a destra) rappresentano il livello di output ed infine i livelli interposti fra i due sono chiamati livelli nascosti (possono essere più di uno).

Il neurone dunque è schematizzabile come una unità computazionale (fig.16), che riceve degli input pesati, li somma e restituisce un'uscita di valore reale dato dalla funzione di attivazione applicata a questa somma pesata (può essere presente anche il bias, utile a spostare la distribuzione di valori della funzione di attivazione). Il tipo di rete neurale (topologia) invece, cambia al variare del modo in cui connettiamo i vari neuroni fra loro.

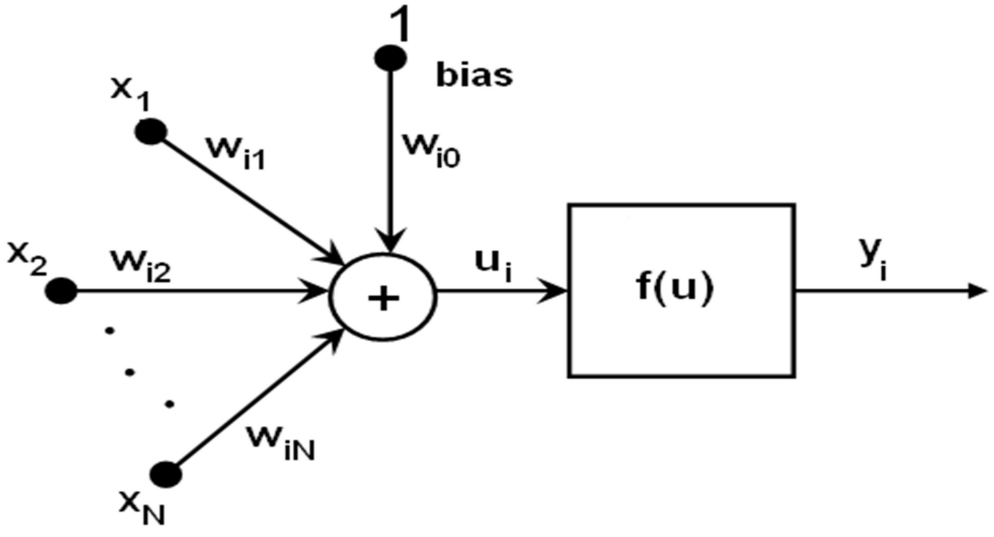


Figura 16

Topologie di reti neurali diverse, con funzioni e strutture anche radicalmente diversi, sono:

- Multi Layer Perceptron (MLP), ovvero una rete composta da almeno un livello di input, un livello di output ed uno o più livelli nascosti. Queste reti sono caratterizzate da connessioni solo fra nodi di livelli adiacenti, assenza di connessioni fra nodi di uno stesso livello e da assenza di cicli (quindi il livello uno è connesso al due, ma non vale il viceversa);
- Recurrent Neural Networks, nelle quali le connessioni possono essere invece anche fra neuroni dello stesso livello e di livelli non adiacenti, formando anche dei cicli o circuiti fra neuroni.

All'interno di queste due macro topologie in realtà, sono presente tante e differenti sottocategorie di reti, ognuna devota ad uno specifico obiettivo o portata a risolvere determinate tipologie di problemi.

Di seguito l'immagine di una rete neurale MLP (fig.17).

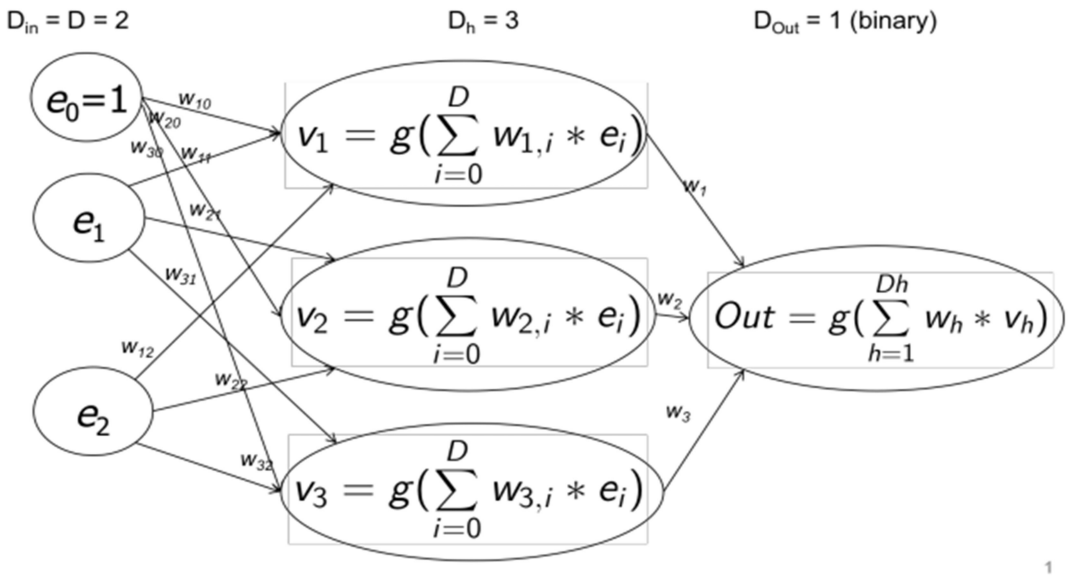


Figura 17

Introdotte a livello formale le reti, verrà spiegato in modo esaustivo il loro funzionamento nel caso di apprendimento supervisionato, ovvero quello di interesse per la classificazione delle partite di calcio.

2.2 Metodologia di apprendimento

Le reti neurali, per effettuare l'apprendimento vero e proprio, si basano, aldilà della topologia, sul concetto di Bakpropagation[17]. Questo è stato l'algoritmo fondamentale alla diffusione delle reti neurali, visto che permette un calcolo veloce ed immediato del contributo di errore per ogni livello di una qualunque rete. Questo metodo è essenziale per l'addestramento, dato che lo scopo principale in caso di apprendimento supervisionato è quello di trovare la giusta corrispondenza fra input ed output (dunque trovare le classi corrette dati vettori n dimensionali in ingresso, minimizzando l'errore di predizione).

Di seguito un'immagine della Backpropagation in azione (fig.18),

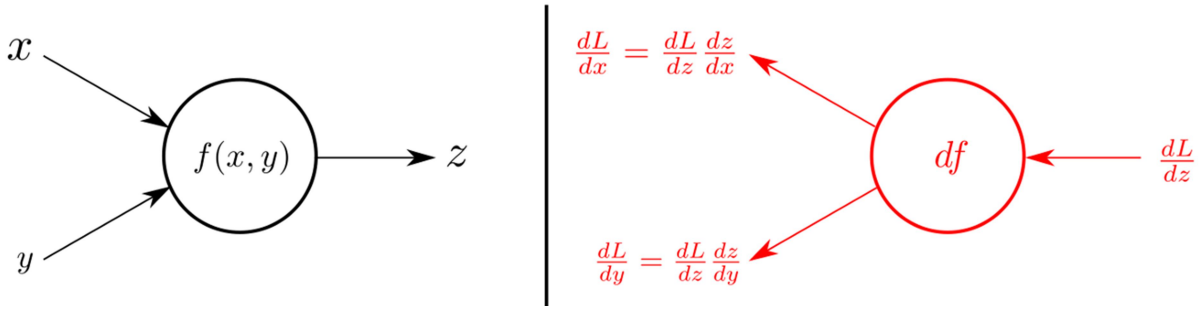


Figura 18

Come si può notare in figura 18, la parte a sinistra riguarda la normale propagazione dell'input attraverso i neuroni della rete e la parte a destra invece mostra come, data una funzione di errore (Loss), la quantità di errore si propaghi all'indietro secondo la chain rule (fig.19) delle derivate (dunque fornendo i singoli contributi di errore utili a cambiare i pesi attuali delle due connessioni, $W_{x,z}$ e $W_{y,z}$).

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

Figura 19

Una funzione di errore, di norma, dipende dalla natura dei dati (categorici o numerici) e dal numero di classi di cui dover calcolare l'errore, nonostante ciò il meccanismo di Backpropagation è generale e funziona in tutti i casi, con la sola incognita del metodo di discesa del gradiente utilizzato (ovvero di quanto e come aggiorno i pesi per minimizzare l'errore), di cui esistono diverse implementazioni.

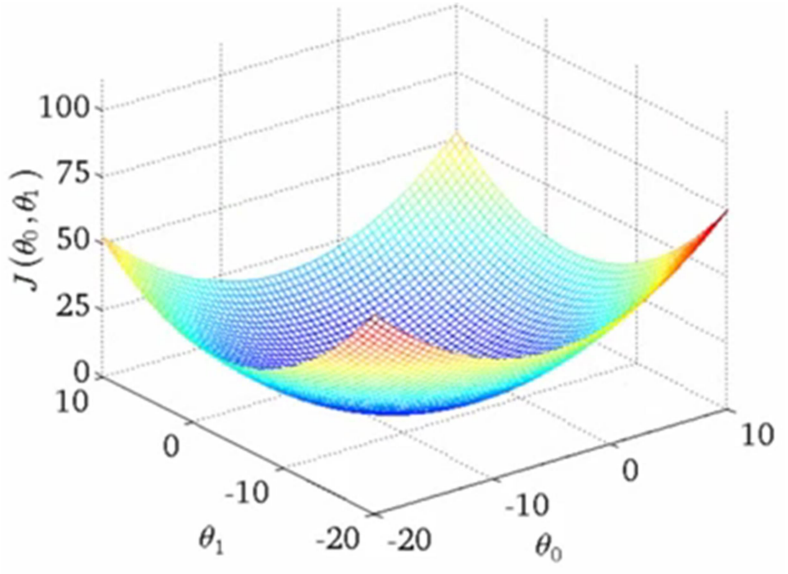


Figura 20

In questo caso vediamo un profilo di una generica funzione di errore (fig.20) dove il nostro obiettivo è minimizzarlo, correggendo i pesi della rete in modo da raggiungere il punto di errore minimo partendo da quelli attuali, tramite l'uso di uno specifico algoritmo di propagazione del gradiente applicato ai pesi di ogni livello, tramite Backpropagation (θ_0 e θ_1 rappresentano i pesi della nostra rete mentre $J(\theta_0, \theta_1)$ è la funzione di errore).

Una generica funzione di correzione dei pesi è data dalla formula nella figura 21.

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{1}^{m} L_m(w)$$

Figura 21

In figura 21, la sommatoria è dovuta dal fatto che l'aggiornamento dei pesi viene eseguito dopo m passi di classificazione mentre L_m rappresenta una generica funzione di errore, come l'errore quadratico medio (Minimum Squared Error, MSE, fig.22). Il parametro α rappresenta il learning rate, uno dei parametri fondamentali nelle reti, cioè la frazione di errore da prendere in considerazione per correggere i pesi (non si conosce la funzione di distribuzione di probabilità dell'errore, per questo in via cautelare si usa il learning rate).

$$\mathbf{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Figura 22

In figura 22, y_i rappresenta la classe predetta dal nostro modello mentre $\tilde{y}_i$ rappresenta quella reale e la sommatoria è il calcolo dell'errore per ogni possibile classe da predire (tutto diviso per il numero di classi, in quanto è un errore medio).

Tornando per un attimo a trattare di funzioni d'attivazione, un ulteriore punto a favore nelle reti neurali sta nell'uso di funzioni non lineari, che permettono di trovare corrispondenze fra input n dimensionali molto grandi, sparsi e classi di uscita, grazie alla non linearità della trasformazione input/output. Altri vantaggi nell'uso di funzioni non-lineari sono, come già detto, lo schiacciare gli input in range ben definiti ma, nel caso delle reti neurali, abbiamo in più l'utilizzo di funzioni con derivate del primo ordine velocemente calcolabili, cosa

indispensabile per ridurre la complessità del calcolo di derivate per funzioni non lineari e il costo totale dato dall'enorme numero di neuroni di cui calcolare il contributo di errore.

Le principali funzioni di attivazione usate per le reti neurali dunque sono:

- Sigmoide, di equazione $\frac{1}{1-e^{-x}}$ (fig.23);
- Tangente iperbolica, di equazione $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ (fig.24);
- Rectified Linear Unit, di equazione $\max(0, x)$ (fig.25);

Il codominio delle funzioni è [0, 1] per la sigmoide, [-1, 1] per la tangente iperbolica e [0, +inf] per la ReLU (Rectified Linear Unit).

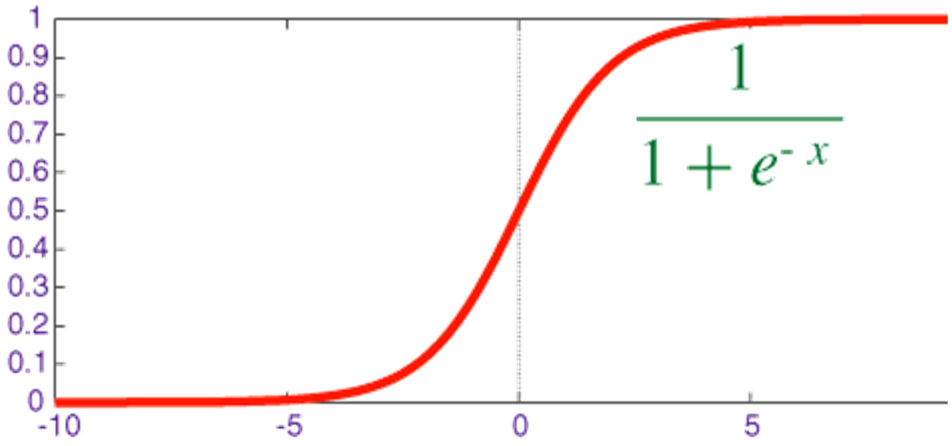


Figura 23

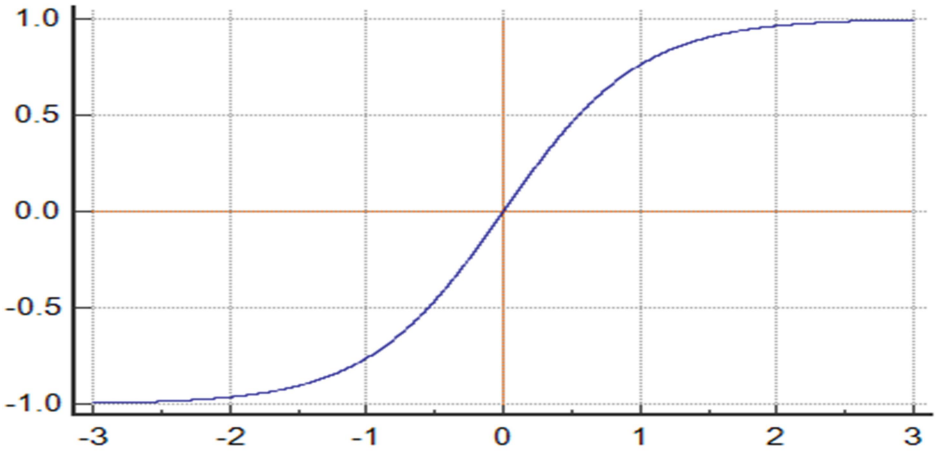


Figura 24

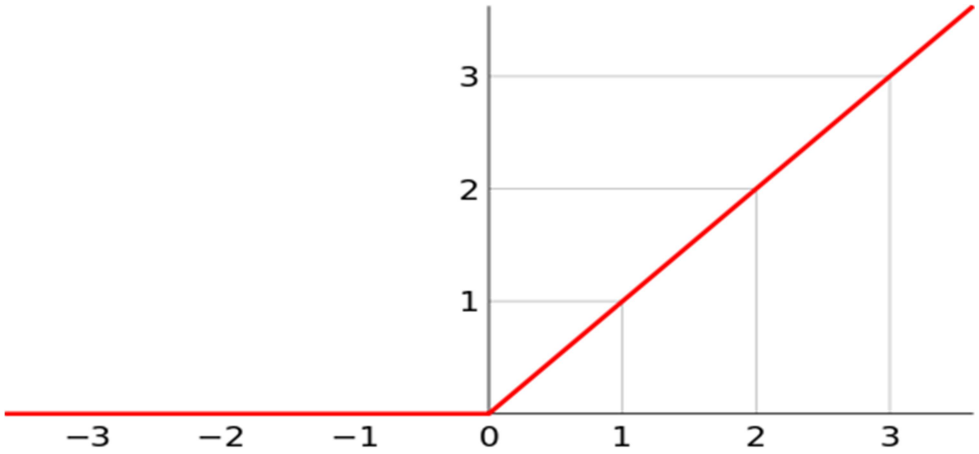


Figura 25

Nel caso della sigmoide in particolare, la formula di derivazione è, data $s(x)$ sigmoide ed $s'(x)$ la derivata del primo ordine, $s'(x) = s(x) * (1 - s(x))$ e quindi ciò riduce il tempo di calcolo non di poco nel caso di discesa del gradiente (essendo la derivata esprimibile come funzione della sigmoide stessa).

2.3 Metodi di discesa del gradiente

In questo paragrafo verranno introdotte le principali metodologie di discesa del gradiente, dove le differenze possono variare anche sensibilmente nelle modalità di aggiornamento. Verranno spiegate in dettaglio le tipologie di discesa del gradiente usate nella tesi con vantaggi e svantaggi documentati da letteratura.

I metodi discussi saranno tre:

- Adam (Adaptive Momentum)[18];
- Nadam (Nesterov Adaptive Momentum)[19];
- RMSProp [20].

Tutti i metodi trattati nascono, in linea di principio, come estensione e miglioramento di un metodo chiamato Adagrad[21], il primo ad utilizzare non più un learning rate fisso e preselezionato ma uno adattivo, calcolato durante il training in base alla magnitudo dei

valori precedenti di gradiente (al passare delle epoche quindi diminuisce sempre di più). In figura 26 la formula generale di aggiornamento dei pesi per Adagrad.

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt[2]{G_t + \varepsilon}} * g_t$$

Figura 26

In questo caso θ_t rappresenta i parametri della rete al tempo t, μ è il learning rate iniziale, g_t rappresenta il gradiente all'istante attuale, ε rappresenta una costante per evitare problemi in caso di valore pari a zero al denominatore e la matrice G_t rappresenta, a sua volta, tutti i passati gradienti g_t^2, per ogni istante t di addestramento effettuato (più epoche passano, più la somma dei g_t^2 da $t = 0$ a $t = t'$, con t' l'istante attuale, aumenta).

In tutti i metodi, il gradiente g_t è inteso come equivalente alla formula in figura 27 e ci porremmo sempre in ottica di minimizzazione dell'errore (i metodi che saranno citati valgono equivalentemente in caso di massimizzazione).

$$\nabla_w \sum_{1}^{m} L_m(w)$$

Figura 27

2.3.1. RMSProp

E' stato uno dei primi metodi di discesa del gradiente proposto, in origine, come alternativa al principale problema di Adagrad, cioè il suo graduale ridurre il learning rate al passare delle epoche di addestramento, fino al rendere quasi nullo il contributo dell'errore, bloccando di fatto l'apprendimento[22].

L'eliminazione di questo problema avviene eseguendo una media, a decadimento esponenziale, del quadrato del valore attuale di gradiente e dei valori passati di gradiente

(anche se in questo caso ne viene eseguita una media durante l'esecuzione, non si tengono come invece per Adagrad tutti i singoli valori del quadrato del gradiente).

Ecco come si aggiornano i parametri con RMSProp all'istante t di aggiornamento:

1. Si calcola la media corrente dei gradienti, secondo la formula $E[g^2]_t = \gamma * E[g^2]_{t-1} + (1-\gamma)\, g_t^2$, dove in questo caso γ rappresenta un ulteriore parametro di learning per la media dei valori quadratici di gradiente;
2. Si aggiornano i parametri della rete secondo la formula $\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt[2]{E[g^2]_t + \varepsilon}} * g_t$, dove si usa quindi la stima di valori storici medi di gradiente, così da avere una riduzione dinamica del learning rate al passare delle epoche meno drastico e che non blocchi l'apprendimento arrivati ad un certo punto.

2.3.2. Adam

Questo metodo è una specie di miglioramento di RMSProp, in quanto oltre a considerare il contributo della stima della media del quadrato dei gradienti, si calcola anche ciò che viene chiamato momento, ovvero una stima della media di valori del gradiente. Il momento ha una funzione di correzione del gradiente, in quanto ne permette l'aumento se non cambia segno per alcune epoche mentre ha l'effetto opposto se tende ad oscillare fra valori positivi e negativi. L'effetto correttivo del momento sta quindi nel velocizzare la discesa del gradiente in regioni piatte (o comunque che vanno nella stessa direzione) della funzione di errore e di rallentare la discesa invece in regioni dove l'errore ha un profilo "appuntito", cioè con cambi di direzione improvvisi al minimo cambio di input (si evita così un'oscillazione intorno ad un minimo).

L'aggiornamento dei parametri con Adam quindi è effettuata secondo questi passi:

1) Si calcola sia la stima della media dei gradienti quadratici (varianza non centrata, v_t), sia la stima della media dei gradienti del primo ordine (momento, m_t). Il calcolo dei due avviene secondo questo due formule:
   a. $m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$;
   b. $v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$;
2) Si correggono i due valori in quanto essendo inizializzati a zero, senza il fattore correttivo non si discosterebbero da zero al passare delle epoche. Ecco le formule con i valori corretti di momento e varianza non centrata:
   a. $\widehat{m}_t = \frac{m_t}{(1-\beta_1^t)}$;
   b. $\hat{v}_t = \frac{v_t}{(1-\beta_2^t)}$;
3) Come ultimo passo, otteniamo l'aggiornamento vero e proprio dei parametri di una generica rete, secondo la seguente formula:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt[2]{\hat{v}_t + \varepsilon}} * \widehat{m}_t$$

I termini β_1 e β_2 servono come fattori di apprendimento per la stima della media e della varianza, oltre al classico learning rate μ sempre presente.

2.3.3. Nadam

L'ultima metodologia di discesa del gradiente trattata è una variante di Adam, in cui invece di calcolarlo in maniera "classica" si calcola secondo una variante proposta da Nesterov[23], con performance migliori dei metodi di discesa del gradiente tradizionali ed adattabile ad una sorta di metodo di calcolo del momento migliorato.

In breve sostanza il metodo migliora il calcolo del momento applicando la correzione dei pesi in "avanti", cioè prima del calcolo dell'errore, poi al passo successivo viene calcolato l'errore effettivo.

La differenza nei passi fra Adam e Nadam si può riassumere nel passo finale:

1) Come ultimo passo, otteniamo l'aggiornamento vero e proprio dei parametri della rete sotto considerazione secondo la seguente formula modificata:

$$\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt[2]{\hat{v}_t + \varepsilon}} * (\beta_1 * \hat{m}_t + \frac{(1-\beta_1) * g_t}{1-\beta_1^t}).$$

Si può notare l'aggiunta di un termine g_t, che serve appunto per accelerare la discesa del gradiente (Nesterov Accelerated Gradient, NAG) secondo Nesterov, applicandolo un passo temporale in anticipo.

2.4 Tecniche di affinamento delle reti

Per tecniche di affinamento delle reti si intendono tutta quella serie di strumenti che servono generalmente a migliorare, in qualche modo, le performance delle reti neurali, affrontando specifici problemi. Fra le tecniche si annoverano soprattutto quelle di inizializzazione dei pesi di una rete neurale ed i layer avanzati, così da migliorarne la predittività e minimizzarne l'errore.

2.4.1. Inizializzazione dei pesi

Parte importante dell'apprendimento per le reti neurali è il valore iniziale dei pesi delle connessioni, in quanto pesi iniziali adeguati possono ridurre drasticamente il tempo necessario alla convergenza della rete. Come esempio, basti pensare che impostando i valori di tutti i pesi a zero, una rete neurale diventa identica ad un modello lineare, in quanto tutti i neuroni in tutti gli hidden layer assumono gli stessi valori di peso e dunque la rete neurale perde gran parte della sua utilità e capacità di classificazione.

Altri fenomeni che l'inizializzazione dei pesi cerca di alleviare prendono rispettivamente il nome di esplosione[25] e scomparsa del gradiente[26], ovvero i fenomeni per cui il training ad un certo punto diventa instabile oppure si ferma senza migliorare più, cercando di mitigare questi effetti al passare delle epoche di training.

2.4.1.1. He inizialization

Questa tipologia di inizializzazione di pesi nelle reti neurali è stata pensata nel caso di utilizzo della funzione di attivazione ReLU, che permette di mitigare la scomparsa del gradiente grazie al suo peculiare output, dove non sono ammessi valori negativi e non viene appiattito in corrispondenza di valori di ingresso molto alti (come invece avviene nel caso di sigmoide o tangente iperbolica, caratterizzate da scarsa sensibilità alla variazione degli input negli estremi dell'intervallo). Insieme alla ReLU quindi l'inizializzazione mitiga questi problemi imponendo il campionamento dei pesi da una distribuzione gaussiana di media zero e deviazione standard pari a $\sqrt[2]{\frac{2}{nl}}$, dove nl è il numero di neuroni nel livello l-esimo.

2.4.2. Layer avanzati

Fra i layer per reti neurali che si possono considerare avanzati o comunque molto importanti ed utili per migliorarne le qualità di predizione, possiamo annoverare la Batch Normalization e la regolarizzazione.

2.4.2.1. Batch Normalization layer

La tecnica chiamata batch normalization è stata in origine pensata per ridurre il fenomeno noto come internal covariate shift (ics, [28]), ovvero un fenomeno per cui l'input di ogni livello di una rete, in particolare se a tanti livelli, tende a variare in base ai parametri del layer precedente, provocando un'indesiderata variazione della distribuzione degli input e che dunque impone una scelta attenta dei vari parametri di training, minandone la capacità e velocità di convergenza. Inoltre, questo fenomeno tende a portare a saturazione le uscite dei neuroni di livelli differenti, provocando nel caso migliore un rallentamento se non il blocco definitivo dell'addestramento della rete nel suo complesso.

Studi successivi in realtà, hanno rifiutato questa ipotesi, seppur confermando le potenzialità ed il successo di questa tecnica nel velocizzare e portare a migliori performance di training. Si attribuisce infatti a questo layer la capacità di migliorare quella che viene chiamata la

"Lipschitzness"[29] della funzione di loss, rendendo di fatto l'intero training più stabile e graduale, permettendo learning rate per le funzioni di discesa del gradiente più elevati, ottenendo una maggiore velocità di convergenza e migliorando le performance generali di accuratezza e riduzione dell'errore [30]. In figura 28 si può vedere la proprietà di una funzione per essere considerata Lipschitz continua, dove L rappresenta un numero reale costante.

$$|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\|$$

Figura 28

L'effetto misurabile di una funzione che ha questa proprietà è in sostanza un "panorama" più smussato, nel caso di una funzione di loss questo significa avere una funzione caratterizzata da meno picchi e meno regioni piatte, garantendo a ragionevolmente piccole variazione degli input piccoli cambiamenti nella funzione di errore (fig.29). In generale ciò permette di avere maggiori garanzie di accuratezza anche nell'approssimazione del gradiente in quanto, nonostante che il calcolo dello stesso su un mini-batch sia solo un'approssimazione di quello reale, le differenze fra lo stimato e il reale diventano più piccole(fig.30), a vantaggio della qualità di predizione (non ci saranno differenze troppo elevate fra gradiente stimato ed effettivo).

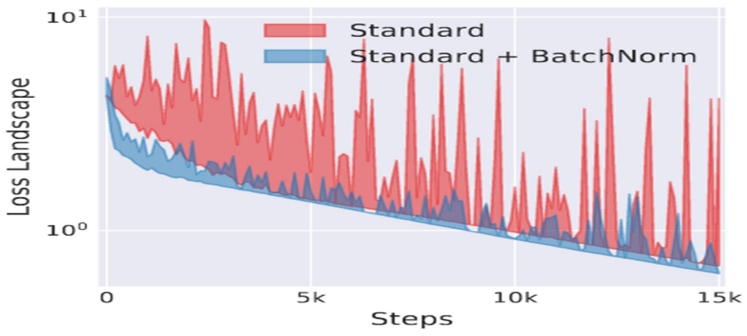


Figura 29(tratto da arXiv:1805.11604)

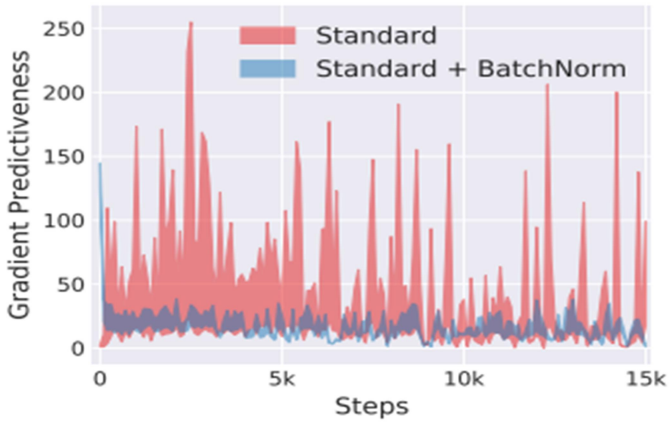


Figura 30(tratto da arXiv:1805.11604)

Il layer di Batch Normalization dunque si applica effettuando un passo di normalizzazione degli input, prima di sottoporli alla trasformazione tramite funzione di attivazione.

Dunque, dati m vettori $\boldsymbol{x}$ a d dimensioni ($\boldsymbol{x_i} = (x^1, x^2, , \dots, x^d)$), si applica ad ognuno di questi vettori il calcolo di media e varianza per ognuna delle d dimensioni, in base alle m osservazioni di input per quel particolare mini-batch (ovvero sottopongo m vettori di input alla rete e calcolo i parametri di media e varianza, per ogni dimensione, di quello specifico mini-batch).

Il calcolo dunque, eseguito separatamente e da capo per ogni mini-batch di m elementi è il seguente:

- Si calcola la media per ogni dimensione d (verrà omesso per semplicità l'apice corrispondente alla dimensione di cui si calcola la media), cioè $\mu_b = \frac{1}{m} * \sum_1^m x_i$;
- Si calcola la varianza per ogni dimensione, $\sigma_b^2 = \frac{1}{m} * \sum_1^m (x_i - \mu_b)^2$;
- Avviene la normalizzazione dei valori in quella dimensione, secondo la formula $\hat{x} = x_i - \frac{\mu_b}{\sqrt[2]{\sigma_b^2 + \varepsilon}}$;

- Infine se ne calcola il valore normalizzato, usando per il calcolo due parametri addizionali da apprendere durante il training (β, γ), fondamentali per ricostruire gli input originali. La formula finale per il calcolo è dunque $y = \gamma * \hat{x} + \beta$.

Un altro dei vantaggi dell'algoritmo è che l'apprendimento dei due parametri (β, γ) avviene durante tutto l'addestramento, non solo sul singolo mini-batch, aumentando anche la capacità di generalizzazione del modello grazie a questi parametri che fanno da "storia" per ogni mini-batch successivo, fornendo quindi una visione d'insieme della distribuzione degli input fra mini-batch diversi.

2.4.2.2. Layer di regolarizzazione

La regolarizzazione è una tecnica che, in generale, si occupa di introdurre una penalità nell'utilizzo di pesi elevati per le connessioni, in modo da indurre nella rete una maggiore capacità di generalizzazione. Fra le tecniche più usate troviamo la regolarizzazione tramite norma L1 o norma L2, che rappresentano le funzioni per calcolare i contributi dei pesi.

Il termine di regolarizzazione, per i due casi, si calcola in questo modo:

$$R_{L_1}(W) = \sum_i \sum_j |W_{i,j}|$$

$$R_{L_2}(W) = \sum_i \sum_j W_{i,j}^2$$

Nelle formule sopra, i termini i e j rappresentano i pesi della connessione fra i neuroni i-esimi e j-esimi, appartenenti a livelli adiacenti. In generale i pesi da introdurre come regolarizzatori nella funzione di loss possono appartenere a specifici layer e possono anche essere presi i pesi di più layer contemporaneamente, in base a ciò che si vuole ottenere come effetto di regolazione.

La formula finale quindi di una generica funzione di errore con inseriti il termine o termini di regolazione diventa:

$$L(x) = L(x) + \lambda * R_{L_p}(W)$$

Il termine λ rappresenta il contributo (in genere è molto piccolo) di questo termine nell'errore totale e L_p sta ad indicare una generica funzione di regolarizzazione, che può essere una fra le due indicate in precedenza. L'effetto che si cerca di indurre è che a parità di accuratezza e precisione di predizione, il modello abbia anche il minimo valore di pesi, così da evitare overfitting mantenendone inalterata la capacità di classificazione e predizione.

2.4.3. Early stop

Elencate le tecniche algoritmiche, ora avverrà una breve spiegazione di una tecnica meno giustificata a livello formale ma molto utilizzata e di provata efficacia, l'early stop[31].

Questa tecnica è usata prevalentemente per la prevenzione dell'overfitting ed è imposta scegliendo il numero di epoche per il quale si è disposti ad aspettare prima di fermare il training. In generale, deciso un numero di epoche che si concede alla rete per imparare a classificare correttamente l'input, si attende il termine dell'addestramento mentre con l'early stop viene deciso un numero massimo di iterazioni che si concede al modello per non diminuire l'errore fra epoche diverse, chiamato pazienza.

Una volta deciso questo parametro, se per un numero di iterazioni n maggiore a quello stabilito in $pazienza$ l'errore non diminuisce, viene arrestato il training, per non rischiare di ottenere una rete caratterizzata da overfitting o comunque che non migliora più le proprie capacità di predizione (l'errore aumenta invece di decrescere o in maniera equivalente l'accuratezza diminuisce invece di aumentare). In figura 31 viene mostrato un esempio dell'effetto che si vorrebbe avesse l'early stop sulle performance di predizione del modello, dove si raggiunge il miglior compromesso fra accuratezza sul training set e prevenzione di overfitting.

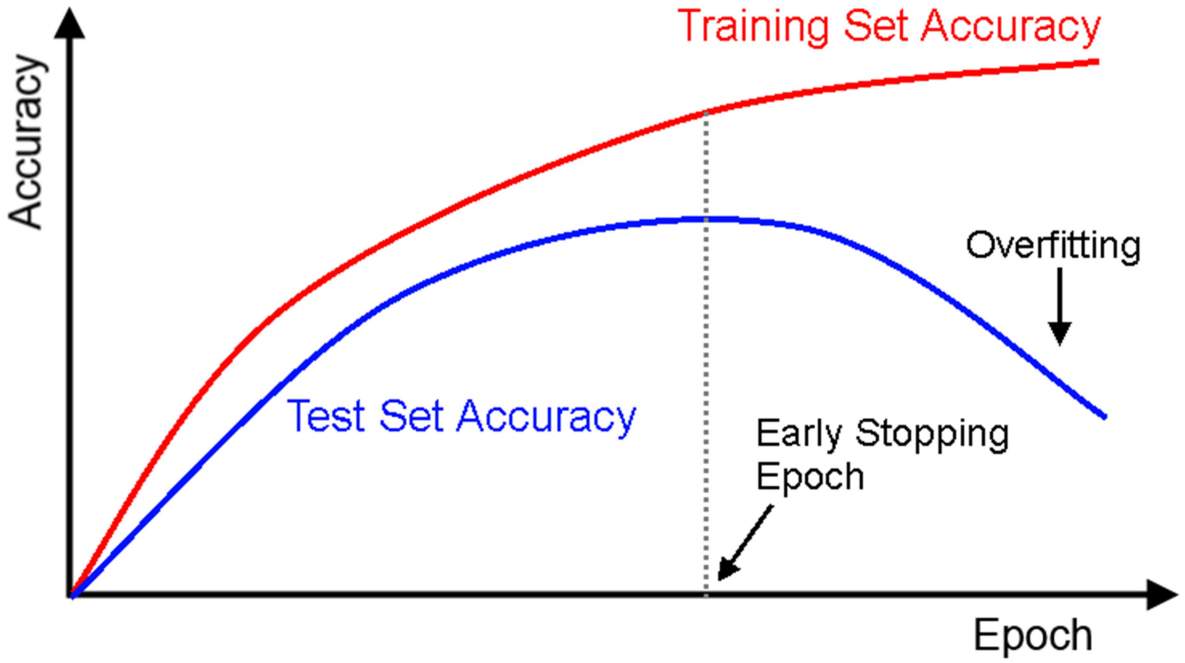


Figura 31

Come si può notare, il giusto parametro di early stop permette al modello di mantenere le proprie capacità di generalizzazione senza cadere nell'overfitting. Ciò implica anche l'uso di vari test per decidere e trovare il parametro di stop più adeguato, oltre che l'uso di tecniche per fare le suddette prove su test set diversi costruiti casualmente.

Un metodo adeguato per scegliere questo parametro in modo corretto è in coppia a qualche tecnica di validazione del modello, come l'holdout e il k-fold (stratificato o meno), così da avere una buona misura dell'errore senza però sopravvalutarlo o sottovalutarlo a causa del rumore o della particolare distribuzione delle classi nel modello di validazione scelto.

2.5 Funzioni d'errore e classificazione multi-classe

Nell'ultima parte di questo capitolo verranno introdotte le funzioni d'errore, le funzioni di attivazione più appropriate ed i metodi di codifica da dover adottare per la classificazione multi-classe con le reti neurali.

La funzione di errore usata nell'implementazione del modello di predizione sviluppato sarà la cross entropy, che serve a misurare la differenza fra due distribuzione di probabilità. La sua formula è la seguente:

$$CE = -\sum_{x} p(x)\ \log q(x)$$

Nella formula $p(x)$ rappresenta la distribuzione di probabilità reale di un'istanza e $q(x)$ invece quella calcolata dalla rete neurale, dato il vettore x come input. Dobbiamo immaginare, nel caso della classificazione, come distribuzione di probabilità reale un vettore binario con un "bit" di codifica per ogni classe possibile (dunque $p(x)$ sarà composto da bit), in cui tutti i valori saranno a zero a parte per il bit della classe effettiva dell'istanza specifica x che sarà "1", con un valore finale di errore dato dalla seguente funzione di loss (per la specifica classe k, tutte le altre non sono considerate):

$$L_k = -\log P\big(Y = k,\ x\big) = -\log\left(\frac{e^{f_k(x)}}{\sum_j e^{f_j(x)}}\right)$$

Nella formula si intende L_k come la loss della specifica classe che si doveva predire, per $-logP(Y = k, x)$ si intende il logaritmo negativo calcolato per la classe k e l'input x ed infine come valore specifico di probabilità si prende quello calcolato dalla funzione softmax (è il valore di probabilità calcolato dalla rete neurale, cioè $q(x)$ nell'equazione generale della cross entropy).

La funzione di attivazione softmax calcola le probabilità normalizzate delle singole classi, partendo da quelle non normalizzate fornite dalla rete neurale come score $f_j(x)$, secondo la formula $P(Y = k, x) = \frac{e^{f_k(x)}}{\sum_j e^{f_j(x)}}$, in cui il denominatore è la somma delle probabilità non normalizzate di tutte le classi ed $f_k(x)$ è invece il contributo di probabilità della singola classe k di cui vogliamo il valore normalizzato.

3 Analytics nel calcio: SiCS e l'Indice di Pericolosità

Nel calcio, come in tanti altri settori per motivi diversi, sta prendendo sempre più piede l'utilizzo delle statistiche e delle analisi come supporto alle decisioni tecnico-tattiche durante le partite[32]. Questa tendenza è nata, dal punto di vista tecnologico, sia grazie alla sempre maggiore disponibilità di informazioni (i già citati Big Data) e sia grazie alla maggiore disponibilità di strumenti potenti e performanti, capaci di elaborare questa enorme mole di dati ed informazioni in brevissimo tempo.

Un caso noto ed emblematico di utilizzo di strumenti per analisi e raccolta di informazioni è l'NBA (National Basketball Association), dove a partire dal 2009 sono state introdotte telecamere per la rilevazione dei movimenti dei giocatori e della palla ad una velocità di 25 frame al secondo, con il conseguente esorbitante incremento del numero di dati ed informazioni disponibili per tutte le squadre del campionato.

Questo ha portato ad una sostanziale rivoluzione di questo gioco negli anni, con conseguenze su settori come l'osservazione (scouting) ed il reclutamento (recruiting), dove adesso ogni squadra del campionato è dotata di figure come gli analisti di dati a supporto di allenatore, staff tecnico e preparatori atletici, per la prevenzione di infortuni, studio di nuove tecniche o schemi, analisi delle giocate più efficaci per segnare punti e vincere le partite[33].

Detto ciò, ora ci focalizzeremo principalmente sull'introduzione di queste tecniche di analisi nel mondo del calcio, tramite una panoramica di ciò che è già stato sviluppato e che è oggetto di tesi mentre successivamente si farà una breve panoramica di altre tecniche sviluppate e di possibile interesse in questa direzione.

3.1. SiCS ed indice IPO

Nel mondo del calcio, seppur ancora mancano statistiche avanzate e massive come nel mondo del basket, si cominciano a muovere i primi passi verso un'evoluzione tecnologica di questo sport. A tal proposito esiste una società italiana, di nome SiCS ([34], logo in figura 32), che a

partire dal 2014 ha ideato e sviluppato un indice chiamato IPO, ovvero Indice di Pericolosità Offensiva.

Figura 32

L'indice in questione in realtà è stato sviluppato da questa società in collaborazione con esperti del settore, sia allenatori che match analyst, nelle figure in particolare di Antonio Gagliardi, Maurizio Viscidi e Marco Scarpa, in cui il primo è capo del settore Match Analysis della FIGC (Federazione Italiana Giuoco Calcio), il secondo è coordinatore della nazionali giovanili maschili di calcio ed il terzo è nello staff tecnico dell'attuale commissario tecnico Roberto Mancini[35].

Tornando all'IPO, questo indice ha come scopo principale quello di dare una misura di pericolosità di una qualunque squadra, aldilà dei soli goal segnati, fornendo un'indicazione sintetica di occasioni create o comunque di quanto gioco offensivo produce in media quella specifica formazione.

La formula matematica con cui l'indice può essere descritto è la seguente:

$$IPO = \sum_i peso_i * eventi_i$$

Come si può notare, la formula non è altro che la somma del numero di occorrenze in una partita di ogni i-esimo evento (dove questi eventi sono stati definiti a priori dagli esperti creatori dell'indice) pesato per il relativo valore di importanza, sempre prestabilito e diverso per ognuno degli eventi. Come si può intuire, l'IPO non è un valore "magico" che indica senza margine di errore l'esito delle partite ma è un utile strumento però per capire le

potenzialità e dare un'indicazione di quantità di gioco di una squadra sul lungo termine, fornendo un'indicazione piuttosto affidabile di ciò che ci si può aspettare sia l'esito di un campionato. Integrato con altre metodologie dunque, l'IPO può essere considerato un utile se non fondamentale strumento di supporto, seppur abbia ampi spazi per essere esteso e migliorato, come per quanto riguarda la parte di calcolo, selezione e peso di ogni singolo evento, scopo tra l'altro di questa tesi.

Il punto di vista che ha scaturito la creazione di questo indice è stata l'espressa volontà, da parte dei suoi ideatori, di stimolare lo sviluppo di strumenti non solo meramente frutto di analisi statistiche e matematiche ma frutto di conoscenze di persone, in primis, esperte del settore, motivo per cui è stato sviluppato con eventi stabiliti e peso stabilito da persone formate a livello sportivo più che matematico[36].

Grazie all'indice si è empiricamente stabilito che in un intervallo di IPO compreso fra i 20 e 45 punti si segna, con una media di 30 o 33 punti circa[37]. La predizione può cambiare sensibilmente, anche se di rado, a parità di punteggio IPO ma anche qui bisogna avere conoscenza del dominio (calcio) per comprendere il fenomeno.

Come esempio basta pensare che ad una squadra con degli ottimi difensori può capitare di subire più azioni offensive contro senza subire goal, ma ciò può dipendere anche dalla qualità dei difensori, dalla buona partita del portiere e da tanti altri fattori, pur sempre restando in valido, in media, il punteggio e l'indice sul lungo termine di un campionato.

3.2. Ulteriori tecniche di analisi

Altra tecnica per la misurazione delle performance di una squadra è anche il goal expectation (ExpG [38]), che tiene conto di vari fattori per stabilire quanti goal ci si aspetti che una squadra segni in una determinata partita, in base alla posizione da cui viene effettuato il tiro , dalla media goal del giocatore specifico, dal tasso di conversone goal del giocatore, dal posizionamento delle difese avversarie.

Questo parametro, a differenza dell'IPO, è una metrica specifica per i tiri in porta e di conseguenza tiene meno conto degli aspetti tecnici della partita, del suo svolgimento e della qualità della squadra in se per se, in quanto è stata sviluppata più come tecnica statistica che come strumento di supporto sviluppato da esperti del settore.

4. Reti Neurali e predizione di partite

In questo ultimo capitolo avverrà una fase iniziale di spiegazione dell'intuizione dietro al lavoro di tesi, successivamente avverrà un'approfondita spiegazione ed illustrazione delle principali tecnologie usate, con un'enfasi finale sull'intero lavoro di codifica degli input, pre-processing dei dati, tipi di rete neurale implementate e grafici dei risultati sperimentali ottenuti.

4.1 Indice di pericolosità e reti neurali

Il primo punto del lavoro dunque è stato l'immaginare come poter estendere ed integrare l'indice di pericolosità offensiva con tecnologie come le reti neurali.

Dietro il lavoro di estensione c'è l'idea di come i dati, forniti da SiCS (figura 33), possano essere utilizzati ed interpretati non solo da un punto di vista di indice, con degli indicatori prefissati e scelti a priori da esperti, ma anche da strumenti automatici di predizione e classificazione. Le reti neurali, che sono fra gli strumenti che stanno prendendo maggiormente piede grazie soprattutto al Deep Learning e all'eccezionale successo che hanno avuto in vari compiti ad elevata complessità (nel capitolo "Machine Learning e reti neurali" ne sono stati presentati alcuni), si prestano molto bene alla classificazione e predizione, motivo per il quale sono state scelte per la realizzazione di questa idea.

L'intuizione alla base dell'intero lavoro è che l'IPO può essere immaginato come un indice per predire le partite di calcio, dunque riuscendo nel dare indicazione dell'esito di una partita in termini di classificazione dove le classi da predire sono "1" per al vittoria della squadra in casa, "0" per il pareggio e "2" per la sconfitta.

Ciò in cui può essere fondamentale una rete neurale è nell'integrare un meccanismo di calcolo e stima, basata sull'apprendimento, di quelli che sono i pesi da attribuire ad ogni evento, in modo da rendere questo indice non soggetto al solo punto di vista degli esperti ma anche all'esperienza, intesa qui come apprendimento supervisionato della rete così da renderla sempre più precisa nella previsione dei match.

L'indice di pericolosità infatti viene calcolato, come già specificato, secondo la formula $IPO = \sum_i peso_i * eventi_i$, dove la rete neurale può intervenire nell'estendere questo meccanismo sostituendosi nelle due principali componenti, ovvero:

- Scelta del valore dei pesi, intesa come dare la giusta importanza agli eventi di una partita;
- Scelta degli eventi rilevanti durante il match.

Come si intuisce dunque, la rete neurale può integrare ed estendere il punto di vista degli esperti pur essendo totalmente all'oscuro di ciò che rappresentano tali dati, focalizzando l'attenzione sulla scoperta di schemi, pattern, nuovi fattori che sono determinanti nella predizione delle partite ma da un punto di vista nuovo.

A livello più formale, essendo una rete neurale un ambiente universale di approssimazione, fornisce gli strumenti per poter stimare qualunque tipo di funzione che sia definita in uno spazio n-dimensionale finito, motivo per il quale è potenzialmente in grado non solo di riprodurre le capacità di predizione dell'IPO con una buona precisione (almeno in linea di principio) ma può addirittura trovare la funzione che discrimina gli eventi che portano alla vittoria piuttosto che quelli che determinano un pareggio od una sconfitta, aldilà di ciò che in realtà potrebbe indicare l'IPO stesso[39].

Per queste ragioni sono state sviluppate due reti neurali distinte che, seppur a livello architetturale puro sono molto simili, lavorano una con lo scopo di riprodurre l'IPO originale, usando cioè lo stesso input che forma l'indice creato dagli esperti per cercare di effettuare le predizioni mentre l'altra usando tutti i dati raccolti dai database forniti da **SiCS** (**http://www.sics.it/**, logo in figura 33) predice le partite creando un meccanismo dunque ex-novo di classificazione e predizione.

Di seguito verranno presentati e spiegati gli strumenti utilizzati per realizzare le due reti.

Figura 33

4.2. Strumenti e tecnologie implementative

In questo paragrafo avverrà una rassegna e spiegazione approfondita delle tecniche e linguaggi utilizzati per la realizzazione delle reti neurali oggetto di tesi.

Punto in comune degli strumenti, a parte per ciò che riguarda la gestione del database, è l'utilizzo di Python (**https://www.python.org/**) come linguaggio di programmazione, in particolare la versione 3.6 dell'interprete e come ambiente di sviluppo è stato utilizzato PyCharm Community Edition versione 2018.2.2, sviluppato dalla software house JetBrains (**https://www.jetbrains.com/pycharm/**).

Il motivo principale di utilizzo di questo linguaggio è l'enorme disponibilità di moduli per il machine learning, costantemente in aggiornamento come il Python stesso, soprattutto in ambito di reti neurali, dove infatti sono presenti pacchetti sia per il calcolo efficiente di grossi moli di dati (anche multidimensionali), sia di supporto alle statistiche.

Un altro punto a favore del Python è la presenza di molti algoritmi e tecniche propri del Data Mining, cosa che permette un rapido e facile sviluppo di applicazioni scientifiche.

I principali moduli e strumenti usati in Python per la realizzazione delle reti sono Tensorflow (**https://www.tensorflow.org/**) e Keras (**https://keras.io/**), fra i principali strumenti usati nel settore delle reti neurali in generale e per il Deep Learning. Di seguito avverrà un'approfondita spiegazione di questi due strumenti, in quanto fondamentali nello sviluppo e testing dell'intera applicazione.

4.2.1. Tensorflow

Tensorflow, già dal nome, fa capire parte delle sue funzionalità, che sono nello specifico il calcolo di tensori secondo un modello di programmazione a flusso di dati. Non è scopo della tesi approfondire il calcolo tensoriale, ma quanto spiegare l'utilità di questa libreria per il supporto al machine learning ed in particolare per lo sviluppo delle reti neurali.

In generale i dati in input alle reti neurali possono essere visti come vettori ad n dimensioni, basti pensare al caso di immagini, piuttosto che vettori piatti come nel caso della rappresentazione scelta in questa tesi per le partite di calcio, dove inoltre si hanno tanti individui singoli di una popolazione utili ad individuare le caratteristiche principali che determinano il mapping fra input e classi di uscita.

La varietà dell'input di una rete porta dunque con se la necessità di strumenti per il calcolo di vettori complessi, o tensori, efficienti ed adatti alla computazione su calcolatore, vista anche l'enormità che caratterizza spesso i dati da dare in "pasto" alla rete come insiemi di addestramento e test delle performance (figura 34).

$$\begin{bmatrix} x_1 & x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 \end{bmatrix}$$

Figura 34

Come si può notare, ogni "individuo" in input è una colonna della matrice sopra riportata e dunque più individui e dimensioni abbiamo, più le righe e colonne diventano un'enormità, giustificando lo sviluppo di strumenti per il calcolo efficiente e veloci di questi insiemi di dati. Infatti, dato un vettore n dimensionale ed r individui, la matrice dei soli input diventerebbe di $n * r$ elementi, tenendo infine conto delle connessioni (che rappresentano dunque un'altra matrice, in cui il numero di elementi dipende dai tipi di layer e dal numero

di neuroni connessi fra i diversi livelli) otteniamo in totale un numero di moltiplicazioni pari a $n * r * c$, con c numero di connessioni (una per neurone nel layer successivo e per ogni neurone di quello precedente, in caso di layer completamente connessi), che può diventare senza problemi dell'ordine del milioni di parametri solo al primo livello.

Le tecniche dunque su cui si basa Tensorflow per la risoluzione efficiente e veloce di problemi su grandi moli di dati sono il dataflow programming ed il differentiable programming.

4.2.1.1. Dataflow Programming

Questo paradigma di programmazione fu per primo introdotto nel 1960 da un professore del MIT e dai suoi studenti, con lo principale di introdurre alcuni concetti della programmazione funzionale a supporto delle applicazioni con un'attenzione particolare verso il calcolo numerico[40]. Ciò è chiaramente a grande vantaggio delle applicazioni nel campo del machine learning, ottenuto grazie alla visione di un programma come un grafo diretto in cui le varie operazioni vengono eseguite secondo l'ordine di elaborazione del flusso di dati (figura 35).

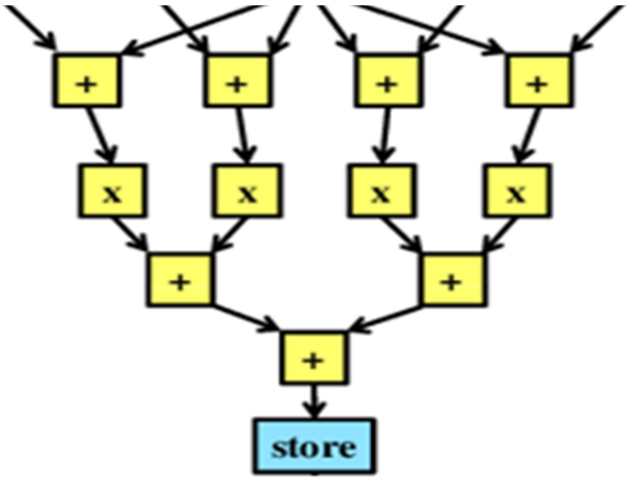


Figura 35

Come si può intuire dunque, i singoli "blocchi" rappresentano le operazioni e le frecce direzionali i flussi di dati e da quali altri blocchi provengono, dove dunque il programma non è altro che una serie di operazioni dipendenti solo dalla disponibilità degli input da cui dipendono.

Questo tipo di programmazione dunque permette un parallelismo maggiore ed implicito nelle operazioni, in quanto non ci sono nozioni di stato da dover memorizzare e dunque nemmeno meccanismi di sincronizzazione necessari da implementare, a chiaro favore di velocità di esecuzione (il parallelismo di per se è sempre un vantaggio a livello di tempistiche). Tutto ciò permette quindi ai blocchi di essere dipendenti solo dai flussi di dati (in questo caso i dati sono tensori), da cui l'origine del nome della libreria (TensorFlow appunto).

4.2.1.2. Differentiable Programming

E' una tipologia di programmazione che si concentra sul calcolo differenziale eseguito durante l'esecuzione, solitamente eseguita secondo la tecnica della differenziazione automatica[41].

La differenziazione automatica, di per se, è una tecnica che è stata sviluppata nel 1964[42] ed è un metodo di calcolo efficiente delle derivate, di qualunque ordine, sfruttando la regola della composizione di derivate. Il grande vantaggio nell'uso di questa tecnica sta nella sua capacità di essere precisa e veloce nel calcolo esatto delle derivate, grazie anche alla visione delle operazioni e dunque delle funzioni come un grafo in cui si intrecciano gli elementi di input fino a comporre l'output finale (figura 36).

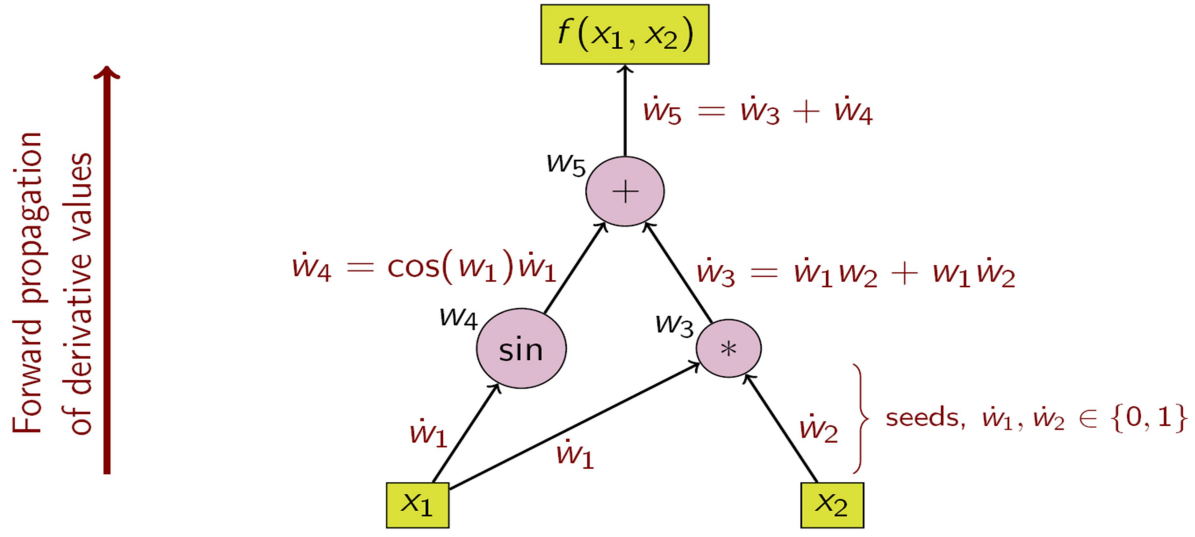


Figura 36 [43]

In figura viene mostrato un esempio di differenziazione automatica e come si può notare il calcolo delle derivate avviene già all'inizio partendo da x_1 ed x_2, si calcolano subito le derivate rispetto a questi due valori (ottenendo zero ed uno rispettivamente in questo caso) e da qui in poi, applicando la regola della composizione delle derivate, si usano questi valori intermedi per il calcolo delle stesse nei nodi successivi, garantendo precisione e velocità di calcolo (essendo svolto durante l'esecuzione stessa). In realtà questa è una delle due modalità di calcolo, in quanto se invece il nostro obiettivo è il contrario, cioè partire dalla funzione finale ed arrivare ai contributi di derivata dati dai valori "originari" x_1 ed x_2, , il procedimento è analogo anche se in genere computazionalmente più lungo (si deve eseguire il passo "in avanti", memorizzando i risultati come sopra, in seguito si deve procedere all'indietro per calcolare i singoli contributi, tenendo i valori parziali), mostrato in figura 37.

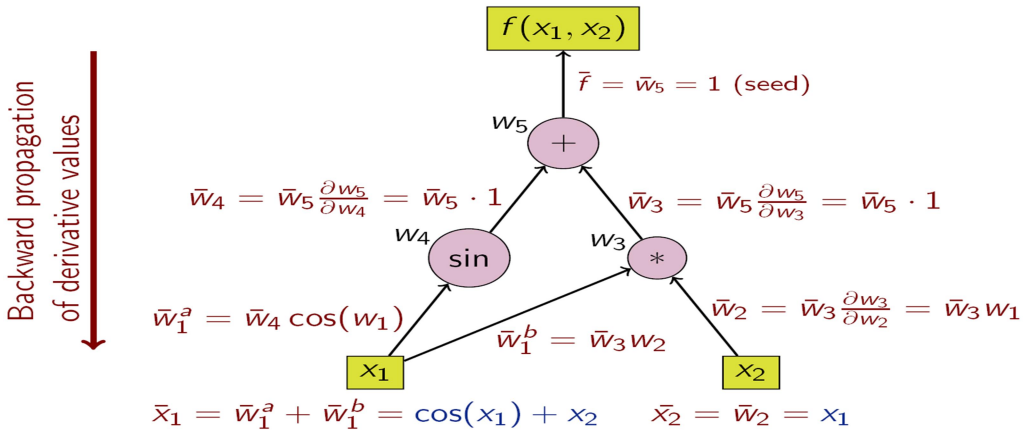


Figura 37

Si può subito notare la somiglianza fra la figura 37 e la 18 (Capitolo 2), infatti il procedimento mostrato tramite la differenziazione automatica non è altro che quello di backpropagation eseguito nelle reti neurali per la propagazione dell'errore all'indietro, ovvero calcolo delle correzioni di pesi tramite propagazione del contributo di errore di ogni connessione all'indietro, in base alla differenza fra classe predetta e reale. Questo permette ora di capire perché questa tecnica, insieme alla metodologia del dataflow programming, sia fondamentale nel successo della libreria, motivo per il quale sono state entrambe implementate ed integrate al suo interno, soprattutto per elaborazioni con le reti neurali.

Di seguito avverrà una panoramica degli strumenti disponibili per lo sviluppo di architetture di reti neurali.

4.2.1.3. Componenti principali di Tensorflow

I principali componenti di questa libreria, almeno per lo sviluppo di reti neurali, sono:

- Funzioni di errore (mean squared error, logarithmic loss, cross entropy, ecc…);
- Funzioni di attivazione (softmax, ReLU, sigmoide, tangente iperbolica, ecc…);
- Metriche per la valutazione delle performance, anche nel caso di uso della rete per classificazione (precisione, accuratezza, recall, f1-score, media dell'errore quadratico e tante altre);
- Layer, in cui sono presenti la maggioranza delle tipologie di livelli implementabili per reti neurali, cioè partendo da quelli di inizializzazione dei pesi fino a quelli di Batch Normalization, passando per i layer di convoluzione, max-pooling, Dropout e tanti altri;

I componenti presentati non sono tutti quelli possibili presenti nella libreria, ce ne sono tanti altri, però questo rende l'idea della completezza e maturità dello strumento, motivo per il quale è fra quelli più diffusi e seguiti dalla comunità open source. Grazie a questi componenti ed al modello di base costruito su dataflow e differentiable programming, si ha anche l'enorme vantaggio di avere uno strumento che può essere eseguito sia su architetture distribuite che su componenti specifici (CPU, GPU, TPU), dove si ha un enorme guadagno in termini di efficienza nell'uso di risorse, velocità di esecuzione e scalabilità. Altro vantaggio fondamentale è la relativa semplicità di utilizzo (seppure non banale), che permette di realizzare architetture di rete potenti e molto diverse fra loro anche a persone non prettamente di formazione informatica o con elementi di programmazione forti (seppur siano necessarie alcune conoscenze di base).

Queste caratteristiche hanno permesso lo sviluppo di distribuzioni particolari di Tensorflow, realizzate appositamente per l'esecuzione su schede video e schede appositamente create per

velocizzare il calcolo generale delle reti neurali (TPU [44]), cosa che permette di accelerare il training di diversi ordini di grandezza rispetto al tempo richiesto su CPU tradizionali [45][46].

Una volta introdotto e descritto Tensorflow, è ora di parlare dell'altro componente fondamentale nello sviluppo di reti neurali in questa tesi, ovvero Keras.

4.2.2. Keras

In origine Keras (in figura 38) doveva essere una semplice interfaccia, non un ambiente a se stante di machine learning. Con il tempo invece è diventato un ambiente di sviluppo vero e proprio, vista la semplicità di utilizzo, la user-friendliness, la sua natura modulare ed estensibile.

Figura 38

L'obiettivo principale di questo strumento dunque era il permettere uno sviluppo veloce di modelli di rete neurale, grazie ad un'interfaccia semplificata ma allo stesso tempo completa ed alla sua interoperabilità con altri framework di elaborazione. Keras dunque si pone come una sorta di "Middleware", in cui con un unico insieme di API permette costruzione, addestramento ed elaborazione di modelli di rete neurale eseguibili tramite diversi motori di esecuzione trasparentemente.

I motori di esecuzione supportati da Keras sono:

- TensorFlow;
- Microsoft Cognitive Toolkit (https://www.microsoft.com/en-us/cognitive-toolkit/);
- PlaidML (https://www.intel.ai/plaidml/);
- Theano (http://www.deeplearning.net/software/theano/).

Per questo motivo è possibile creare tranquillamente i modelli di rete neurale, senza doversi preoccupare del motore di esecuzione (può essere impostato uno dei quattro presentati sopra, di default è TensorFlow). Dalla versione 2.0 di TensorFlow, Keras sarà integrato direttamente nella libreria stessa come API principale di alto livello (anche Microsoft Cognitive Toolkit integra già la libreria).

Un altro strumento fondamentale è stato MySQL (https://www.mysql.com/it/), per poter analizzare il database fornito da SiCS contenente gli eventi delle partite di calcio di due stagioni di serie A italiana.

4.2.3. MySQL

Lo strumento utilizzato per analizzare ed estrarre conoscenza dal database fornitoci è stato MySQL, grazie al quale è stato possibile analizzare il database per comprenderne meglio la struttura ed il metodo di rappresentazione del dominio degli eventi del calcio.

Una volta compreso il modello dei dati, effettuando le opportune query, è stato possibile scegliere il metodo di codifica degli input più adatto all'utilizzo tramite reti neurali, preceduta da correzione dei dati errati, eliminazione dei dati in esubero o mancanti e selezione degli attributi e le relazione di interesse per la classificazione, tutte attività tipiche delle procedure di ETL (Extraction, Transformation and Loading, capitolo 1).

L'ultima fase è stata quella di caricamento dei dati dunque, in un formato elaborabile da algoritmi di machine learning per reti neurali (in questo caso è stato scelto il formato CSV).

Il paragrafo che segue contiene la spiegazione tecnica ed i dettagli dell'implementazione delle reti neurali.

4.3. Strategia di risoluzione

Nell'ultimo paragrafo avverrà un'analisi approfondita della soluzione adottata, insieme ad una spiegazione del perché di tutte le scelte progettuali effettuate, insieme ai principali parametri di setting usati per le reti neurali.

Verranno mostrate anche graficamente le due reti realizzate ed i risultati sperimentali di performance tramite le metriche di classificazione scelte.

4.3.1. Rappresentazione del dominio

Gli eventi recuperati dal database di SiCS riguardano una serie anche molto differente di interazioni durante la partita, dunque il primo pensiero è stato quello di recuperare sia tutti gli eventi e sia recuperare quelli usati in origine per calcolare l'IPO, così da fare un confronto quantitativo delle potenzialità degli strumenti usati (reti neurali).

Per prima cosa dunque sono stati costruiti due differenti training e test, in una versione vengono presi in considerazione solo gli eventi con cui è stato costruito l'IPO originale mentre nell'altra si prendono in considerazione tutti gli eventi estraibili dal database.

In particolare, dato un numero n di eventi e dato ev_i l'i-esimo evento, i vettori che compongono il training set e test set, nel caso di indice calcolato seguendo le indicazioni originali, hanno un numero di dimensioni (eventi chiave) pari a 30 mentre nel caso di indice calcolato ex-novo sono 392.

In figura 39 possiamo vedere un esempio visuale di vettore nel caso di considerare gli eventi scelti dagli esperti (eventi per l'IPO originale).

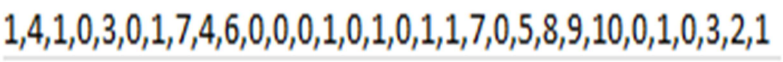


Figura 39

Come si può notare nell'immagine sopra, i singoli eventi sono diventati le "colonne" del vettore, dove ogni colonna rappresenta la somma di eventi avvenuti in una partita singola,

per la squadra di casa (i primi quindici elementi) e la squadra fuori casa, dove l'ultimo elemento rappresenta la classe (in questo caso "1", dunque pareggio). Formalmente, data la k-esima partita, il vettore che la rappresenta è $p_k = (ev_1, ev_2, \dots, ev_{n-1}, ev_n)$, dove ogni dimensione è il numero di occorrenze dell'evento i-esimo, a parte l'ultimo che rappresenta la classe, dunque il risultato finale rispetto però la squadra di casa, "0" per la vittoria della squadra di casa.

Per quanto riguarda l'output invece, si è deciso di rappresentare le tre possibili classi, cioè vittoria o pareggio o sconfitta, come dei valori numerici (0,1,2). A loro volta i valori zero, uno e due sono stati codificati secondo lo schema one-hot, diventando tre vettori binari composti da tre bit (figura 40).

Classi
0
1
2

One-hot Encoding		
0	0	1
1	0	0
0	1	0

Figura 40

Quello che si ottiene sono dei vettori di eventi, per ogni partita, frutto di un'aggregazione compiuta sul database fornito da **SiCS** e questo rappresenterà l'input della rete neurale mentre l'output sarà la classe predetta (nella forma però di valore zero, uno o due, non in forma di one-hot encoding).

4.3.2. Funzioni di pre-processing

Le partite raccolte dalla fase precedente sono 380 per ogni stagione, che rappresenta il numero di partite in un anno di campionato a venti squadre (struttura del campionato maggiore italiano allo stato attuale).

4.3.2.1. MaxAbsScale

A causa della forte diseguaglianza nel numero di occorrenze per ogni evento (alcuni "tag" detengono la maggioranza delle occorrenze), sono state necessarie varie fasi di pre-processing diverse, al fine di portare gli intervalli di valori per ogni evento ad avere valori limitati e definiti, nei vari dataset.

In figura 41 si può notare la distribuzione degli eventi nel caso del calcolo IPO originale considerando 30 eventi.

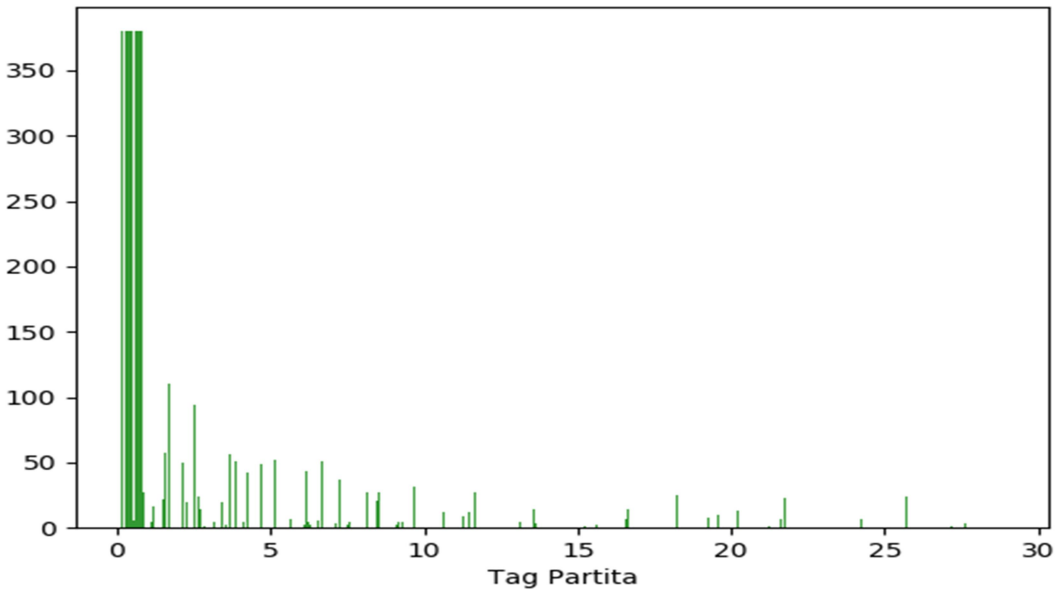


Figura 41

Si nota subito in figura 41 come esiste un forte squilibrio di occorrenze per i diversi tag, motivo per cui è stato eseguito un tipo di pre-processing chiamato "MaxAbsScaler", ovvero ogni evento, singolarmente, è stato portato in intervalli di valore massimo uguale ad "1" per il valore più alto presente. Ecco un esempio dell'operazione:

$$x_1 = (4,1,0),\ x_2 = (2,1,-1)$$

$$MaxAbsScale(x_1, x_2) \rightarrow x_1 = (1,1,0),\ x_2 = (\frac{1}{2}, 1, -1)$$

L'operazione è svolta prendendo il massimo per ogni colonna (o dimensione) e dividendo la magnitudo di ogni dimensione per il massimo valore assoluto presente.

L'operazione scala dunque tutti i valori fra [-1, 1], cosa importante a scopi di convergenza per le reti neurali ma anche per cercare di non indurre nel classificatore una maggiore attenzione verso certi eventi solo determinata dalla magnitudo, invece che per la loro reale capacità di predizione della classi di uscita. Altro effetto per cui è stata usata questa funzione è che non stravolge comunque le caratteristiche del training set, così da non rischiare di cambiare l'effettiva distribuzione degli eventi delle partite[47], dove invece una standardizzazione maggiore dei dati o l'eliminazione dei valori "estremi" potrebbe far perdere le caratteristiche di unicità che caratterizzano il calcio come sport, minando le capacità di predizione del modello. D'altro canto il calcio è caratterizzato da una forte componente casuale di per se, in quanto le partite si determinano con punteggi risicati, a differenza di altri sport come il basket dove essendoci alti valori di punteggio è più semplice caratterizzare gli elementi determinanti l'esito di una partita (nel calcio un singolo goal fortuito può portare alla vittoria, dove invece si hanno punteggi molto più elevati è molto più difficile che un singolo evento fortuito determini una gara).

4.3.2.2. Smote

Questa tecnica è l'acronimo di "Synthetic Minority Over-sampling TEchnique", una tecnica di sovra-campionamento delle classi di minoranza, utile ad affrontare l'altro problema di cui soffriva il dataset ovvero lo sbilanciamento delle classi da predire (figura 42).

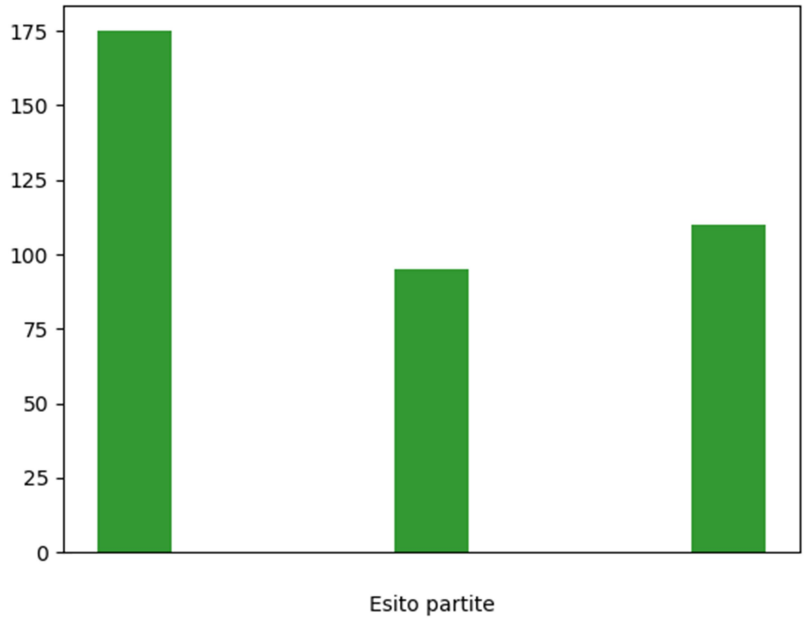


Figura 415 (In figura la classe a sinistra è quella rappresentata dalla vittoria, quella al centro dal pareggio e quella a sinistra dalla sconfitta.)

Quando si addestrano i modelli di classificazione in generale, è necessario prendere in considerazione lo sbilanciamento fra classi, in quanto se non si adoperano contromisure è molto probabile che le classi che verranno predette con un maggior grado di accuratezza saranno quelle presenti in maggioranza, a scapito di tutte le altre. L'errore di predizione in se per se può essere dipendente anche dal dominio, in quanto non tutte le classi predette erroneamente hanno la stessa gravità, in termini di impatto. Immaginiamo ad esempio lo sviluppo di un modello di predizione della presenza di una malattia, chiaramente è preferibile ottenere un modello che da un maggior numero di falsi positivi rispetto a falsi negativi, considerando che si sta parlando di salute di un paziente, così come in generale gli errori effettuati su classi diverse possono non avere lo stesso peso, a seconda della realtà sotto considerazione [48].

Fatte queste premesse, la tecnica dunque si occupa di creare dei nuovi punti (dunque dei nuovi vettori p_k di partite) partendo dalle classi di minoranza.

I passi dell'algoritmo si possono dunque riassumere in:

- Selezione della classe o classi di cui dover creare nuove istanze;
- Impostazione della percentuale di oversample $N_{\%}$ (posso decidere la percentuale di sovra-campionamento, dunque il 200%, 300% o quantità maggiori delle classi scelte);
- Decisione del numero k di nearest neighbour da prendere in considerazione;
- Calcolo, per ogni punto di ogni classe di minoranza ed in base al fattore di oversample, della distanza fra punto i-esimo e k-esimo nearest neighbour d_{ik};
- Campionamento, da una funzione di distribuzione di probabilità uniforme, di un valore $r \epsilon [0,\ 1]$;
- Creazione del punto sintetico come somma delle sue coordinate nello spazio n-dimensionale con il vettore distanza d_{ik} moltiplicato per r (di fatto il nuovo punto creato è generato casualmente nell'asse di congiunzione fra punto i e k-esimo vicino).

Il vantaggio di questa tecnica rispetto a quelle di sotto-campionamento della classe di maggioranza o di sovra-campionamento di quella di minoranza sta nell'approccio di generazione dei nuovi punti, in quanto il generare individui della popolazione non come copia di punti pre-esistenti, rende il dataset più generale e favorisce dunque la creazione di modelli di classificazione più robusti per quanto riguarda overfitting o valori fuori scala (eccezioni) [48]. Di seguito l'immagine della proporzione delle classi nel dataset dopo SMOTE:

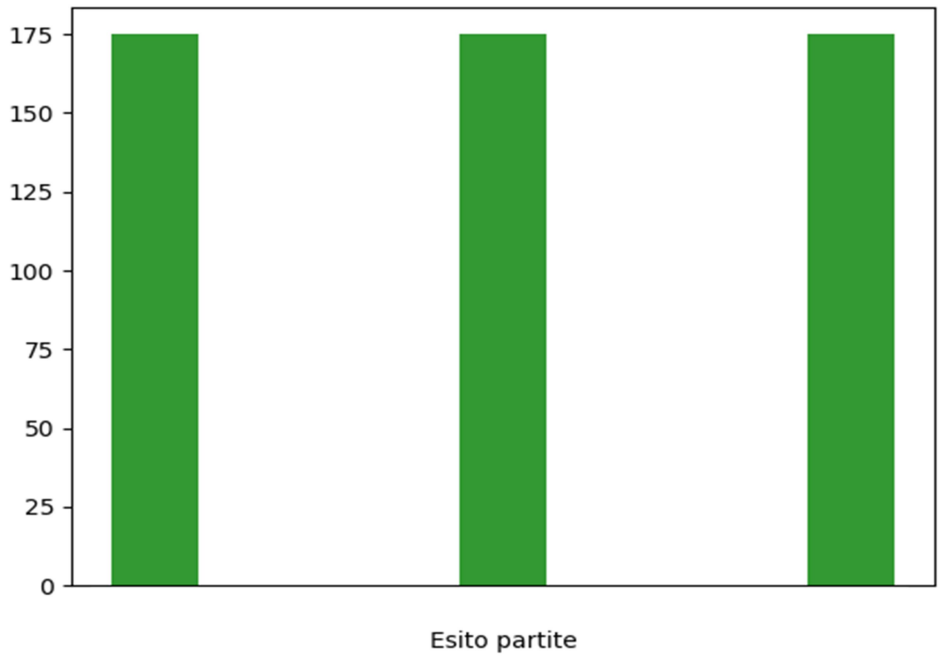


Figura 43

Il prossimo paragrafo sarà invece dedicato alla spiegazione delle reti neurali sviluppate, insieme alla loro rappresentazione grafica ed i parametri di setting utilizzati nello sviluppo del modello.

4.3.2.3. Topologie di rete e parametri di settings

Come è stato anticipato durante tutto il capitolo, la realizzazione della predizione è stata divisa in due reti neurali separate, una che esegue una stima dell'IPO originale e l'altra che esegue un calcolo ex-novo di questo parametro. La struttura delle due reti viene illustrata nelle figure 44(IPO originale) e 45(IPO ex-novo).

Figura 44

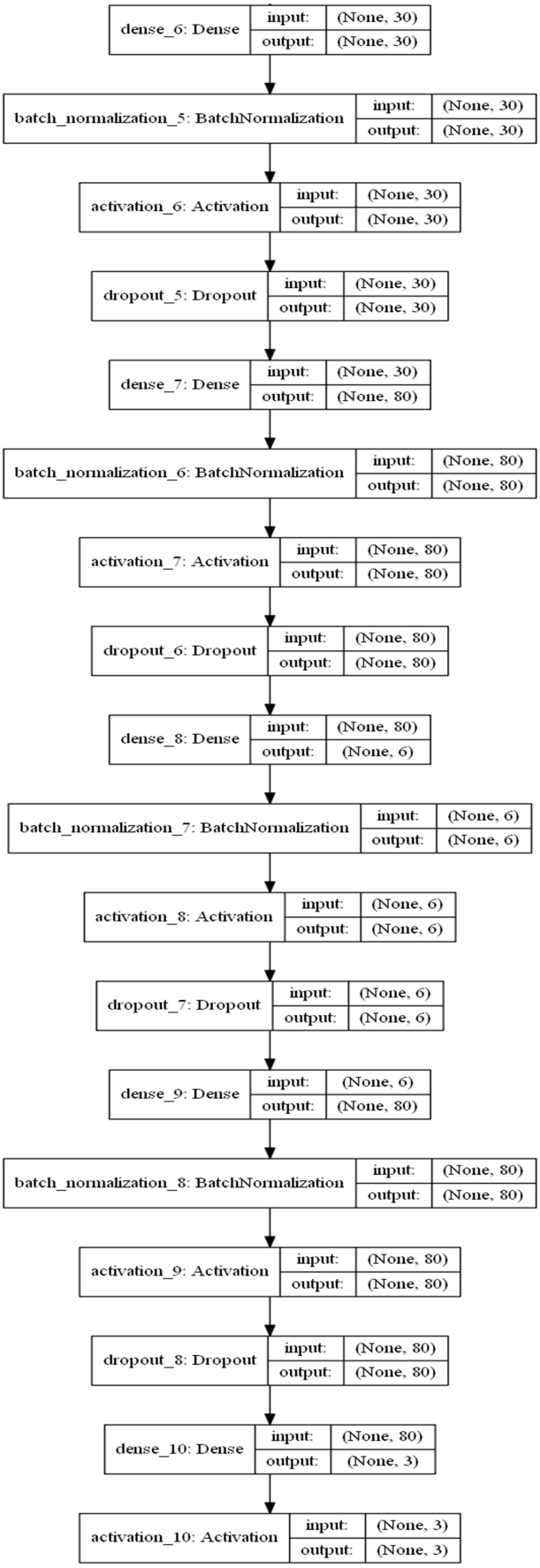
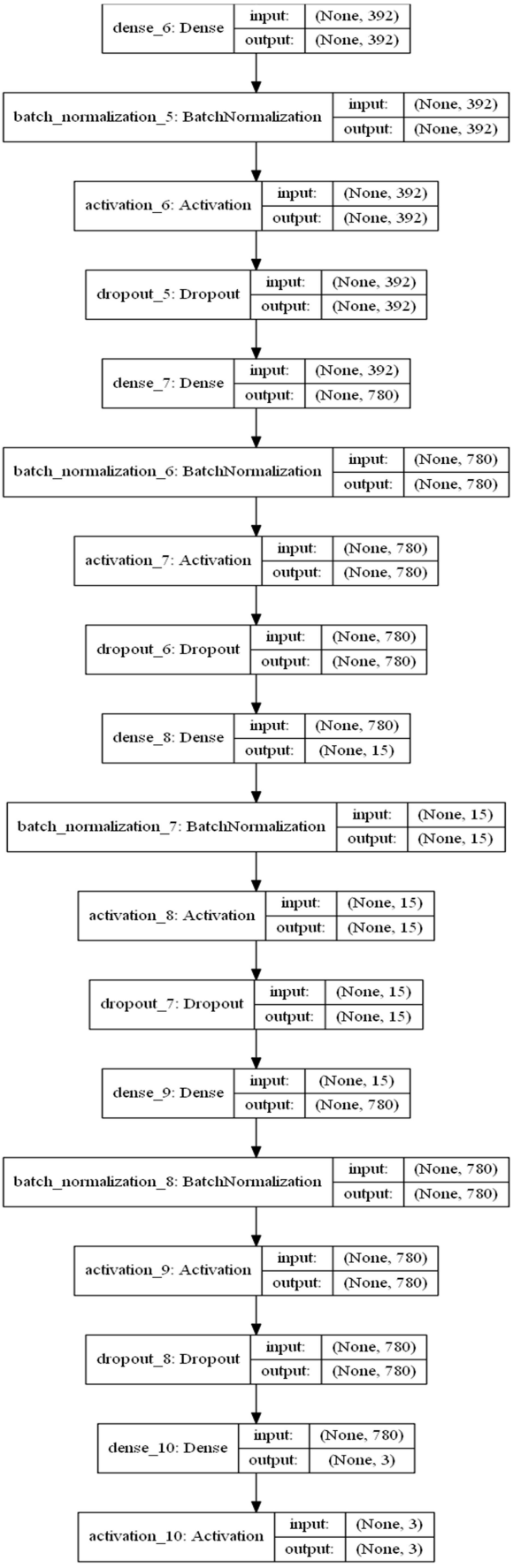


Figura 45



Nelle figure si può notare come le due reti abbiano la stessa struttura, con gli stessi layer, cambia solo la dimensionalità dell'input (in un caso si hanno i soli eventi dell'IPO originale mentre nell'altro caso sono tutti). L'unico componente che non è rappresentato nelle immagini è il layer di inizializzazione (secondo l'inizializzazione di He), incorporato in ogni livello "Fully Connected" o Denso.

Alcuni fra i principali parametri di configurazione delle due reti sono:

- La scelta del seme random, fondamentale per la riproducibilità dei risultati, in quanto la presenza dei layer di inizializzazione porta ad avere degli elementi di casualità ulteriori. Infatti se non si imposta un determinato seme da cui questi numeri vengono generati, rischiamo di introdurre un fattore di incertezza e variabilità delle performance non dovuto solo ai parametri ed iperparametri di addestramento;
- La scelta della funzione di loss, in questo caso è stata scelta la "categorical cross entropy", ovvero la differenza fra la distribuzione effettiva di probabilità reale p e la sua stima q (calcolata tramite la rete neurale, formula della funzione specifica di loss nel Capitolo 2) delle classi. E' utilizzabile nel caso di classificazione multi-classe;
- Le metriche di performance scelte per la rete sono l'errore quadratico medio e l'accuratezza, entrambi utilizzati dalla rete per monitorare il training;
- Come metriche di valutazione della classificazione invece, sono state utilizzate precisione, recall, f1-score (media armonica fra precisione e recall) e la kappa di cohen;
- Metodi di discesa del gradiente, i tre utilizzati sono stati AMSGrad (una variante di Adam che ne garantisce la convergenza in tutti i casi[49]), Nadam ed RMSProp. Il learning rate di RMSProp è fissato a 0.005;

Fra gli iperparametri utilizzati invece, dunque non sottoposti ad addestramento ma scelti con i test e le prove, ci sono:

- Il numero di epoche, impostato a 150 (quante volte eseguo un intero addestramento sul training set);

- Il parametro k, per scegliere le proporzioni di validation e training set nella Cross Validation (utilizzata in questa implementazione per entrambe le reti);
- Dimensione del mini-batch, impostata a 76, ovvero il numero di iterazioni su cui calcolare l'errore medio e dunque effettuare la discesa del gradiente (si riduce dunque la variabilità dell'errore e si ottiene un apprendimento più stabile);

Altri metodi di tuning delle reti sono stati ottenuti utilizzando early stop con valore di pazienza pari a 30 (epoche che si è disposti ad aspettare prima che l'errore diminuisca) ed addestramento delle reti su training set solo dopo che è stato sottoposto a mescolamento delle istanze (così evito che l'addestramento e la fase di predizione siano influenzate dall'ordine stesso delle singole istanze nel training e test set). Early stop e shuffle dunque cercano di rendere i modelli più generali e di evitare che vadano in overfitting.

4.4. Risultati sperimentali

In questo ultimo paragrafo vengono mostrate le performance dei due modelli di rete neurali diverse ed anche per ogni metodo di discesa del gradiente utilizzato.

Per fare il punto, vengono di nuovo esplicitamente indicati i parametri di configurazione usati (uguali per entrambe le reti):

- $Epoche = 150$;
- $mini\_batch = 76$;
- $patience\_early\_stop = 30$;
- $k = 5$ ed $n\_repeats = 10$, entrambi parametri della cross validation, dove il secondo parametro indica il numero di ripetizioni del k-fold, con training e validation set sempre diversi, per avere un valore più affidabile delle performance.

Non è stato necessario, nel caso del k-fold, eseguire esplicitamente un campionamento stratificato delle classi, in quanto l'utilizzo di SMOTE sul dataset prima di sottoporlo ad addestramento ha creato un perfetto bilanciamento fra le classi.

I risultati vengono ora riportati in tabella, suddivisi per rete, ottimizzatore e metriche di performance utilizzate.

Le metriche di riferimento usate per il calcolo sono state la media e la deviazione standard, ottenute tramite dieci esperimenti indipendenti, eseguiti per ogni ottimizzatore di ogni rete e tramite k-fold ad ogni iterazione (per ridurre al minimo la variabilità dei risultati).

In tabella 1, 2 e 3 vengono mostrate le performance dei vari ottimizzatori per la rete1 (calcolo dell'IPO originale).

Tabella 1

Nadam	Media	Deviazione standard
Recall	0.57910526	0.01287496
Precisione	0.57708839	0.0099425
F1-score	0.5770596	0.01025678
Kappa di Cohen	0.33018788	0.01837001

Tabella 2

Adam	Media	Deviazione standard
Recall	0.58236842	0.00873193
Precisione	0.57347052	0.00849466
F1-score	0.57725632	0.00813584
Kappa di Cohen	0.33023166	0.01367159

Tabella 3

RMSProp	Media	Deviazione standard
Recall	0.55615789	0.01615892
Precisione	0.55955491	0.01213479
F1-score	0.55682181	0.01290249
Kappa di Cohen	0.29613999	0.02153339

Come si può notare dalle tabelle, almeno per ciò che riguarda la rete 1, la media dei risultati di predizione è piuttosto bassa, infatti anche nel caso migliore, tramite ottimizzatore Adam, non si va oltre una precisione media del 57,3% circa nella predizione delle tre classi.

Questo risultato è giustificabile in quanto lo scarso numero di eventi che si prende in considerazione all'origine, quindici per ogni squadra ed in totale trenta per partita, non sono sufficienti alla rete per apprendere degli schemi che permettano un'accurata predizioni dei risultati.

Nel caso della prima rete infatti il fattore soggettivo dato dall'esperienza degli esperti del settore e dei fautori dell'indice si dimostra dunque fondamentale per riuscire ad avere buone predizioni, dove invece la rete fa fatica ad emularne la conoscenza pregressa e l'esperienza.

Nelle tabelle 4, 5 e 6 invece si mostreranno i risultati della rete neurale dedicata allo sviluppo di un "nuovo" indice IPO, costruito sulla falsa riga dell'originale ma con la rete che può liberamente scegliere fra tutti gli eventi di una partita, dunque può cercare schemi e percorsi che ne determinano il risultato autonomamente, facendosi "guidare" dai dati.

Come per i precedenti, si mostrano i risultati divisi per ottimizzatore e con i dati di media e deviazione standard ottenuti per ogni metrica di riferimento scelta.

Tabella 4

Nadam	Media	Deviazione standard
Recall	0.74957895	0.0249618
Precisione	0.74404082	0.01865402
F1-score	0.74107477	0.02167944
Kappa di Cohen	0.59276744	0.03787124

Tabella 5

Adam	Media	Deviazione standard
Recall	0.76252632	0.01083188
Precisione	0.74624985	0.00700074
F1-score	0.75116869	0.00750818
Kappa di Cohen	0.61259268	0.01510976

Tabella 6

RMSProp	Media	Deviazione standard
Recall	0.73694737	0.0213009
Precisione	0.74106075	0.01869357
F1-score	0.73600319	0.01830138
Kappa di Cohen	0.57988747	0.03169984

Nella rete 2, si può notare una forte differenza nei risultati ottenuti.

La costante anche in questo caso specifico è il metodo di ottimizzazione che fornisce i migliori risultati, di nuovo Adam, seppur con un margine piccolo di miglioramento rispetto al secondo migliore, come nel caso della rete 1.

A livello puramente di performance però c'è una differenza non trascurabile nella capacità di predizione fra i due modelli, dove la rete2 ottiene un miglioramento del 17% netto ul migliore della rete1, passando ad una precisione media di poco inferiore al 75%.

Il risultato ottenuto quindi dalla rete 2, che utilizza un nuovo approccio di esplorazione nello spazio degli eventi della partita, è incoraggiante e dimostra come le reti neurali possano essere applicate con successo anche ad uno sport con interazioni complesse ed un'elevata componente casuale come il calcio. E' necessario menzionare anche come le performance superiori si siano ottenute solo grazie alla libertà fornita alla rete di avere accesso a tutti gli eventi delle partite, dove invece un approccio di stima ed estensione che seguisse l'indice già sviluppato si è dimostrato scarsamente efficace.

Conclusioni

Il lavoro di tesi svolto, si è focalizzato principalmente sullo sviluppo di nuovi strumenti di supporto, estensione ed integrazione di un indice chiamato indice di pericolosità offensiva (IPO), con risultati alquanto promettenti.

Ciò che si evince dal confronto effettuato fra la rete neurale di simulazione dell'indice originale e l'altra di sviluppo di un indice ex-novo, agnostico rispetto al dominio di applicazione, è l'accresciuta capacità di predizione.

Se infatti la prima rete, come mostrato, ha performance medie di precisione che si attestano sul 58% circa, la seconda con totale libertà di elaborazione ha la meglio con una performance di circa il 75%.

Questo mostra non solo come le nuove tecnologie, o perlomeno quelle più in voga, del Data Mining ed in particolare le reti neurali (profonde in questo caso) siano capaci a svolgere anche compiti complessi ma apre a possibili nuovi scenari, in cui l'utilizzo di questa tecnologia può essere abilitante alla diffusione di strumenti di supporto all'avanguardia, per l'analisi tattica in tempo reale, previsione delle partite nell'ambito del calcio.

Merita approfondimento anche la diffusione di una "cultura" nell'ambito di questo sport orientata alla creazione e gestione di dati più precisi e capillari, dove ad esempio la pallacanestro americana si è mossa in anticipo creando un circolo virtuoso in cui anche realtà un tempo minori sono riuscite ad emergere, aumentando la competitività e spettacolarità del torneo a vantaggio di addetti ai lavori, pubblico, business. Si ritiene dunque necessario un processo di revisione, ai vertici del calcio italiano e non solo, in cui si stimoli l'introduzione di nuove tecnologie per la raccolta e messa a disposizione di dati delle partite, performance fisiche dei calciatori, così da ottenere strumenti sempre più accurati ed utili a questo sport.

Infatti, nonostante i risultati ottenuti siano molto positivi, ci sono ampi margini di estensione e miglioramento del sistema, dove come già accennato l'introduzione di parametri fisici dei

calciatori come indicatori dello stato di salute, stress muscolare, stress psico-fisico, performance atletiche, potrebbero essere sicuramente di enorme supporto nello sviluppo di modelli di predizione sempre più accurati, che colmino quel solco informativo presente, causa principale dei limiti di performance attuali.

Bibliografia

[1] https://it.wikipedia.org/wiki/Data_mining

[2] https://www.vidiemme.it/ambiti-utilizzo-data-mining/

[3] https://en.wikipedia.org/wiki/C4.5_algorithm

[4] On the effects of dimensionality on data analysis with neural networks, autori M. Verleysen, D. François, G. Simon, V. Wertz

[5] https://en.wikipedia.org/wiki/Dimensionality_reduction

[6] https://en.wikipedia.org/wiki/Principal_component_analysis

[7] Deep Learning, autori Ian Goodfellow, Yoshua Bengio, Aaron Courville, Capitolo 2, pagine da 45 a 49.

[8] https://en.wikipedia.org/wiki/Standard_score

[9] Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance, autori Maciej A. Mazurowski, Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker, Georgia D. Tourassi

[10] https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

[11]https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

[12] https://en.wikipedia.org/wiki/Minimum_description_length

[13] https://en.wikipedia.org/wiki/AlexNet

[14]https://ai.googleblog.com/2015/07/deepdream-code-example-for-visualizing.html

[15] https://en.wikipedia.org/wiki/Artificial_neural_network

[16] https://en.wikipedia.org/wiki/Connectionism

[17] Deep Learning, autori Ian Goodfellow, Yoshua Bengio, Aaron Courville, Capitolo 6, pagine da 197 a 198

[18] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pagine 1–13, 2015

[19] Timothy Dozat. Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, (1): 2013–2016, 2016

[20]http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[21] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011

[22] https://arxiv.org/abs/1609.04747v2

[23] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pagine 372–376, 1983

[24]Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pagine 1139–1147, 2013

[25] Deep Learning, autori Ian Goodfellow, Yoshua Bengio, Aaron Courville, Capitolo 8, pagina 281

[26] https://www.wikiwand.com/en/Vanishing_gradient_problem

[27] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification" arXiv preprint, vol. arXiv:1502.01852, 2015.

[28] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe, Christian Szegedy

[29] https://en.wikipedia.org/wiki/Lipschitz_continuity

[30] How Does Batch Normalization Help Optimization? Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry

[31] https://en.wikipedia.org/wiki/Early_stopping

[32] https://www.calcioefinanza.it/2019/01/11/manchester-city-sap/

[33] https://qz.com/1104922/data-analytics-have-revolutionized-the-nba/

[34] http://www.sics.it/

[35] http://www.ultimouomo.com/oltre-risultato/

[36] https://www.ultimouomo.com/a-che-punto-sono-le-statistiche-nel-calcio/5/

[37] https://www.sportmediaset.mediaset.it/calcio/mister-dove-vai-se-il-tattico-non-ce-l-hai-_1255000-201902a.shtml

[38] https://statsbomb.com/2013/08/goal-expectation-and-efficiency/

[39] Deep Learning, autori Ian Goodfellow, Yoshua Bengio, Aaron Courville, Capitolo 6, pagine 192-195

[40] https://en.wikipedia.org/wiki/Dataflow_programming

[41] https://en.wikipedia.org/wiki/Differentiable_programming

[42] A simple automatic derivative evaluation program, autore R.E.Wengert, anno 1964, doi:10.1145/355586.364791

[43] By Berland at English Wikipedia - Transferred from en.wikipedia to Commons by JRGomà., Public Domain, https://commons.wikimedia.org/w/index.php?curid=4860859

[44] https://en.wikipedia.org/wiki/Tensor_processing_unit

[45]https://www.tomshw.it/hardware/google-cloud-tpu-potenza-bruta-per-allenare-le-reti-neurali/

[46] https://doi.org/10.1145/3079856.3080246

[47]https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

[48] SMOTE: Synthetic Minority Over-sampling Technique, autore Nitesh V. Chawla et al., https://doi.org/10.1613/jair.953

[49] J. Reddi, Sashank & Kale, Satyen & Kumar, Sanjiv. (2018). On the convergence of Adam & Beyond.