Politecnico di Torino
Ingegneria del cinema e dei mezzi di comunicazione

Tesi di Laurea Magistrale

Generazione automatica di contenuti 3D per applicazioni di realtà virtuale mediante tecniche di machine learning

Relatore
Prof F. Lamberti

Candidato
Basilicò Gianpaolo

Sessione di Ottobre 2017

Dediche

"(...)Si faccia una vita interiore, di studio, di affetti, che non siano soltanto di "arrivare" ma di "essere", e vedrà che la vita avrà un significato"

Cesare Pavese

Alla mia famiglia, pilastro portante della mia vita .
Ad Aldo, Francesca, Giuseppe, Marco e Martina per i consigli, le parole e la musica.
A Torino.

Riconoscimenti

Ringrazio il Prof. F. Lamberti per avermi guidato lungo il percorso di questo lavoro.
Ringrazio Alberto Cannavò per i suoi consigli e il suo aiuto e tutto il laboratorio per avermi ospitato durante questi mesi.
Un enorme grazie a Giuseppe e Marco per il loro contributo pratico e, soprattuto, morale.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Grazie agli sviluppi a cui si sta assistendo nella Realtà Virtuale, è possibile oggi replicare artificialmente una partita sportivo così da consentire a chiunque di poterlo rivivere come se si fosse presenti. Lo sviluppo di queste simulazioni avviene grazie al tracking dei giocatori e della palla durante la partita. Ormai, in quasi in tutte le discipline sportive si sta assistendo ad un ampliamento delle soluzioni di acquisizione ed elaborazione dei dati, con una conseguente focalizzazione nella creazione, sfruttando, ad esempio, il machine learning, di informazioni e che possano essere direttamente utilizzate dalle società sportive per analisi prestazionali sia dei proprio atleti che degli avversari. Per questo motivo, col passare degli anni si sta assistendo ad un incremento di aziende specializzate nell'acquisizione di dati sportivi.
STATS [1], ad esempio, è una società che opera nel calcio oltre che in altri sport quali il football americano, il baseball, il rugby, il basket, ecc. Il principio fondamentale di STATS è quello di aiutare le persone interessate a prendere delle decisioni veloci che si basano sulle informazioni estrapolate da collezioni di dati. Per questo, la società si focalizza sul concetto di contesto, da desumere dai dati presenti nei database e come output degli algoritmi di machine learning che usa per raggiungere tale obiettivo. Questo permette di avere delle informazioni facili da analizzare e quindi, una velocità decisionale maggiore da parte degli utilizzatori di queste tecnologie (allenatori e giocatori, ad esempio).
Le tecnologie che oggi permettono di acquisire questi dati sono diverse, e spaziano da quelle di computer vision [2] a quelle che fanno uso di sensori come il GPS (Global Positioning Systems) [3]. Da entrambe le soluzioni si ottengono le informazioni dalle quali è poi possibile ricostruire le posizioni in campo da gioco sia dei giocatori che della palla. Come si è accennato precedentemente, la possibilità di utilizzare i suddetti dati assieme alla Realtà Virtuale ha condotto allo sviluppo di sistemi di simulazione capaci di riprodurre fedelmente un evento sportivo. Usando dei visori di Realtà Virtuale è possibile far rivivere una partita sia ai giocatori che agli allenatori, così da migliorare l'analisi delle prestazioni e poter utilizzare questo strumento in fase di allenamento. La partita virtuale offre una libertà di movimento assoluta e la possibilità di rivedere uno stesso momento della partita da diverse angolazioni. Questo comporta un risparmio sia di tempo che economico in quanto, oggi, per poter analizzare una porzione di partita cambiando angolazione è necessaria l'installazione di diverse telecamere. La replica virtuale di un partita porterebbe dei vantaggi anche dal punto di vista dell'intrattenimento, permettendo ai tifosi di riviverla in prima persona impersonificando, magari, il proprio giocatore preferito. La soggetti-

vità dell'esperienza potrebbe inoltre condurre ad una diversificazione della fruizione in base al giocatore o allenatore scelto per rivivere la partita e, di conseguenza, avere un impatto nella fase di merchandising di ogni squadra e nella promozione dei singoli atleti.
Questo è, per esempio, l'esperienza che prova ad offrire Beyond Sports in alcuni dei suoi prodotti commerciali [4]. La verosimiglianza è data dalla capacità di riprodurre in modo accurato, oltre alle posizioni, anche i movimenti e quindi le animazioni degli atleti durante la simulazione di una partita. Se infatti i dati spazio-temporali danno le informazioni riguardo le posizioni degli atleti, mancano quelli che permettono la ricostruzione virtuale dei loro movimenti.
In Beyond Mocap [5], una soluzione per la generazione automqtica di un evento sportivo, si è provato a migliorare la generazione dei contenuti sfruttando l'utilizzo dei dati di tracking e del motion capture per la creazione delle animazioni dei giocatori attraverso un sistema di annotazioni che identificano un particolare evento, come può essere, ad esempio, un passaggio, un tiro o un fallo, in un preciso momento. È in questo contesto che si inserisce il presente lavoro di tesi, chiedendosi se, sfruttando tecniche di machine learning, sia possibile eliminare il tempo necessario per le annotazioni creando un sistema in grado di comprendere, attraverso le informazioni spaziali dei giocatori e della palla, gli eventi che si susseguono, dando così un adeguato livello di verosimiglianza all'incontro sportivo virtuale.
A differenza di Beyond Mocap che sfrutta dei dati di tracking di un incontro calcistico, si adoperano in questo lavoro quelli prsenti su github e riguardanti i campionati NBA (National Basketball Association). Si sono utilizzati, in particolare, i dati relativi alla partita tra San Antonio Spurs e Minnesota del 23-12-2015 [6].
La possibilità di lavorare in un contesto sportivo differente da quello utilizzato in precedenti esperienze è stato spunto di considerazioni sulla capacità di categorizzare gli eventi di uno sport cercando di evidenziare le differenze che esso può avere rispetto agli altri, e sfruttare poi le informazioni ottenute nella fase di addestramento dell'algoritmo di machine learning. In questo caso, dopo aver studiato le regole del gioco del basket, ci si è concentrati sull'evento del passaggio, e si sono estratti dai dati a disposizione le caratteristiche che lo differenziano sia dagli altri tipi di eventi dello stesso sport, come possono essere il tiro e il fallo, sia dallo stesso evento in sport differenti.
Si è individuato il paradigma di interazione tra i giocatori e la palla nell'evento del passaggio attraverso lo studio delle sue peculiarità e si è proceduto poi:

1. all'estrazione delle caratteristiche dell'evento passaggio elaborando i dati di tracciamento messi disposizione da STATS;

2. alla generazione di un dataset formato sia dai dati di tracciamento che dalle feature precedentemente estratte;

3. all'annotazione del suddetto dataset utilizzando un video della partita (al fine di addestrare una serie di algoritmi di machine learning);

4. alla valutazione dei suddetti algoritmi ed alla identificazione di quello in grado di fornire le prestazioni megliori nelle condizioni considerate.

1.1 Obittivi di ricerca

Poiché la fase di annotazione manuale degli eventi di una partita risulta dispendiosa in termini di tempo e soggetta ad errori umani, l'obiettivo di questa tesi è , utilizzando i dati di tracking e il machine learning, la realizzazione di un sistema di annotazione automatico per il riconoscimento di eventi di una partita di basket e la rigenerazione dei contenuti attraverso una simulazione in Realtà Virtuale.
In particolare, dopo aver analizzato il lavoro presentato in [7], in cui si sono usate le informazioni di posizione relative alla palla per l'addestramento degli algoritmi di machine learning con lo scopo di riconoscere il passaggio nel calcio, si è focalizzata la ricerca sul riconoscimento del passaggio nel basket.
La possibilità di reperire, dai dati di tracking di STATS, le informazioni sull'altezza della posizione della palla (componente che invece risulta mancante nel lavoro di cui sopra) ha permesso, inoltre, di valutare in modo quantitativo e qualitativo l'influenza che questa componente e tutte le feature da essa derivate hanno sul miglioramento dell'efficienza di un algoritmo di machine learning applicato in tale contesto.

1.2 Organizzazione del documento

Il documento è organizzato in sei capitoli. Nel presente capitolo viene illustrato l'ambito in cui si è sviluppato il lavoro di tesi, introducendo nella sezione 1.1 gli obiettivi prefissati e nella sezione 1.2 l'organizzazione completa del documento. Le sezioni che seguono introducono il contesto del lavoro, presentando sommariamente i vari ambiti tecnologici toccati.
La sezione 1.3 presenta il concetto di simulazione e il suo legame con la Realtà Virtuale; la sezione 1.4 fornisce una panoramica sulla storia della Realtà virtuale mostrando che, come per la maggior parte delle tecnologie, non è possibile definire un percorso unico e tangibile che ha portato al suo sviluppo e al suo utilizzo. Esso è, invece, un intreccio di differenti fattori e contesti che hanno portato, col passare del tempo, all'evoluzione di particolari tecniche in grado di soddisfare differenti campi di applicazione. Nella sezione 1.5 viene introdotto il concetto di machine learning. La sezione 1.6 la tipologia dei dati raccolti nel tracking dei giocatori, chiamati dati (o serie) spazio-temporali.
Il capitolo 2 introduce lo stato dell'arte e mira a mostrare alcuni esempi d'applicazione del machine learning. In particolare, approfondisce il lavoro già menzionato in cui vengono analizzati i dati di tracking di alcune partite di calcio per il riconoscimento di eventi come il passaggio, ricezione, tiro in porta e rinvio [7], che funge da riferimento per la tesi. Il capitolo 3, mostra le tecnologie software sfruttate in questo lavoro evidenziando, in particolare, l'utilizzo di Visual Studio Code per la pre-elaborazione e l'analisi dei dati e di RapidMiner per il machine learning.
Nel quarto capitolo viene discusso il lavoro svolto in questa tesi presentando le feature sviluppate e spiegando come si è arrivati ai risultati valutati nel capitolo successivo, il quinto. Nel sesto ed ultimo capitolo si espongono le conclusioni e si accenna ad alcuni eventuali spunti futuri.

1.3 La simulazione

Con il termine simulazione ci si riferisce ad un modello dinamico e imitativo della realtà che, con il trascorrere del tempo, tende al cambiamento a causa di eventi che si susseguono e che rappresentano la simulazione stessa.
I contesti di applicazione di ambienti simulati sono molteplici e abbracciano campi quali il collaudo, l'educazione, il training e il mondo ludico.

La *simulation fidelity* è usata per la valutazione dell'accuratezza di una simulazione ed è importante perché è alla base di quello che distingue una simulazione da qualsiasi altro programma informatico. La distinzione tra i due tipi di applicazioni sta proprio nello scopo della simulazione che, rispetto a quello di un qualsiasi programma, è di rappresentare il comportamento di in oggetto nel mondo reale [8]. Essa è generalmente suddivisa in tre differenti categorie che, rispettivamente, rispecchiano maggiormente il mondo reale:

- *Bassa*, la fedeltà minima richiesta da un sistema per rispondere a degli input e fornire degli output;
- *Media*, il sistema risponde con limitata veridicità agli stimoli esterni;
- *Alta*, il sistema è il più simile possibile a quello reale.

La simulazione virtuale rappresenta una definita tipologia di simulazione che fa uso di specifici strumenti per la costruzione di ambienti simulati. Attraverso differenti strumenti di input e di output, l'utente ha la possibilità di interagire con un mondo fittizio che diviene, durante la simulazione, una nuova realtà, una Realtà Virtuale, appunto.

1.4 Storia della Realta Virtuale

La Realtà Virtuale è un ambiente artificiale in cui l'utente vive l'esperienza attraverso stimoli sensoriali (come quelli visivi e uditivi) forniti da un calcolatore e in cui una sua azione determina ciò che accade nell'ambiente stesso [9]. La costruzione di ambienti artificiali non ha di per sé uno stretto legame con l'evoluzione dei moderni calcolatori. Il Sensorama fu uno dei primi dispositivi a mirare allo stimolo della completa "dotazione sensoristica" umana (Figura 1.1). Ideato da M. Heilig nel 1962, aspirava a coinvolgere oltre che la vista e l'udito, anche l'olfatto e il tatto dello spettatore attraverso filmati pensi ad-hoc per essere usufruiti con l'apparecchio [10] . Il dispositivo era composto da una sedia in grado di muoversi sincronicamente con la simulazione, da un grande schermo stereoscopico, fornito di un congegno per la diffusione di odori, e dalle casse stereo. Il sistema era anche composto da un tunnel del vento capace di creare flussi d'aria e di emanare odori [11]. Il Sensorama non riuscì però ad attirare l'interesse degli investitori forse perché troppo in anticipo con i tempi.

Figura 1.1: Sensorama durante l'uso.
(fonte http://www.mortonheilig.com/InventorVR.html)

Tra il 1965 e il 1968, all'università dello Utah, Ivan Sutherland diede vita a quello che è considerato il primo vero visore di Realtà Virtuale. Il dispositivo faceva uso di due tubi a raggi catodici, uno per ogni occhio, ed elementi ottici per proiettare le immagini generate dal computer negli occhi dell'operatore. Come è possibile osservare in Figura 1.2, la grandezza del primo prototipo costringeva i ricercatori a impiegare un braccio collegato al soffitto per il sostegno e , questo, gli valse il nome di "The Sword Of Damocles", la spada di Damocle [12].

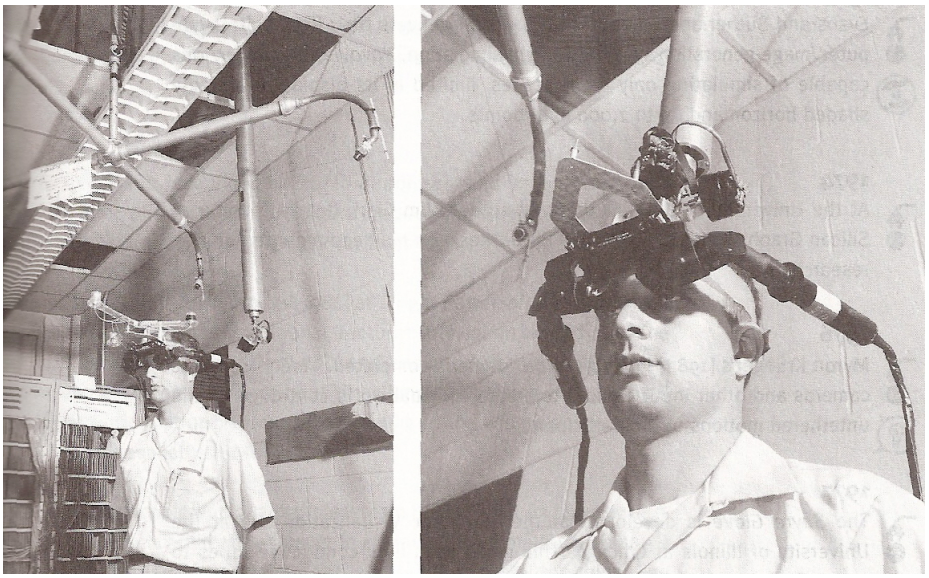


Figura 1.2: figura 2 Ivan Sutherland e la sua "spada di Damocle"
(fonte https://www.vrs.org.uk/virtual-reality/history.html)

Nel 1977 scienziati del MIT sviluppano l'Aspen Movie Map, una sorta di antenato di Google Street View in grado di portare gli utenti in giro per il Massachusetts

attraverso un tour virtuale [13]. Negli anni successivi è il mondo ludico a dare una forte spinta a questa nuova ed emergente tecnologia. Nel 1982, Atari crea un dipartimento di ricerca dedicato alla Realtà Virtuale e nel 1987 Jaron Lanier fonda la VPL Research, la prima compagnia in grado di mettere in commercio i prodotti di Realtà Virtuale. Alla VPL Research vengono infatti sviluppati dispositivi come l'Audio Sphere, il Data Glove e l'EyePhone (Figura 1.3).

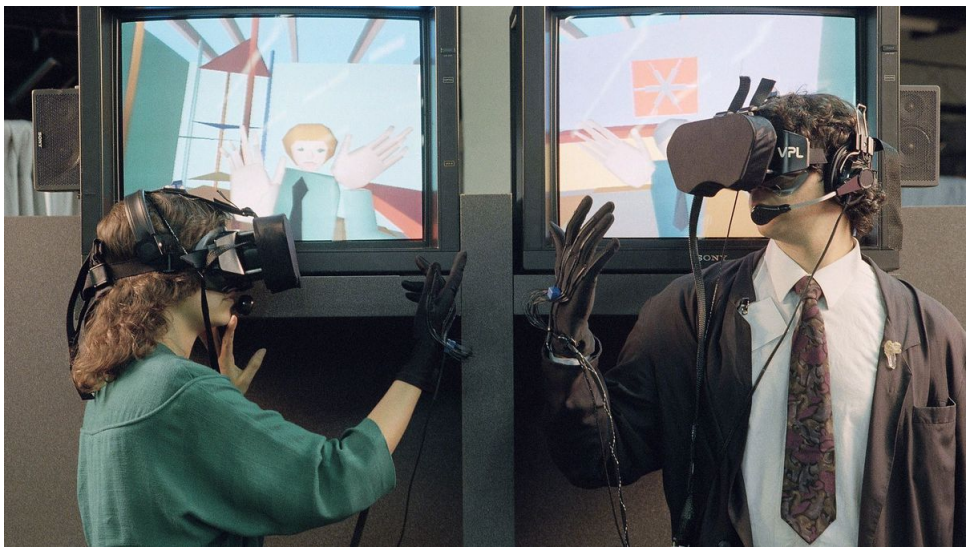


Figura 1.3: EyePhone, rilasciato il 7 Giungno 1989 (costo 1989)

Negli anni '90 è la volta di SEGA e di Nintendo, che sviluppano rispettivamente il Sega VR e il Virtual Boy, due visori pensati per fini ludici che non trovarono decisamente fortuna. Gli anni 2000 non videro sostanziali novità poiché lo sforzo fatto nei decenni precedenti e i vari fallimenti lasciarono una sorta di scetticismo generale che non portò grandi novità nel campo della Realtà Virtuale. La tendenza sta cambiando in questi ultimi anni, grazie soprattutto al progresso nel mobile che ha condotto a dispositivi con display sempre più definiti e a prezzi sempre più bassi in grado di facilitare l'ingresso della Realtà Virtuale nell'elettronica di consumo. È il caso, per esempio, di Oculus che nel 2012 , grazie ad una campagna kickstarter, finanziò lo sviluppo del visore denominato Rift (Figura 1.4).

Figura 1.4: Rendering dell' Oculus Rift della campagna kickstarter

1.4.1 Presenza e immersione

Oggi, la potenza a disposizione permette la creazione di mondi virtuali in grado di avvicinarsi sempre di più alla realtà. La sensazione di essere a tutti gli effetti in un mondo reale passa dalla capacità della tecnologia utilizzata di nascondersi all'utente nel momento della fruizione. La caratteristica della Realtà Virtuale di coinvolgere completamente il consumatore facendogli perdere la percezione di ciò che è reale e ciò che non lo è si basa sul concetto di *immersione* e su quello di *presenza*.

Immersione

Esistono diverse scuole di pensiero riguardo al significato di immersione in un ambiente virtuale. Witmer e Singer la definiscono come uno stato fisiologico in grado di restituire al fruitore la sensazione di essere "coinvolto in" e "in interazione con" un ambiente capace di restituire stimoli sensoriali continui. Altri ricercatori hanno avanzato una nozione di immersione come una misura oggettiva della capacità del sistema di riprodurre la sensazione fisica del mondo reale all'interno dell'ambiente virtuale con il quale l'utente interagisce. L'immersione dipende quindi dalla tecnologia utilizzata e da quanto essa sia in grado di riprodurre nel migliore dei modi gli stimoli sensoriali di un essere umano.

Presenza

Al contrario dell'immersione, la presenza è un concetto soggettivo e misura quanto il fruitore percepisce effettivamente di essere presente nel mondo virtuale. Anche in questo caso i ricercatori le hanno attribuito diverse definizioni. Per Sheridan, ad esempio, essa è definita come la sensazione d'essere trasportati in un ambiente sintetico creato artificialmente per opera di un calcolatore utilizzando differenti sistemi. Lombart descrive la presenza come uno stato fisiologico nel quale l'utente non percepisce l'uso della tecnologia nella fruizione dell'esperienza virtuale. Per O'Brian,

Büscher, Rodden, e Trevor avanzarono l'idea di presenza come un'estensione del mondo reale all'interno di quello virtuale, enfatizzando l'importanza di trasportare le attività eseguite normalmente nella realtà in contesto artificiale [14].

1.5 Machine learning

L'apprendimento automatico, o machine learning in inglese, è un campo dell'intelligenza artificiale che mira alla creazione di sistemi capaci di imparare e migliorare automaticamente senza essere esplicitamente programmati. Per risalire alla nascita di questo nuovo approccio informatico bisogna tornare indietro al 1959, alla definizione di Arthur Samuel [15]:

"Machine Learning: A field of study that gives computers the ability to learn without being explicitly programmed"

Arthur Samuel è uno dei pionieri del machine learning che, nel periodo in cui lavorò all' IBM, riuscì a sviluppare un programma capace non solo di imparare a giocare a scacchi, ma persino di batterlo.
Il presupposto del machine learning è quello di sviluppare algoritmi in grado di attingere da una collezione di informazioni e utilizzarli per l'auto-apprendimento. Questo processo avviene "istruendo" il sistema attraverso l'estrazione di "pattern" dai dati di input, al fine di poter creare degli schemi e utilizzarli nelle decisioni future. L'obiettivo primario è quello di permettere ai calcolcatori di imparare automaticamente e di adeguare le azioni senza l'intervento e assistenza da parte degli essere umani.

1.5.1 Algoritmi di machine learning

Gli algoritmi di machine learning sono spesso classificati come *supervisionati* o non *supervisionati* (in inglese *supervised* o *unsupervised*).

- *Supervisionati*: sono gli algoritmi che sfruttano l'apprendimento passato e lo applicano sui nuovi dati in ingresso. Attraverso l'uso di un data set di training, l'algoritmo di apprendimento diventa capace di fare predizioni future utilizzando le analisi svolte precedentemente e di propagare all'indietro gli errori commessi confrontando le predizioni con i valori reali, così da auto-migliorarsi. Il data set di training è formato da un insieme di record (tipicamente un vettore), ognuno dei quali ha associato un'etichetta (o label) che ne definisce la classe di appartenenza. Ogni coppia vettore-etichetta viene analizzata dall'algoritmo affinché venga generata una funzione in grado di associare ad un nuovo dato di input, che è quindi privo di etichetta, la classe appropriata.

- Gli algoritmi non *supervisionati*, viceversa, vengono utilizzati quando i dati di training non sono categorizzati. Questi sistemi esplorano il data set analizzando le possibili relazioni e cercando di dedurre dei pattern per descrivere il nesso tra i dati stessi.

- Esiste poi la categoria "ibrida" dei cosiddetti algoritmi *semi-supervisionati*, cche usano un data set di training formato sia da record dotati di etichetta

sia da record che ne sono sprovvisti. Sistemi di questo tipo sono capaci di migliorare considerabilmente l'accuratezza di apprendimento e vengono utilizzati quando la completa acquisizione di dati etichettati risulta impossibile.

- Infine, quello di *rinforzo* è un algoritmo di ricerca di tipo "trial and error" che riceve un segnale di ricompensa atto a precisare la correttezza della scelta fatta in risposta ad un dato in ingresso. Grazie a questo segnale, l'algoritmo corregge la propria strategia in modo da massimizzare la ricompensa. Questo metodo permette di determinare automaticamente il comportamento ideale in un contesto specifico così da aumentare le prestazioni.

1.5.2 Problemi risolvibili con il machine learning

In base al tipo di output che un determinato algoritmo fornisce, è possibile categorizzare tre principali problemi affrontabili con il machine learnin.

- *Regressione*
  È un problema di apprendimento supervisionato nel quale la risposta che deve essere appresa dal sistema è un valore continuo. È possibile fare l'esempio di un algoritmo che viene addestrato con un database di case vendute e il loro relativo prezzo, e riesce a definire quello di case in vendita che ne sono ancora sprovviste.

- *Classificazione*
  È un problema di apprendimento supervisionato nel quale si deve riuscire a definire il valore di un nuovo dato in ingresso tra tutti i possibili valori che il dato può avere. Quando ci sono solo due possibili valori che possono essere assegnati e si è, quindi, in presenza di due sole categorie si dice che il problema è un problema di classificazione *binaria*. Quando si è in presenza di più categorie ci si riferisce invece ad un problema di classificazione *multi-classe*. È comunque possibile trasformare un problema di classificazione multi-classe in problemi di classificazione binaria usando, per esempio, la tecnica del one-vs-all [16].

- *Clustering*
  Quello della segmentazione è un problema che ha l'obiettivo di raggruppare i dati di input in uno o più insiemi omogenei al loro interno. L'appartenenza ad un insieme viene deciso rispetto alla "vicinanza" del dato attualmente considerato ai dati precedentemente analizzati.

1.5.3 Processo di comprensione dei dati

Col passare del tempo, la quantità di dati che viene accumulata e storicizzata diventa sempre più grande. Si presenta quindi il bisogno di escogitare dei metodi in grado di aiutare nell'estrazione delle sole informazioni che siano effettivamente utili tra l'enorme mole di informazioni che oggi viene prodotta.
Queste tecniche fanno parte di un processo che prende il nome di KDD (Knowledge Discovery in Database).

Fayyad, Piatetsky-Shapiro e Smyth definiscono la KDD come il non semplice processo di identificazione di valide, nuove, utili e comprensibili relazioni nei dati [17].

La 1.5 mostra i passi di cui il suddetto processo è composto:

- selezione (selection);
- pre-elaborazione (preprocessing);
- trasformazione (transformation);
- estrazione (data mining);
- valutazione e interpretazione (interpretation/evaluation).

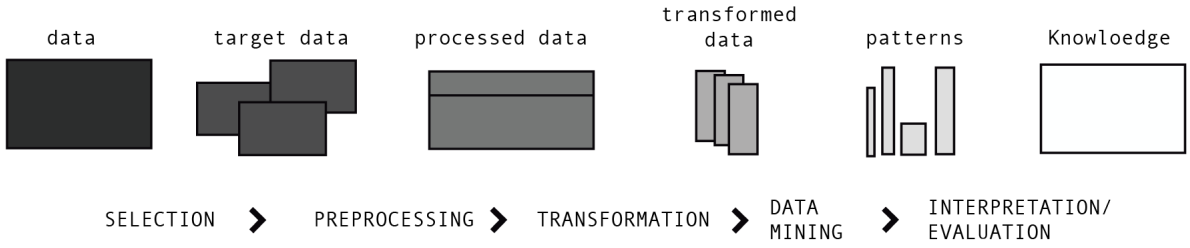


Figura 1.5: I Passi che illustrano il processo di comprensione dei dati

Selezione

Gli algoritmi supervisionati necessitano di un "addestramento" per poter costruire un modello capace di raggiungere l'obiettivo che, come si è detto, consiste nel determinare le caratteristiche che discriminano le differenze tra i dati in modo da poterli associare al giusto valore o classe di appartenenza. La selezione coinvolge quindi il processo di identificazione di un data set a cui è già accostata la relativa classificazione. Il data set viene poi diviso in due, cosi da creare il *training set* e il *test set*.
Gli algoritmi deducono, dal training set, i parametri che permettono di differenziare i dati e restituiscono in output un modello con le regole di predizione che verranno usate successivamente quando bisognerà associare ad un nuovo dato in ingresso un responso. Il test set viene invece utilizzato nella fase di validazione del modello.

Pre-eleaborazione

Una volta in possesso del training set è fondamentale una sua pre-elaborazione, in modo da aumentarne la qualità e la robustezza dei dati prima di darli in input ad

un determinato algoritmo.
Spesso si eseguono due differenti passi, ovvero:

- pulizia;
- integrazione;

La pulizia dà al training set una migliore qualità andando a lavorare sulla rimozione degli effetti di un ipotetico rumore di cui i dati possono essere affetti e degli *outliers* che ,come mostra la figura 1.6, sono delle tuple tuple con delle caratteristiche ampiamente differenti rispetto a quelle delle maggior parte dei dati.

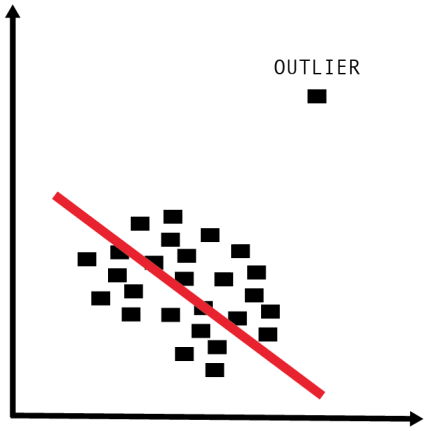


Figura 1.6: Identificazione degli outlier in un insieme di dati: i valori si discostano decisamente da quelli medi

Dopo la pulizia, l'integrazione mira ad aumentare l'efficienza del training set effettuando delle operazioni che possono essere delle tipologie illustrate di seguito [18].

- Aggregazione: combina due o più attributi in un singolo attributo. Questo permette di ottenere una riduzione della quantità dei dati e una loro maggiore stabilità.
- Campionamento: il suo utilizzo deriva dall'impossibilità di sfruttare l'intera basi di dati e/o dal fatto che elaborare un numero elevato di tuple richiede molto tempo e/o una elevata capacità di calcolo. Il campionamento riduce la cardinalità del data set.
- Creazione di nuove feature: consiste nell'analisi dei dati del data set iniziale e nella creazione di nuovi attributi da loro derivati in grado di rendere l'algoritmo di apprendimento più efficiente.

Trasformazione

La trasformazione mappa l'intero insieme di valori di un attributo con un nuovo insieme, in modo tale che ogni vecchio valore dell'attributo venga identificato con uno del nuovo insieme.

La normalizzazione è un tipo di trasformazione dei dati. Essa ha il compito di riposizionare un valore all'interno di un determinato intervallo. Le prassi di normalizzazione più comuni sono menzionate di seguito.

- *Normalizzazione Z-score*:riduce i dati considerando la media e la deviazione standard. È definita come:

$$norm = \frac{v - mean}{stanDev} \tag{1.1}$$

- *Normalizzazione Min-Max*: trasforma i dati in modo lineare in un intervallo compreso tra il valore minimo Min e il valore massimo Max ed è definita come:

$$norm = \frac{v - min}{max - min} \times (newMax - newMin) + newMin \tag{1.2}$$

- *Scalabilità decimale*:effettua la riduzione spostando il punto decimale del valore di un attributo ed è definita come:

$$norm = v \div 10 \times \frac{v}{10^j} \tag{1.3}$$

Estrazione

Questo passo del processo ha l'obiettivo di analizzare il data set e di trovare dei pattern in grado di evidenziare delle correlazioni tra i dati.
Lo scopo dell'analisi può essere quello di verificare una certa ipotesi generata dall'utente stesso sulla base della sua esperienza: in questo caso si parla di approccio deduttivo (*verification*).
Oppure, la scoperta di nuove relazioni tra i dati (*discovery*), con il fine di predire un comportamento futuro di una determinata entità (*prediction*). Oppure ancora, di evidenziare le relazioni stesse descrivendole in modo comprensibile per l'essere umano (*description*) [17] .

Valutazione e interpretazione

Come già scritto precedentemente, per la valutazione della bontà di un modello si ricorre alla suddivisione dell'intero data set etichettato in due data set differenti, e si impiega una prima parte come training set per l'addestramento del sistema ed una seconda parte come test set per la validazione dello stesso.
È possibile procedere al partizionamento del data set seguendo differenti approcci, tra cui quelli illustrati di seguito.

- K-fold cross validation: consiste nel ripartire il data set in k parti e di iterarlo k utilizzando per ogni passo un sottoinsieme diverso come test set e i restanti come training set. Un vantaggio non trascurabile di questo approccio è quello di ridurre la possibilità di overfitting, ovvero un adattamento eccessivo di un modello alle peculiarità del training set, che comporta una elevata prestazione nella valutazione del suddetto set (basso errore di predizione), ma un netto peggioramento nella valutazione invece del test set (alto errore di predizione).

La figura 1.7 mostra un classico esempio di overfitting. L'errore commesso nella fase di training set è rappresentato in azzurro, mentre quello commesso nella fase di test set è rappresentato in rosso. All'aumentare della complessità del modello, al crescere cioè del suo adattamento al training set, si evidenzia un abbassamento dell'errore di predizione del training set , ma un aumento di quello del test set. In questo caso è possibile avere una situazione di overfitting. La complessità ottimale del modello è quella che sia nel minimo globale della curva di errore di test set.

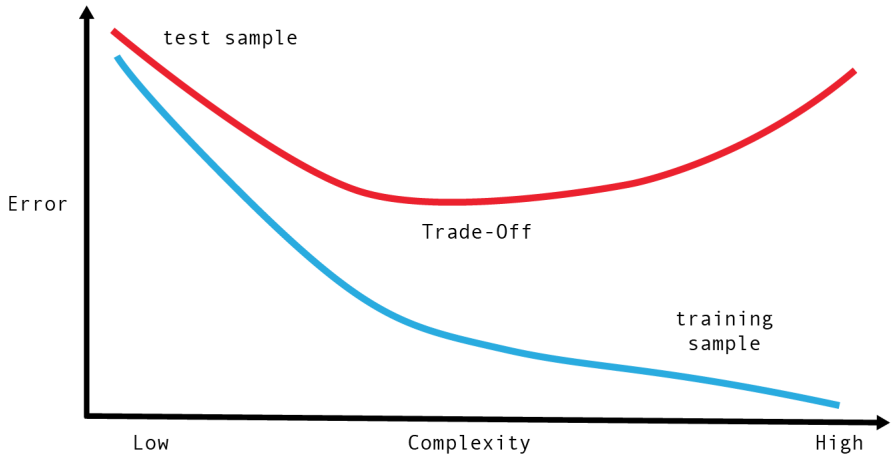


Figura 1.7: Esempio di overfitting in un algoritmo supervisionato

- Holdout: suddivide il data set in due partizioni differenti. Di solito $\frac{2}{3}$ per il training set e $\frac{1}{3}$ per il test set. Può essere considerato la variante più semplice della tecnica di cross validation

Alla fase di valutazione del modello, segue quella di definizione della metrica utilizzata per esprimere in maniera quantitativa l'efficacia di un algoritmo.
Le decisioni prese in fase di predizione sono rappresentate in una tabella chiamata "matrice di confusione", dove in colonna sono rappresentati i valori predetti dal classificatore e per righe i valori reali di ogni istanza (tupla) del test set.
Come mostrato in tabella 1.1, nel caso di un classificatore binario la matrice di confusione è formata dal numero di falsi positivi (in inglese, false positive o fp), falsi negativi (in inglese, false negative o fn), veri positivi (in inglese, true positive o tp) e veri negativi (in inglese, true negative o tn). Essi sono definiti come:

- falsi positivi: numero di istanze negative classificate come positive;
- falsi negativi: numero di istanze positive classificate come negative;
- veri positivi: numero di istanze positive classificate come positive;
- veri negativi: numero di istanze negative classificate come negative

	p (predetti)	n (predetti)
p(reali)	vp	fn
n(reali)	fp	tn

Tabella 1.1: Matrice di Confusione

È possibile derivare da questa tabella alcune metriche di valutazione, come illustrato di seguito.

- *Accuratezza*:il numero delle predizioni corrette diviso PER il totale delle predizioni fatte. Non sempre però, questa metrica è affidabile. Per esempio, nel caso di data set fortemente sbilanciati (in cui cioè il numero di oggetti di una classe è molto maggiore rispetto al numero degli oggetti dell'altra) e/o nel caso in cui una classe abbia maggiore importanza delle altre, il risultato che l'accuratezza restituita sarà fuorviante.
  La tabella 1.2 evidenzia come, nel caso di un data set fortemente sbilanciato, l'accuratezza non è rappresentativa delle reali prestazioni dell'algoritmo. Nel sottostante esempio l'obiettivo è predire quali, tra le mail in arrivo, dovranno essere etichettate come "Spam". Se il classificatore viene addestrato con un data set formato da 990 tuple etichettate come "Non-Spam" e 10 etichettate come "Spam" e le classifica tutte come "Non-Spam" l'accuratezza totale sarà del 99 % , ma nessuna , tra le tuple etichettate come "Spam" è stata predetta come tale.

	Spam (predetti)	Spam (predetti)	Accuratezza
Spam (reali)	0	10	0.0
Non-Spam(reali)	0	990	100.0
Accuratezza Totale			99

Tabella 1.2: Esempio di poca affidabilità dell'acuratezza

A causa dei suddetti problemi, è necessario utilizzare anche altre metriche [19].

- *Precisione* (in inglese, precision): sia C una delle classi a cui l'algoritmo può associare un record, allora la precisione è il numero di record correttamente assegnati a C diviso il numero totale di record assegnati a C. Questa metrica, come quelle a seguire, sono valutate separatamente per ogni classe C.

- *Richiamo* Richiamo (in inglese, recall): il numero di oggetti correttamente assegnati C diviso il numero di oggetti appartenenti a C.

- *F-Measure*: definita come la media armonica di precisione e richiamo, si esprime come:

$$fmeasure = 2 \times \frac{precisione \times richiamo}{precisione + richiamo} \tag{1.4}$$

- *Sensitività* (in inglese, sensitivity): frazione di veri positivi individuati, rispetto a tutti i positivi:

$$sensitivitá = \frac{tp}{tp + fn} \tag{1.5}$$

- *Specificità* (in inglese, specificity): frazione di veri negativi individuati, rispetto a tutti i negativi:

$$specificitá = \frac{tn}{fp + tn} \tag{1.6}$$

- Curva di ROC (*Receiver Operating Characteristic*): sviluppata durante la seconda guerra mondiale, questa curva è utile per la valutazione di classificatori binari, modelli che etichettano i dati usando due sole classi. La curva di ROC (fiura 1.8) viene ampiamente utilizzata in campo medico per visualizzare l'accuratezza di test diagnostici ed è definita dalla sensibilità e dalla specificità che vengono rappresentate in un diagramma rispettivamente sull'asse y e sull'asse x (1 – specificità). L'area sotto la curva (AUC, Area Under the Curve) misura l'accuratezza di un sistema e più questa si avvicina ad al valore uno, maggiore è l'accuratezza del sistema. Un'area di 0.5 dell'AUC corrisponde ad un sistema completamente casuale [20].

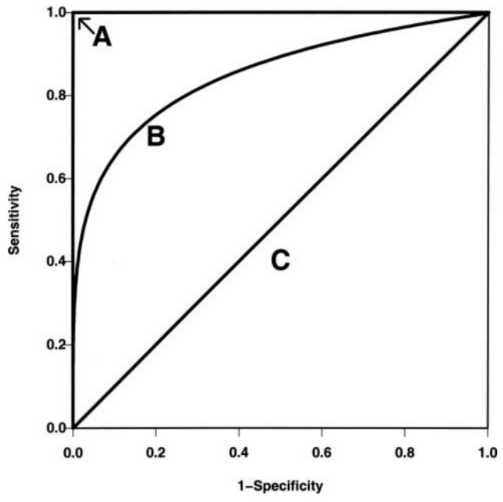


Figura 1.8: Tre differenti curve di ROC. La curva A rappresenta un algoritmo di classificazione perfetta (AUC=1), la curva B una curva di ROC con AUC=0.85 e la curva C un sistema completamente casuale (AUC=0.5)

1.5.4 Dati spazio-temporali

I dati, o serie, spazio-temporali riguardano oggetti in grado di modificare nel tempo la propria posizione e/o la propria forma [21]. Essi possono essere di due tipi:

- punto: un punto è un oggetto che si muove nel tempo di cui si considera solamente il cambiamento di posizione;

- regione:regione: una regione è un oggetto che si muove nel tempo di cui si considera sia il cambiamento di posizione sia il cambiamento forma.

Prendendo in considerazione oggetti puntiformi per esempio, è possibile definire una funzione del tempo (t) che può essere rappresentata come curva in uno spazio 3D (x, y, t) come quello mostrato in figura 1.9.

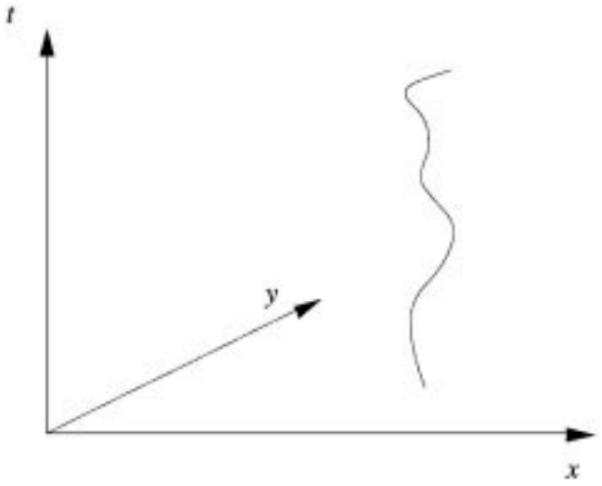


Figura 1.9: Esempio di un punto in movimento rappresentato in uno spazio tridimensionale

Capitolo 2

Stato dell'Arte

In questo capitolo vengono analizzate alcune applicazioni del Machine Learning e della Realtà Virtuale nello sport.

2.1 Applicazione del machine learning

I campi di applicazione del machine learning diventano sempre più numerosi. I database delle aziende diventano sempre più grandi e l'IoT (Internet of Things o Internet delle cose) genera, giornalmente, una enorme quantità di dati che diventa materia di analisi. L'informazione potenzialmente utile è "nascosta" e non evidente e la sua estrazione risulta non banale, per questo motivo, il connubio tra l'IoT e il machine learning si sta facendo sempre più stretto. Ormai quasi tutte le grandi compagnia hanno adottato dei sistemi per l'estrazione e l'analisi delle informazioni possedute. Nel frattempo all' I/O, l'annuale conferenza di Google, la stessa società annuncia che si appresta a rilasciare una versione lite della libreria open source TensorFlow che permette di estendere l'analisi dei dati si dispositivi mobili e di allargare quindi ulteriormente i possibili ambiti di applicazione [22] . Nel seguito, sono riportati alcuni tra i contesti in cui è frequente l'uso del machine learning.

Medicina

Il machine learning, fornendo gli strumenti utili alle diagnostiche mediche ed alla valutazione dei parametri clinici, viene impiegato per le prognosi, le previsioni, quindi, sul decorso e sull'esito delle malattie.
In questo campo, sistemi avanzati forniscono meccanismi per la generazione di ipotesi basati sui dati dei pazienti.
Per esempio, dato un set di casi clinici, è possibile sviluppare un sistema intelligente capace di produrre una descrizione sistematica delle feature che caratterizzano, in modo univoco, una condizione clinica.
Andre Esteva, per esempio, ha usato 129 mila e 450 immagine cliniche di malattie della pelle per addestrare una rete neurale, un algoritmo di machine learning, al fine di classificare lesioni cutanee. Il risultato è un algoritmo in grado di classificare un melanoma maligno o un carcinoma con la stessa accuratezza che avrebbe un esperto dermatologo [23]. È possibile evidenziare un altro campo di applicazione nell'elaborazione dei segnali biomedici. I segnali biologici sono caratterizzati da una variabilità aleatoria dovuta sia agli errori intrinseci delle tecnologie fisiche utilizzate

che a fattori esterni.
Il machine learning, sfruttando le diverse associazioni tra i parametri aleatori, determina, qualora ci fosse, una relazione esistente tra i dati, così da estrarre le caratteristiche in grado di portare un miglioramento ai diversi sistemi di cura [24].
Si può accennare a KARDIO [25], che può essere usato come uno strumento di diagnostica nella comprensione degli elettrocardiogrammi. KARDIO è un sistema medico sviluppato per la diagnosi delle aritmie cardiache, le quali performance sono paragonabili a quelle degli specialisti nella lettura dei segnali cardiaci.

Contextual marketing

La pubblicità è stata sottoposta, nel corso degli anni, ad un'evoluzione significativa per adattarsi alle nuove regole del web 2.0 e alle logiche dell'economia digitale. Questo ha portato, nel corso degli anni, ad una sostanziale modifica nel rapporto tra consumatore e brand.
Il vantaggio offerto dall'era digitale è la possibilità di "targettizzare" i consumatori in tempo reale, di anticipare le necessità e i bisogni del cliente e, quindi, di creare offerte sempre più personalizzate.
Questo nuovo modo di operare, detto appunto "contextual marketing", fa largo uso del machine learning per la generazione di pubblicità ad hoc.
Per questa ragione, sempre più attenzioni vengono rivolte a questo campo e continua a crescere il numero di strumenti utilizzati nell'online marketing.
Un esempio è possibile trovarlo in [26], dove viene proposto un sistema per l'estrazione delle informazioni più rilevanti dalle richieste effettuate tramite i motori di ricerca.

Traduzione automatica

La traduzione automatica, in inglese machine translation (MT), è un campo della scienza della traduzione che si occupa della traduzione dei testi da una lingua naturale ad un'altra attraverso l'uso di software informatici. Oggi, si possono individuare tre differenti tipologie di traduzione automatica:

- basati su regole linguistiche: viene scelta, tra le varie possibili traduzioni, quella più adatta dal punto di vista prettamente linguistico
- basati su un corpus: si basa sullo studio di campioni reali e delle traduzioni a loro associate;
- basati sul contesto: si sceglie la migliore traduzione tenendo in considerazione il discosto in qui la parola è inserita.

Un esempio di traduzione basata su corpus è GNMT (Google Neural Machine Translation), un sistema di riconoscimento end-to-end che usa le reti neurali per incrementare l'affidabilità di Google Translate e per ovviare ai problemi in cui incorrono come, ad esempio, la dispendiosità di cui soffrono e la difficoltà nel tradurre le parole usate raramente [27].

Veicoli autonomi

Negli ultimi tempi, compagnie come Google, Uber e Tesla stanno spingendo costantemente sulle "smart car" e sui piloti automatici. Quasi tutte le moderne macchine a pilota automatico utilizzano un algoritmo chiamato SLAM (Simultaneous Localization and Mapping) per la costruzione del contesto spaziale e un algoritmo di machine vision per la rilevazione degli oggetti reali. In particolare, l'estrazione dei dati dai diversi oggetti, può avvenire attraverso l'uso di reti neurali. Si cita, ad esempio, il lavoro svolto da NVIDIA che usa una rete neurale convoluzionale per il riconoscimento del percorso stradale e del controllo della vettura [28].

Arte

Il mondo artistico è incuriosito dal machine learning. A dimostrarlo è, per esempio, il progetto Magenta [29] portato avanti da Google in collaborazione con diversi artisti.
Magenta si basa su TensorFlow e si domanda in che modo è possibile utilizzare le tecnologie considerate per creare delle opere artistiche e, data la dinamicità dell'arte, in che modo variare col tempo i criteri che portano alla composizione dell'opera artistica.
D'altronde chi fa arte trova, nella vita reale, qualcosa che vale la pena approfondire e su cui porre attenzione, qualcosa che in certo qual modo lo sorprende.
Al MILA Lab di Montreal, Canada, è stato creato un modello (figura 5.3) in grado di "osservare" un'immagine, estrarre da essa, gli elementi in grado di attirare una maggiore attenzione e spiegarne il contenuto attraverso la generazione di una didascalia [30].

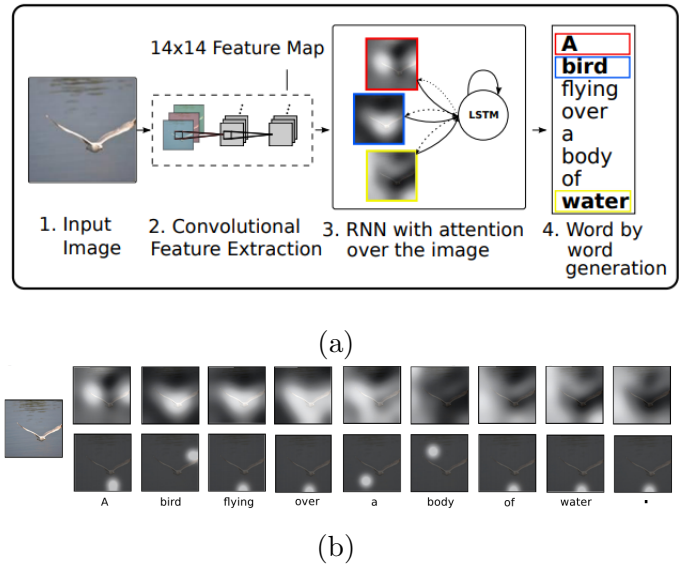


Figura 2.1: Processo di generazione di una didascalia usando come algoritmo una rete neurale (a). Esempio di attenzione che cambia nel tempo: ogni volta che una parola della didascalia viene creata, l'attenzione del modello si sposta su un'altra porzione dell'immagine (b)

L'obiettivo principale di Magenta è lo sviluppo di algoritmi capaci di creare arte e musica. DeepDream, sviluppato da A. Alexander Mordvintsev in Google, è un

esempio tangibile del lavoro che si sta svolgendo in questa direzione [31]. Anche in questo lavoro viene usato un algoritmo di rete neurale per il riconoscimento di oggetti all'interno di un'immagine. L'estrazione delle informazioni da alcuni layer della rete permette la creazione di nuova immagini sulla base di quella data in input all'algoritmo che diventa a tutti gli effetti un'opera artistica (figura 2.2).

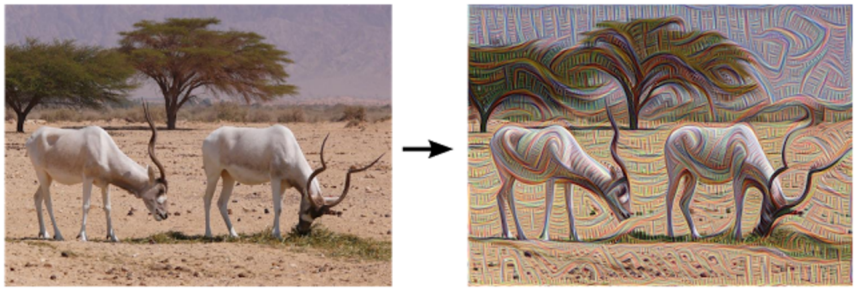


Figura 2.2: Immagine di input a sinistra. Immagine processato con deepDream a destra (fonte https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html)

2.2 Generazione di eventi attraverso l'uso di dati spazio-temporali nel calcio

Per la maggior parte delle società militanti nel calcio professionistico, le analisi delle prestazioni si integrano interamente col processo di allenamento. In questo contesto, molte analisi sono basate su eventi che vengono annotati manualmente e/o sulle informazioni dei giocatori. Questo lascia dei margini ampi di errore dettati dal fattore umano e dalla qualità dei dati forniti. Inoltre, la rilevazione manuale dei dati richiede un enorme spreco di risorse temporali e quindi economiche. In [7], è stata presentata una soluzione per l'estrazione automatica di eventi basati sui dati di tracking dei giocatori e della palla. Nella suddetta soluzione, i dati di tracking della palla vengono elaborati per la creazione di diverse feature, utili poi per la rilevazione di alcuni eventi frequenti nel gioco del calcio. È possibile scomporre il lavoro in questione in quattro diverse parti, che verranno analizzate nei dettagli nel seguito.

2.2.1 Caratteristiche dei dati

La qualità e l'accuratezza dei dati dipende fortemente dalla tecnologia di tracking utilizzata. Tipicamente, la collezione dei dati a disposizione consiste in informazioni di posizione dei giocatori e della palla e nella lista di eventi annotati manualmente così come, eventualmente, in una collezione di metadati riguardanti le squadre e i giocatori. In questo lavoro sono stati adoperati i dati di tracking delle partite della Bundesliga, massima serie del campionato calcistico tedesco. La rilevazione delle posizioni dei giocatori e della palla è stata fatta con una frequenza di 10 Hz. Il campo da gioco viene definito considerando il centro con coordinate (0,0) con una lunghezza di 105x68 metri (figura 2.3)

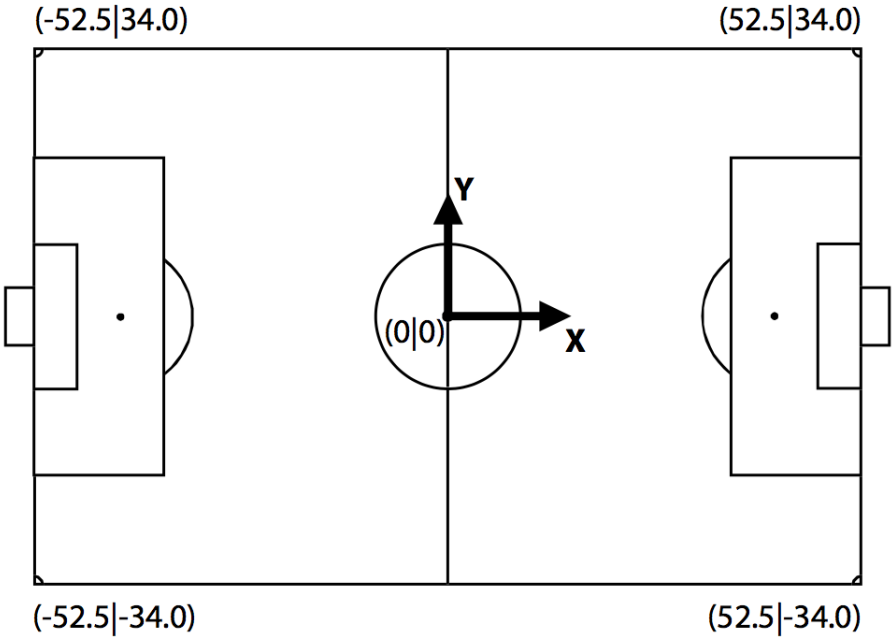


Figura 2.3: Dimensioni campo di calcio

2.2.2 Feature

In questa sezione vengono presentate le definizioni delle feature utilizzate partendo dai dati spazio temporali contenuti nel data set.
Analizzando ed elaborando questi dati è possibile definire gli eventi come una correlazione di feature che acquisiscono determinate caratteristiche allo scaturire dell'evento stesso.
La palla è sicuramente l'elemento centrale nel gioco del calcio. Tutti i giocatori interagiscono con essa durante una partita e, tra tutti gli elementi di cui si tiene traccia, è quello che possiede una variazione più significativa dei valori delle feature. Definendo o come l'oggetto da cui estrarre le features (in questo caso la palla), la posizione di o al tempo t è definita come $p(o,t)$. La direzione della palla al tempo t_1 è data, quindi, da:

$$d(o, t_1) = p(o, t_2) - p(o, t_1) \quad con \quad t_2 = t_1 + 1 \tag{2.1}$$

velocità

Viene usata la direzione per la determinazione della velocità della palla. Essa viene moltiplicata per 10 per avere come unità il $\frac{m}{s}$. La velocità v dell'oggetto o al tempo t è definita come:

$$v(o, t) = \mid d(o, t) \mid \times 10 \tag{2.2}$$

Accelerazione

L'accelerazione è calcolata come differenza di due consecutivi valori di velocità. Il suo risultato sarà già in $\frac{m}{s^2}$. Quindi, l'accelerazione di un oggetto o al tempo t_2 è definita come:

$$a(o,t_1) = a(o,t_2) - a(o,t_1) \quad con \quad t_2 = t_1 + 1 \tag{2.3}$$

Picchi di accelerazione

Data l'alta frequenza di campionamento, l'accelerazione potrebbe essere acquisita in due campioni successivi. Nel lavoro è stata proposta quindi un'aggregazione di due valori di accelerazione consecutivi per trovare l'accelerazione reale(2.4).
Il minimo e il massimo picco di accelerazione, a_min e a_max rispettivamente, di un oggetto o al tempo t_2 è definito come:

$$\begin{aligned} a_{max} &= \sum_{x\epsilon\{t_1,t_2\}} \max(0, a(o,x)) \quad con \quad t_2 = t_1 + 1 \\ a_{min} &= \sum_{x\epsilon\{t_1,t_2\}} \min(0, a(o,x)) \quad con \quad t_2 = t_1 + 1 \end{aligned} \tag{2.4}$$

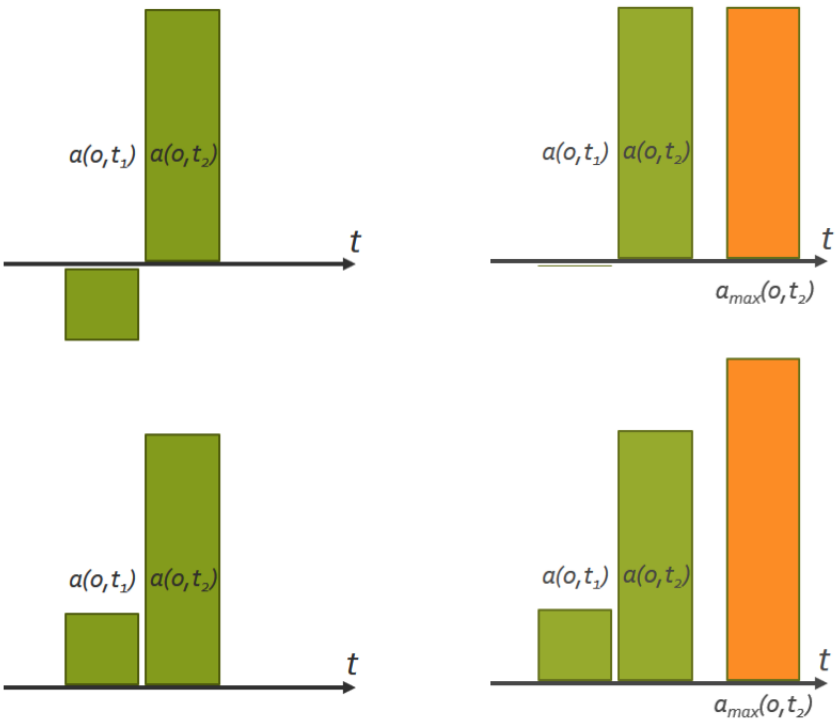


Figura 2.4: Esempio di aggregazione di due valori consecutivi di accelerazione

Per prevenire la rilevazione di due picchi di accelerazione consecutivi si è definito un picco, minimo o massimo, come reale se è soltanto non c'è nessun picco adiacente con un'accelerazione maggiore (figura 2.5).

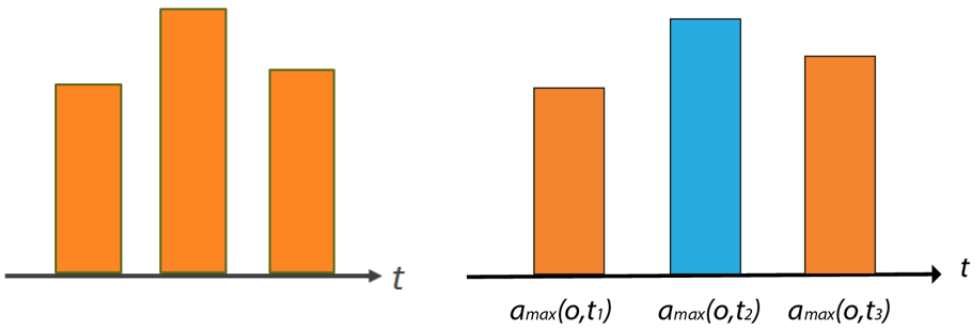


Figura 2.5: Esempio di calcolo di un picco d'accelerazione reale

Esso, quindi, è stato calcolato come:

$$a_{max_{real}} = \begin{cases} a_{max}(o,t_2) & if \quad a_{max}(o,t_2) > a_{max}(o,t_1) \wedge a_{max}(o,t_2) > a_{max}(o,t_3) \\ & con \quad t_1 + 1 = t_2 = t_3 - 1 \\ 0 & altrimenti \end{cases}$$
$$a_{min_{real}} = \begin{cases} a_{min}(o,t_2) & if \quad a_{min}(o,t_2) > a_{min}(o,t_1) \wedge a_{min}(o,t_2) > a_{min}(o,t_3) \\ & con \quad t_1 + 1 = t_2 = t_3 - 1 \\ 0 & altrimenti \end{cases} \tag{2.5}$$

Cambio di direzione

Il cambio di direzione è importante poiché nei movimenti rapidi tende ad assumere un valore elevato. In figura 2.6 è possibile vedere come esso è stato calcolato.

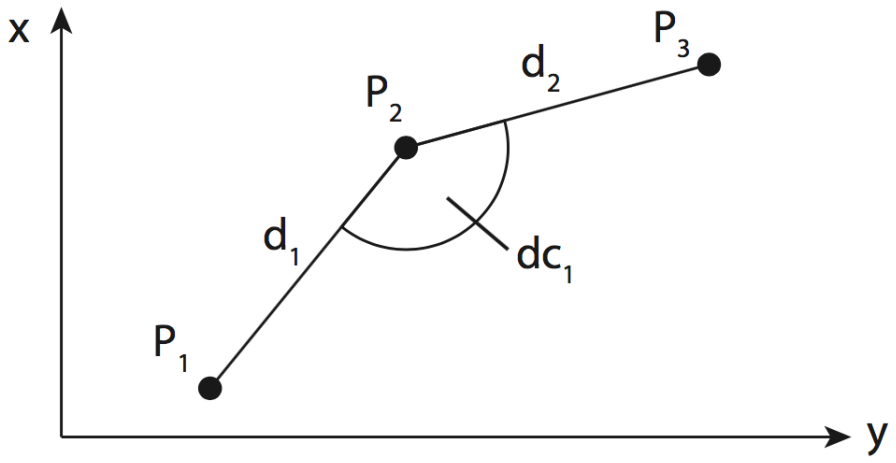


Figura 2.6: Cambio di direzione dell' oggetto

Siano $P_1 = p(o, t_1)$, $P_2 = p(o, t_2)$ e$P_3 = p(o, t_3)$ tre posizioni consecutive di un oggetto o e siano $d_1 = d(o, t_1)$ e $d_2 = d(o, t_2)$ i due vettori direzione. L'angolo creato tra d_1 e d_2 è il cambio di direzione di un oggetto o al tempo t_2 ed è definito come la funzione arcoseno applicata al quoziente del prodotto scalare dei due vettori direzione e il prodotto tra le lunghezze dei due vettori direzione:

$$cd(o, t_2) = \arcsin(\frac{d_1(o, t_1) \times d_2(o, t_2)}{\mid d_1(o, t_1) \mid \times \mid d_2(o, t_2) \mid}) \tag{2.6}$$

Distanza dal target

La scelta del target di riferimento è stata fatta tenendo in considerazione solo il movimento orizzontale della palla. Così facendo il calcolo della feature risulta indipendente dalla sua posizione. Se la palla si muove verso sinistra, allora la porta sinistra è assegnata come target, viceversa sarà assegnata la porta destra. In figura 2.7 è possibile vedere P_1, P_2, P_3 e P_4, quattro differenti posizioni dell'oggetto o, e T_1 e T_2 che sono, invece, le posizioni dei due target. Le frecce, ad ogni posizione, approssimano la direzione assunta dall'oggetto al tempo in cui è stata rilevata la posizione stessa. Per P_1 e P_2 il movimento orizzontale risulta positivo ed il target selezionato è quello a destra mentre per P_3 il movimento orizzontale risulta negativo ed il target selezionato è quello a sinistra. P_4, invece, ha movimento orizzontale nullo ed in questo caso non è possibile selezionare un target.

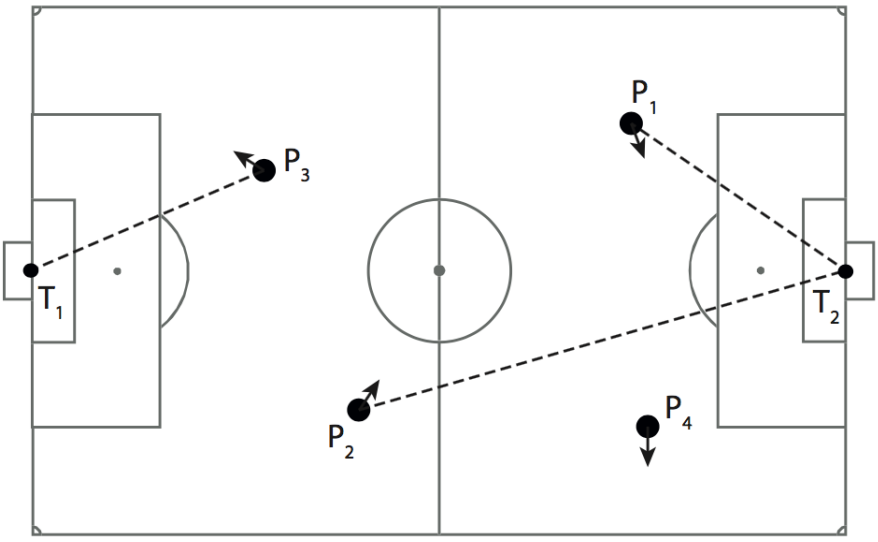


Figura 2.7: possibili posizioni della palla e relativa distanza dal target

Cross on target line

Questa feature informa sulla distanza che la palla ha dal target di riferimento quando attraversa la rispettiva linea di fondo. La figura 2.8 mostra una possibile posizione dell'oggetto o e la sua direzione $d_1 = d(o, t)$. Se l'oggetto o continuasse il suo movimento senza cambiare la sua direzione, attraverserebbe la linea di fondo alla posizione C_1. La distanza tra il target T_2 e C_1 è la misura di questa feature.

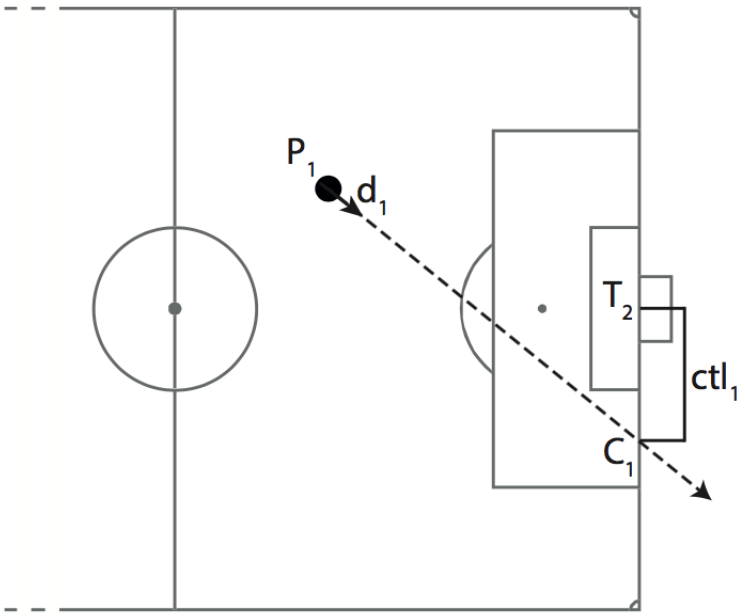


Figura 2.8: Cross on target line

La feature, quindi, è definita come:

$$ctr(o,t) = p_2(o,t) + d_2(o,t) \times \frac{g_1(o,t) - p_1(o,t)}{d_1(o,t)} \tag{2.7}$$

dove $p_1(o,t)$ e $p_2(o,t)$ rappresentano rispettivamente la componente orizzontale e verticale di $p(o,t)$, posizione dell'oggetto o al tempo t, mentre $d_1(o,t)$ e $d_2(o,t)$ rappresentano rispettivamente la componente orizzontale e verticale della direzione $d_(o,t)$ dell'oggetto o al tempo t. La variabile $g_1(o,t)$ è invece la componente orizzontale della posizione del target selezionato $g_(o,t)$

2.2.3 Rilevazione degli errori

Per la classificazione degli eventi, sono state sperimentate le prestazioni di tre differenti algoritmi di machine learning:

- Support Vector Machine (SVM);
- K-nearest neighbors (KNN);
- Random Forest (RForest).

Support Vector Machine

L'algoritmo SVM è un algoritmo di classificazione sviluppato meglio anni '90 da Vladimir Vapnik. Esso prova a dividere i dati in modo da massimizzare la distanza delle classi, chiamata anche margine geometrico.
La sua applicazione iniziale era stata pensata per casi binaria, casi in cui bisogna

classificare solamente due classi, ma oggi esistono algoritmi SVM in grado di operare, anche, in contesti di classificazione mutilasse.
Considerando un caso binario e supponendo che il data set di training $\{x_1, ..., x_n\}$ sia formato da dati di uno spazio $X \subseteq R^d$ e che le loro etichette siano $\{y_1, ..., y_n\}$ con $y_i \in \{1, -1\}$, allora l'algoritmo SVM più semplice suddividerà il data set cercando il margine massimo.
L'etichetta assegnata ad un dato, in questo caso 1 o -1, dipenderà dalla parte in cui questo ricadrà rispetto alla retta che separa lo spazio (figura 2.9) [32].

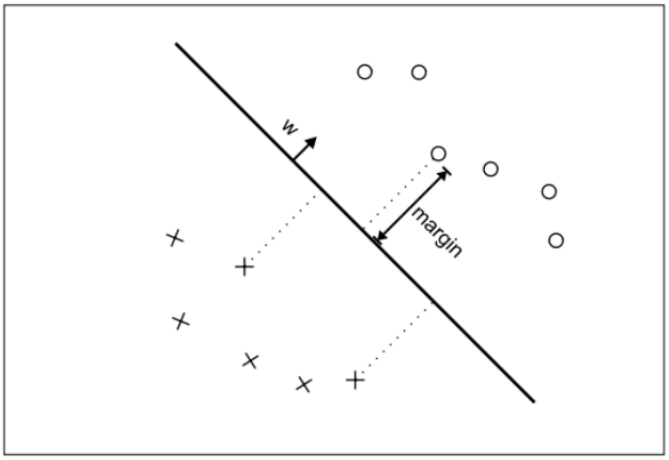


Figura 2.9: Esempio di SVM lineare

Esistono poi altri tipi di algoritmo di SVM che usano funzioni non lineari per la suddivisione dello spazio. In questo lavoro è stato utilizzato un algoritmo SVM lineare.

K-nearest neighbor

Il K-Nearest-Neighbors assegna ad un dato del data set non ancora classificato la stessa etichetta dei dati a lui più vicino tra quelli già classificati [33]. Esso richiede in ingresso tre informazioni:

- il data set;
- il tipo di metrica da usare per calcolare la distanza (per esempio la distanza euclidea);
- K, il numero massimo di dati già etichettati da considerare per l'assegnazione della nuova etichetta (figura 2.10).

Avendo il valore di K, quindi, l'algoritmo opera in questo modo per trovare i vicini e classificare il nuovo dato:

- calcola la distanza dagli altri dati;
- identifica i K dati più vicini;
- usa l'etichette dei vicini per assegnare la nuova etichetta al dato in considerazione (ad esempio tramite maggiornza).

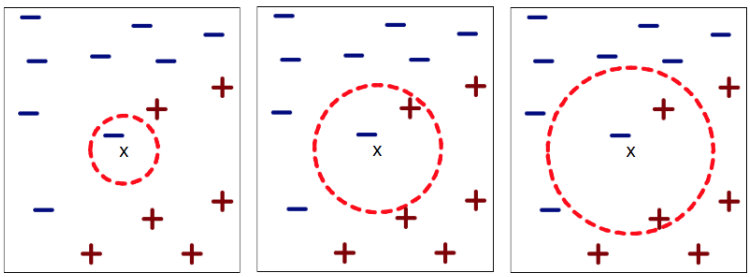


Figura 2.10: I K vicini di un dato x sono i quelli con aventi le K distanza minori da x (fonte [18])

È stato usato, qui, un algoritmo KNN con K=3;

Random Forest

L'algoritmo Random Forest consiste nell'applicazione di più alberi decisionali, al fine di un controllo maggiore dell'overfitting. Per classificare un nuovo vettore del data set non ancora dotato di etichetta si utilizzano diversi alberi decisionali ognuno dei quali "vota" per una classe. L'algoritmo attribuisce al vettore la classe che ha ottenuto il maggior numero di votazioni [34] . In questo articolo si è scelto un numero di alberi decisionali pari a 10.

2.2.4 Risultati

In questa quarta ed ultima sezione sono stati analizzati i risultati ottenuti dagli algoritmi di machine learning. La valutazione della loro qualità si è basata sulle metriche di Precision e Recall di ogni algoritmo. Per la valutazione dettagliata dei risultati, si sono analizzati solo gli eventi di passaggio e ricezione, ed è stata considerata come ulteriore metrica l'F-Measure. In tabella 1 vengono mostrati i risultati ottenuti dall'applicazione dei tre algoritmi.

Algorithm	Event Type	Precision	Recall	F-Measure
KNN	pass	0.320	0.088	0.138
	reception	0.000	0.000	0.000
	ball touch	0.438	0.095	0.156
SVM	pass	0.387	0.641	0.483
	reception	0.059	0.642	0.108
	ball touch	0.305	0.716	0.428
RForest	pass	0.354	0.767	0.484
	reception	0.089	0.436	0.148
	ball touch	0.351	0.595	0.442

Figura 2.11: Risultati ottenuti con gli algoritmi considerat

Dato l'utilizzo di tre diversi algoritmi, gli autori del lavoro hanno pensato ad un'aggregazione dei risultati così da incrementare l'accuratezza. In particolare, hanno scelto tre diversi valori di peso, uno per ogni algoritmo, e un punteggio minimo per ogni evento. Quest'ultimo è stato sommato a quello dato da un classificatore ad ogni predizione fatta. In questo modo soltanto gli eventi con un punteggio maggiore o uguale a quello minimo sono stati effettivamente considerati come predetti. I valori dei pesi usati sono stati 0.4 per SVM e RForest e 0.2 per KNN. Per ogni tipo di evento, invece, il punteggio minimo da ottenere è stato fissato a 0.8.

Capitolo 3

Tecnologie

In questo capitolo viene fatto un excursus sulle tecnologie hardware e software utilizzate per lo sviluppo della tesi.

3.1 Hardware

Per la creazione dell'animazione del passaggio si è usata la tecnica del motion capture sfruttando la tuta Perception Neuron [35].

3.1.1 Perception Neuron

Perception Neuron è una tuta di motion capture realizzata grazie ad una campagna Kickstarter. Per la rilevazione dei movimenti vengono utilizzati 32 sensori chiamati "internal measurement units" oltre ad un magnetometro, giroscopio e accelerometro (figura 3.1). Questa tuta può essere applicata in diversi contesti come VFX, Virtual Reality, Stage Performance e Analisi sportive.

Figura 3.1: Dotazione della tuta Perception neuron
(fonte:https://www.kickstarter.com/projects/1663270989/project-perception-neuron)

3.2 Software

In questa lavoro di tesi è stato usato Unity 3D per la creazione dell'applicazione di Realtà Virtuale.
Sia per la logica dell'applicazione, sia per l'estrapolazione dei dati dal data set è stato utilizzato Visual Studio Code ed il linguaggio C#.
Infine, per il machine learning è stato utilizzato RapidMiner.

3.2.1 Visual Studio Code

Visual Studio Code è un editor che può essere installato su Windows, macOS e Linux ed è basato sul framework open source Electron [36]. Di default, include il supporto a JavaScript, TypeScript and Node.js e permette di aggiungere estensioni per altri linguaggi (come C++, C#, Python, PHP, Go) e runtime (come .NET e Unity). Visual Studio Code adotta una classica interfaccia con un file explorer sulla sinistra , che mostra tutti i file in cui è possibile accedere, e un editor di testo sulla destra che mostra il contenuto dei file che sono stati aperti (figura X) [37]. In questo lavoro si è fatto uso di Visual Studio Code per la generazione delle featur ed il pruning del data set.

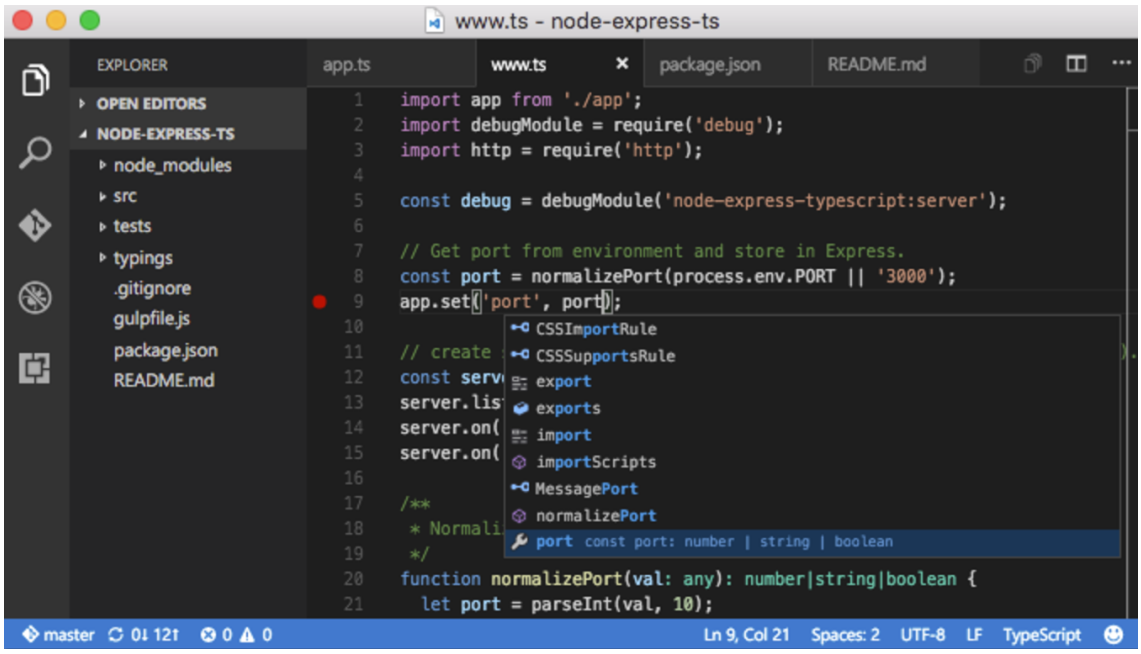


Figura 3.2: Interfaccia di Visual Studio Code con il file explorer a sinistra e l'editor a destra

3.2.2 Unity 3D

Unity 3D, o semplicemente Unity, è un game engine multipiattaforma per lo sviluppo di videogiochi e di altre applicazioni interattive 3D. Con esso è possibile creare applicazioni ed esportarle per piattaforme di sistemi desktop e mobile. Insieme a Unity viene anche utilizzato Monodevelop, editor utilizzato per la gestione della parte logica dell'applicazione. Nonostante non sia, tra i game engine, quello che spicca maggiormente in qualità di rendering e prestazioni, la sua flessibilità lo rende comunque in grado di competere con i principali concorrenti come UDK e Cry Engine (figura 3.3).
Unity è stato utilizzato per lo sviluppo dell'applicazione di Realtà Virtuale e per la generazione delle animazioni predette dall'algoritmo di machine learning.

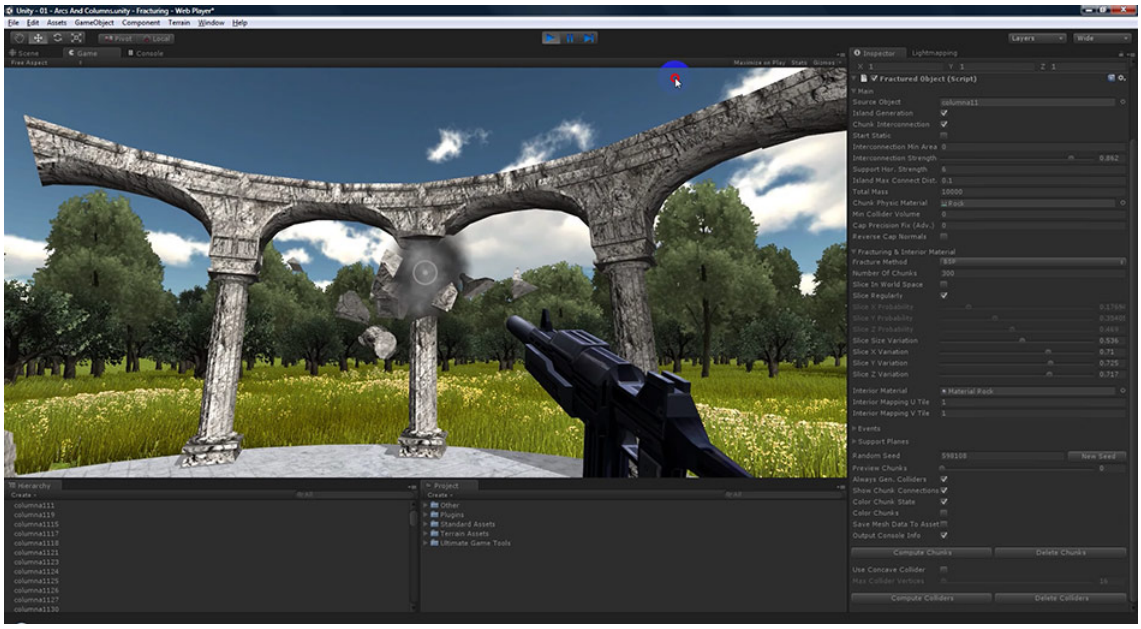


Figura 3.3: Interfaccia di Unity 3D in modalità game. In questo modalità è possibile provare l'applicazione in fase di sviluppo; la costruzione dell'ambiente 3D, invece, viene realizzata passando dalla modalità game alla modalità scene

3.2.3 RipidMiner

RapidMiner è una piattaforma di data mining e analisi di predizioni che permette di analizzare in modo avanzato i dati. Questo software, scritto in Java, offre gli strumenti per varie operazioni di machine learning, quali:

- caricamento e trasformazione dei dati;
- elaborazione e visualizzazione dei dati;
- analisi predittiva e modellazione statistica;
- valutazione.

È possibile impiegare la piattaforma attraverso un'interfaccia per la creazione del flusso di operazioni da attuare sui dati, che in RapidMiner viene chiamato "processo". Un processo è formato da un insieme di "operatori" che vengono rappresentati, graficamente come dei nodi e che compiono una singola azione sui dati. Ogni operazione svolta da un operatore emette un risultato di output che diventa l'input del prossimo, nella catena degli operatori. (figura 3.4).

Alternativamente, è possibile usare RapidMiner sotto forma di API e richiamare le funzioni attraverso la linea di comando.
È possibile estendere le sue funzionalità scrivendo degli script in R o in Python o con plugins addizionali scaricabili dal proprio repository. Questo software è stato impiegato nella fase di machine learning per l'apprendimento dell'algoritmo di classificazione.

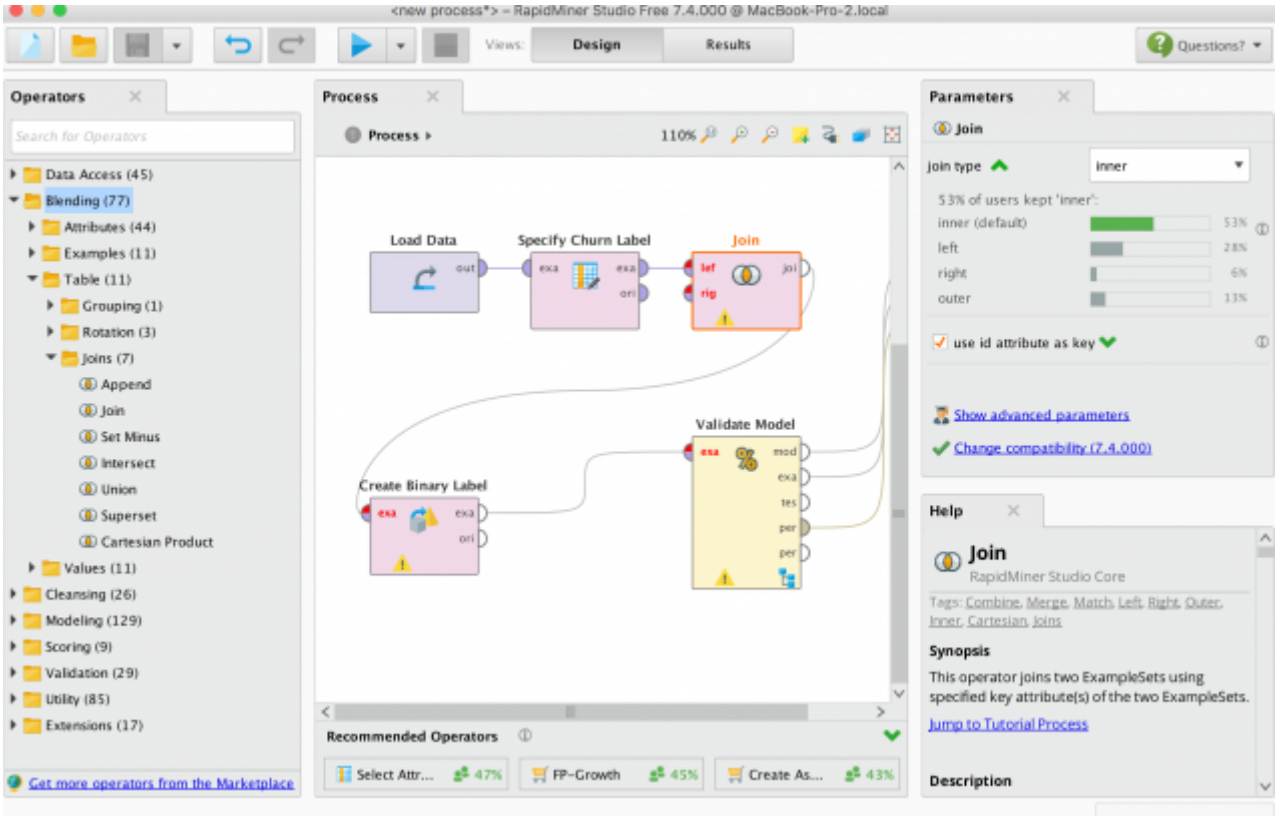


Figura 3.4: Interfaccia di RapidMiner ed esempio di un un processo. Ogni nodo è un operatore che svolge delle operazioni diverse sui dati

Capitolo 4

Progettazione e realizzazione

4.1 I dati

Per lo sviluppo di questo lavoro di tesi sono stati usati i dati di tracking della partita NBA tra San Antonio Spurs e Minnesota. Essi sono stati acquisiti con una frequenza di campionamento F_c pari a 20 Hz.
Il file con i dati, in formato JSON, è stato convertito in formato CSV usando uno script Python [6].
Le informazioni contenute in esso, per ogni oggetto, palla o giocatore per cui è effettuato il tracking sono riportate di seguito:

- $team_{id}$, identificativo della squadra a cui appartiene l'oggetto, –1 se l'oggetto di cui è effettuato il tracking è la palla;
- $player_{id}$:, identificativo dell'oggetto, –1 se l'oggetto di cui è effettuato il tracking è la palla;
- x_{loc}, posizione lungo l'asse orizzontale del campo dell'oggetto di cui è effettuato il tracking;
- y_{loc}, posizione lungo l'asse ortogonale agli altri due assi del campo dell'oggetto di cui è effettuato il tracking;
- z_{loc}, posizione lungo l'asse verticale del campo dell'oggetto di cui è effettuato il tracking (solo per la palla);
- $game_{clock}$, tempo rimasto nel "game";
- $shot_{clock}$, tempo rimasto nello "shot clock";
- $quarter$, quarto della partita;
- $game_{id}$, identificativo della partita;
- $event_{id}$, rappresenta il numero identificativo di una specifica azione.

4.2 Le feature

In questa sezione viene data una spiegazione per ogni feature che è stata creata. Per la creazione delle suddette feature sono stati utilizzati i dati di tracking che sono stati illustrati nella sezione precedente.
Per il loro calcolo è stato utilizzato Visual Studio Code con c#. Le feature che contengono la dicitura 2D all'interno del loro nome non tengono in considerazione la componente relativa all'altezza. Le altre feature, che invece contengono la dicitura 3D all'interno del loro nome, sono state calcolate in tre dimensioni (tenendo conto, quindi, dell'altezza).
Si definisce o l'oggetto per cui sono state calcolate le feature (ovvero la palla), e t come l'inverso della frequenza di campionamento F_c :

$$t = \frac{1}{F_c} = \frac{1}{20} \tag{4.1}$$

Velocità 2D

Per il calcolo della velocità si è usata la posizione dell'oggetto o, $p_1 = (p_{1_x}, p_{1_y})$, al tempo t_1 e la posizione dell'oggetto o, $p_2 = (p_{2_x}, p_{2_y})$, al tempo successivo t_2. Il tempo t necessario per percorrere lo spazio è dato dalla differenza dei due valori di $game_{clock}$ relativi alle due posizioni. La velocità 2D, Vel_{2D}, è quindi definita come:

$$Vel_{2D}(o, t_1) = \frac{p_2 - p_1}{t} \tag{4.2}$$

Accelerazione 2D

Con la velocità 2D di o al tempo t_1, Vel_{2D_1}, e la velocità 2D al tempo t_2 , Vel_{2D_2}, è possibile calcolare l'accelerazione 2D istantanea. L'accelerazione acc2D è scritta come::

$$Acc_{2D}(o, t_1) = \frac{v_{2d_2} - v_{2d_1}}{t} \tag{4.3}$$

Picchi di accelerazione 2D

Questa feature rileva i picchi di accelerazione 2D massimi e minimi e conferisce il valore 0 a quelli che non sono né un massimo né un minimo.
Sia $a_{2d}(o, x)$ l'accelerazione 2D dell'oggetto o. Il picco di accelerazione 2D massimo e minimo dell'oggetto o al tempo t_2 sono definiti come:

$$\begin{aligned} a_{max} &= \sum_{x \epsilon \{t_1, t_2\}} \max(0, a(o, x)) \quad con \quad t_2 = t_1 + 1 \\ a_{min} &= \sum_{x \epsilon \{t_1, t_2\}} \min(0, a(o, x)) \quad con \quad t_2 = t_1 + 1 \end{aligned} \tag{4.4}$$

Si è evitato, data l'alta frequenza di campionamento, che due picchi venissero rilevati in due istanti consecutivi. Per tutti i picchi, quindi, sia massimi che minimi, si è controllato che il valore precedente e quello successivo non siano maggiori. In tal caso al picco è stato dato un valore nullo, non essendo né un massimo, né un minimo. Un picco reale di accelerazione dell'oggetto o al tempo t_2 è definito come:

$$a_{max_{real}} = \begin{cases} a_{max}(o,t_2) & if \quad a_{max}(o,t_2) > a_{max}(o,t_1) \wedge a_{max}(o,t_2) > a_{max}(o,t_3) \\ & con \quad t_1 + 1 = t_2 = t_3 - 1 \\ 0 & altrimenti \end{cases}$$

$$a_{min_{real}} = \begin{cases} a_{min}(o,t_2) & if \quad a_{min}(o,t_2) > a_{min}(o,t_1) \wedge a_{min}(o,t_2) > a_{min}(o,t_3) \\ & con \quad t_1 + 1 = t_2 = t_3 - 1 \\ 0 & altrimenti \end{cases} \tag{4.5}$$

Cambio di direzione 2D

Date le tre posizioni $P_1 = (p_{1_x}, p_{1_y})$, $P_2 = (p_{2_x}, p_{2_y})$, $P_3 = (p_{3_x}, p_{3_y})$ e le due direzioni $d_1 = P_2 - P_1$ e $d_2 = P_3 - P_2$ per calcolare il cambio di direzione, si è applicato l'arcoseno sul quoziente del prodotto scalare tra d_1 e d_2 e del prodotto tra la lunghezza di d_1 e quella di d_2. Il cambio direzione 2D dell'oggetto o al tempo t_2 è quindi:

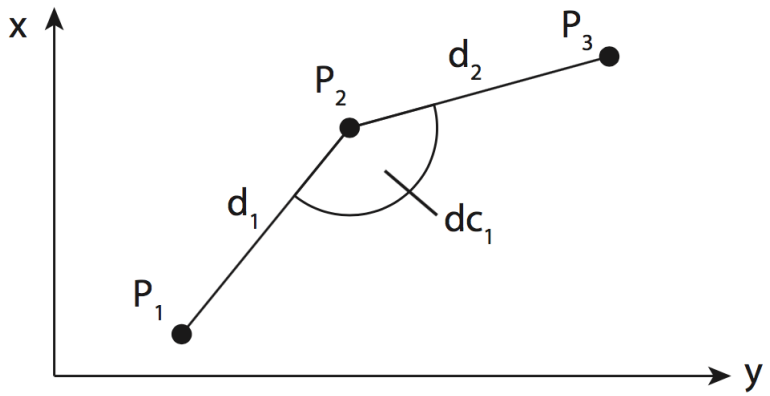


Figura 4.1: Cambio di direzione dell' oggetto

$$cd_{2d}(o,t_2) = \arcsin\left(\frac{d_1(o,t_1) \times d_2(o,t_2)}{\mid d_1(o,t_1) \mid \times \mid d_2(o,t_2) \mid}\right) \tag{4.6}$$

Distanza dal target 2D

L'obiettivo dei giocatori è quello di porre la palla in uno dei due canestri. Essi sono quindi i target verso cui si muoverà la palla durante la partita. Per calcolare questa future si è tenuta in considerazione solo la componente orizzontale (x) della palla, tralasciando la componente verticale (y). Se la palla si muove verso sinistra, allora il target sarà il canestro sinistro, viceversa, se si muove verso destra, il canestro a destra sarà il target di riferimento.

Prese le componenti orizzontali della posizione in due istanti successivi $P_1 = (p_{1_x}, p_{1_y})$ a tempo t_1 e $P_2 = (p_{2_x}, p_{2_y})$ al tempo t_2, la direzione della palla sarà $d(o,t_1) = P_2 - P_1$:

- se $d_x(o,t_1) > 0$, la palla si muove verso destra, quindi movimento orizzontale positivo;

- se $d_x(o, t_1) < 0$, la palla si muove verso sinistra, quindi movimento orizzontale negativo.

La figura 4.2 mostra alcuni alcune situazioni possibili e la distanza dal target. P_1, P_2, P_3 e P_4 rappresentano le posizioni bidimensionali della palla in quattro istanti diversi. La freccia ad ogni posizione indica la sua direzione nel medesimo istante. T_1 e T_2 rappresentano i due target. Come è possibile notare, P_1 e P_2 hanno un movimento orizzontale positivo, quindi il corrispondente target sarà il target T_2. Viceversa, P_3 ha un movimento orizzontale negativo, quindi, spostandosi verso sinistra, il suo target di riferimento sarà T_1. Nel casi di P_4 non c'è componente orizzontale nella direzione ($d_x(o, t_1) = 0$). Nessun target è, di conseguenza, assegnabile.

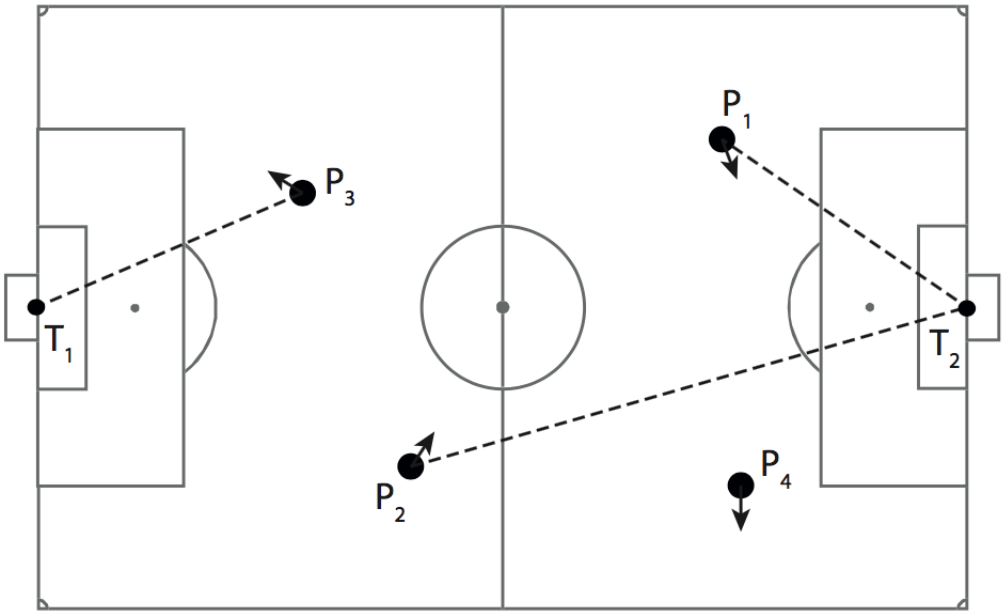


Figura 4.2: possibili posizioni della palla e relativa distanza dal target

Siano $p_1 = (p_{1_x}, p_{1_y})$ e $p_t = (p_{t_x}, p_{t_y})$ le posizioni bidimensionali di, rispettivamente, palla e target selezionato al tempo t, la distanza dal target è allora la distanza euclidea:

$$Dist_{2D}toBasketTarget = |p_1 - p_t| \tag{4.7}$$

Cross on target line orizzontale

Come si è discusso precedentemente, la palla si sposta alternativamente verso uno dei due target. Oltre alla distanza da questo, un'altra feature è la prossimità del suo movimento rispetto al target. Si è definita questa feature come la distanza tra il target e la palla quando questa, assumendo che manterrà la stessa traiettoria, attraversa la corrispettiva linea di fondo.

Il cross on target orizzontale non tiene conto dell'altezza, essa è calcolata sulla x e sulla z. La figura 4.3 mostra la posizione $P_1 = p(o, t_1) = (p_{1_x}, p_{1_y})$ dell'oggetto o e la sua direzione $d_1 = d(o, t_1) = (d_{1_x}, d_{1_y})$. Se l'oggetto continuasse il movimento

senza cambiare la direzione, attraverserebbe la linea di fondo alla posizione C1. La distanza tra il target $t_=(t_x, t_y, t_z)$ e C_1 è la misura di questa feature.

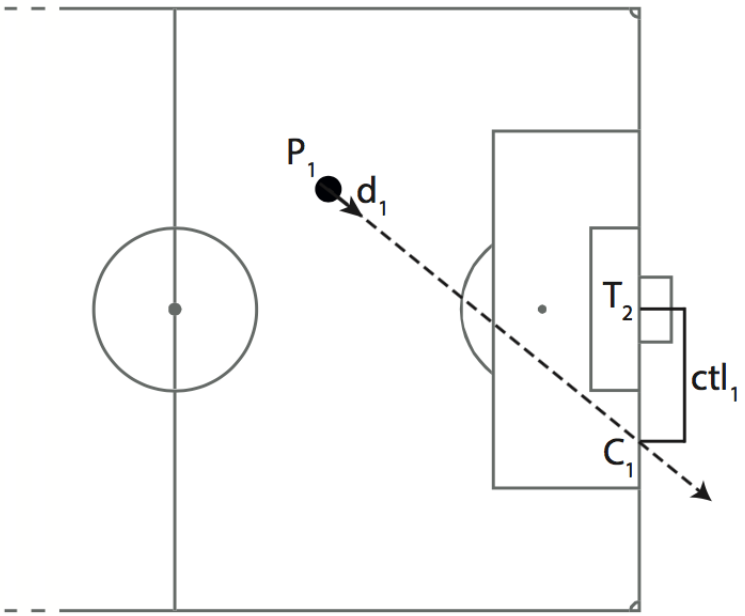


Figura 4.3: Cross on target line

La feature è calcolata come:

$$ctr(o,t) = p_{1_y} + d_y \times \frac{(t_x - p_{1_x})}{d_{1_x}} \tag{4.8}$$

Quelle esposte fino a questo punto sono le feature utilizzate nell'articolo [7], adattate al formato dei dati disponibili nel caso considerato.
Si presentano, ora, le nuove feature analizzate in questo lavoro di tesi, in parte derivate da considerazioni specifiche per lo sport in esame.

Velocità 3D

Data la posizione dell'oggetto o al tempo t_1, $p_1 = p(o,t_1) = (p_{1_x}, p_{1_y}, p_{1_z})$, e la posizione dell'oggetto o al tempo t_2, $p_2 = p(o,t_2) = (p_{2_x}, p_{2_y}, p_{2_z})$, la velocità 3D, Vel_{3D}, è definita come:

$$Vel_{3D}(o,t_1) = \frac{(p2 - p1)}{t} \tag{4.9}$$

Accelerazione 3D

L'accelerazione acc_{3D} è la seguente:

$$Acc_{3D}(o,t_1) = \frac{Vel_{3D_2} - Vel_{3D_1}}{t} \tag{4.10}$$

dove Vel_{3D_1} e Vel_{3D_2} sono rispettivamente la velocità 3D di o al tempo t_1 e t_2.

Picchi di accelerazione 3D

In questa feature si è passati dal 2D al 3D sfruttando l'accelerazione 3D invece dell'accelerazione 2D. I calcoli matematici non differiscono rispetto alla feature dei picchi di accelerazione 2D.

Distanza dal target 3D

Anche in questo caso, come per la feature 2D, si è utilizzata la componente orizzontale per la rilevazione del target. Per il calcolo effettivo della distanza si sono sostitute le posizioni bidimensionali di palla e target con quelle tridimensionali. Dati $p_1 = (p_{1_x}, p_{1_y}, p_{1_z})$ e $t_=(t_x, t_y, t_z)$ e C_1 la distanza dal target 3D Dt_{3D} è data da:

$$Dt_{3D}toBasketTarget = |p_1 - t| \tag{4.11}$$

Cross on target line verticale

Per il calcolo di questa feature si è tenuto conto della traiettoria parabolica della palla. Si è scomposta la velocità 3D nelle componenti orizzontale, V_x, e verticale, V_z. Si è quindi calcolato il tempo, t_{line} che, data la posizione di partenza $P_1 = p(o, t_1) = (p_{1_x}, p_{1_y}, p_{1_z})$, è necessario per raggiungere la linea di fondo del target $t_=(t_x, t_y, t_z)$ verso cui la palla avanza.

$$t_{line} = \frac{t_x - p_{1_x}}{V_x} \tag{4.12}$$

Se g è la forza di gravità, la differenza tra l'altezza che avrebbe la palla sulla linea di fondo e l'altezza del target è la misura di questa feature (figura 4.4).

Si definisce l'altezza della palla sulla linea di fondo campo attraverso

$$Z = -\frac{1}{2} \times g \times t_{line}^2 + V_z \times t_{line} + p_{1_z} \tag{4.13}$$

Allora, la feature è calcolata come:

$$Ctr_{vertical} = -\frac{1}{2} \times g \times t_{line}^2 + V_z \times t_{line} + p_{1_z} \tag{4.14}$$

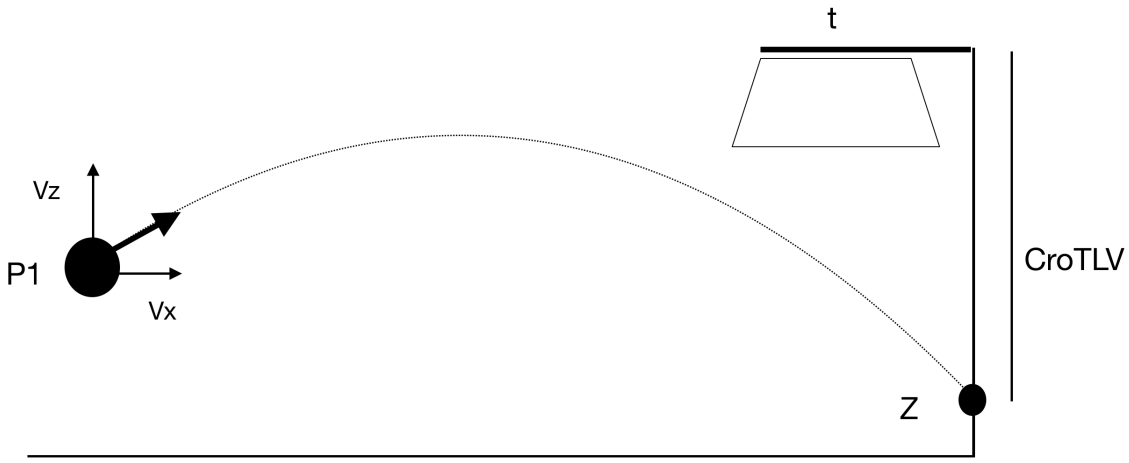


Figura 4.4: Cross on Target vine verticale

Distanza dal giocatore più vicino

La feature DfPC altro non è che la distanza dal giocatore più vicino alla palla al tempo t_1. Sia $P_1 = p(o, t_1)$ la posizione dell'oggetto, e quindi della palla al tempo t_1, essa è definita come la distanza minima tra ogni giocatore e la palla (Figura 4.5).

$$DistfPC = min(dist(p1, pgi)) \tag{4.15}$$

dove pgi è la posizione di ogni giocatore in campo.

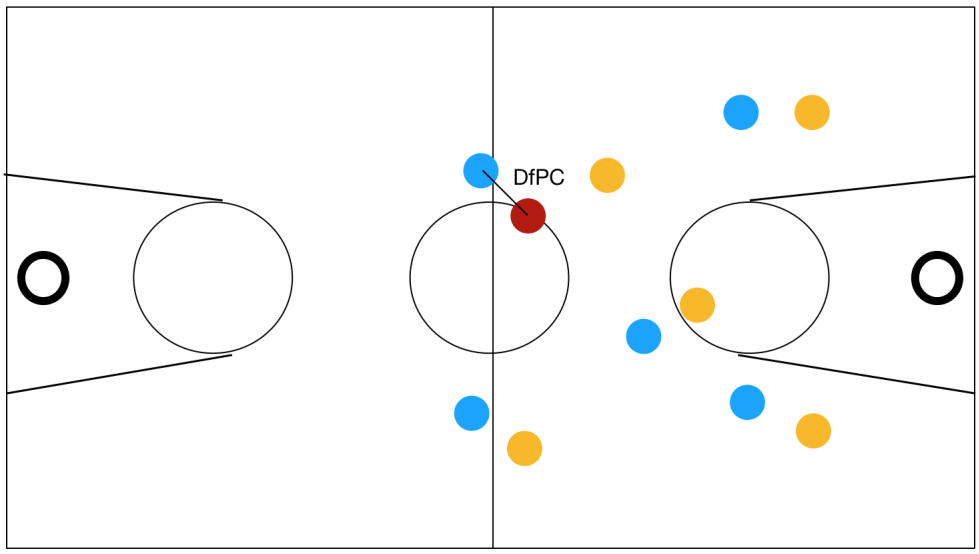


Figura 4.5: Distanza dal giocatore più vicino, DistanceFromPlayerCloser

TeamPlayerCloser

Questa features, $TeamPC$, rappresenta una stringa che identifica la squadra del giocatore più vicino alla palla.

Velocità verticale

La velocità verticale calcola la velocità tenendo in considerazione solamente la componente verticale della posizione della palla. Si tratta di una feature che considera caratteristiche specifiche del basket.
In particolare questa feature accentua i momenti in cui la palla si trova in possesso di un giocatore atto a muoversi palleggiando. In queste situazioni, infatti, i valori della velocità verticale sono più alti rispetto a quando si è in una fase di passaggio dove la velocità verticale si avvicina allo zero, dato che spesso la palla tende ad essere costante in altezza durante la fase di passaggio. Data la posizione verticale p_{y_1} al tempo t_1 e la posizione verticale p_{y_2} al tempo t_2, la velocità verticale V_y è calcolata come:

$$V_y = \frac{p_{y_2} - p_{y_1}}{t_2 - t_1} \tag{4.16}$$

Accelerazione verticale

L'accelerazione verticale si basa sulla velocità verticale V_{y_1} calcolata al tempo t_1 e sulla velocità verticale V_{y_2} calcolata al tempo t_2. Essa è definita come:

$$A_y = \frac{V_{y_2} - V_{y_1}}{t_2 - t_1} \tag{4.17}$$

La caretteristica di questa feature è quella di essere molto alta durante l'esecuzione del palleggio e molto bassa durante la fase di passaggio rafforzando così la distinzione tra i due differenti momenti della partita.

Picco di accelerazione verticale

Il picco di accelerazione verticale usa la stessa matematica applicata al calcolo rilevazione dei picchi negli altri due casi, quello bidimensionale e quello tridimensionale I valori dei picchi, in questo caso, derivano dall'accelerazione verticale piuttosto che dall'accelerazione 2D o 3D. Essi evidenziano un impulso verticale dato alla palla. Questo capita soprattutto durante la fase di tiro a canestro o schiacciata, mentre, in un passaggio, l'impulso verticale tende ad annullarsi a favore delle componenti orizzontali.

Medie e varianze

Analizzando i video della partita di basket che si è usata per l'annotazione, si è visto che il tempo medio di un passaggio è di circa un secondo. Le seguenti feature, quindi, sono state calcolate tenendo in considerazione una finestra temporale pari a un secondo e calcolando, all'interno di essa, la media e la varianza di alcune delle feature esposte nelle sezione precedenti. In particolare, esse sono:

- cambio di direzione 2D;

- distanza dal giocatore più vicino;
- posizione verticale p_y (componente verticale della posizione $p(o,t) = (p_x, p_y, p_z)$ della palla);
- velocità verticale;
- accelerazione verticale;

Data la frequenza di campionamento di 20Hz dei dati di tracking la suddetta finestra è composta da 20 campioni.
Alle feature che tengono conto del secondo precedente e quello successivo si è aggiunto un suffisso ad indicare la finestra temporale analizzata.
Esse sono nominate, rispettivamente, con:

- $nomefeatures_{Before}$;
- $nomefeatures_{After}$;

A questi nomi sono stati aggiunti due differenti prefissi ad indicare il tipo di misura effettuata:

- avg, misurano la media nell'arco di tempo uguale alla finestra utilizzata;
- var, misurano la varianza nell'arco di tempo uguale alla finestra utilizzata.

Si sono quindi calcolate le seguenti medie:

- $avg - cd_{2d} - after$;
- $avg - cd_{2d} - Before$;
- $avg - DfPC - After$;
- $avg - DfPC - Before$;
- $avg - p_y - After$;
- $avg - p_y - Before$;
- $avg - V_y - After$;
- $avg - V_y - Before$;
- $avg - A_y - After$;
- $avg - A_y - Before$.

Quelle che riguardano le varianze, invece, sono:

- $var - cd_{2d} - after$;
- $var - cd_{2d} - Before$;
- $var - DfPC - After$;

- $var - DfPC - Before$;
- $var - p_y - After$;
- $var - p_y - Before$;
- $var - V_y - After$;
- $var - V_y - Before$;
- $var - A_y - After$;
- $var - A_y - Before$.

4.3 Le annotazioni

Per il procedimento d'annotazione si sono presi in considerazione i video riguardanti gli highlights della partita.
Ogni video è, all'interno del database, un evento e possiede il proprio identificativo. Tutte le tuple relative ai dati posizionali dei giocatori fanno riferimento ad un particolare evento della partita ed hanno quindi, tra gli attributi, l'identificativo di quello a cui si riferiscono.
Si sono osservati tutti i video fotogramma per fotogramma andando ad etichettare ognuno di esso come:

- *passaggio*, se in quel fotogramma il giocatore in possesso della palla, la lascia per iniziare un passaggio (figura 4.6 (a);
- *ricezione*, se in quel fotogramma il giocatore a cui è stato indirizzato il passaggio riceve effettivamente la palla (4.6 (b));
- *altro*, per tutti i fotogrammi che non sono né passaggio né ricezione.

(a) esempio di fotogramma etichettato come passaggio

(b) esempio di fotogramma etichettato come ricezione

Figura 4.6: Esempio di annotazione di un passaggio.

Si è creata quindi una nuova tabella dove ogni tupla è formata da un contatore numerico che equivale al numero del fotogramma della partita e dall'etichetta di passaggio, ricezione o altro (figura 4.7).

Figura 4.7: Esempio di una tupla dopo la fase di annotazione

4.4 Costruzione del data set

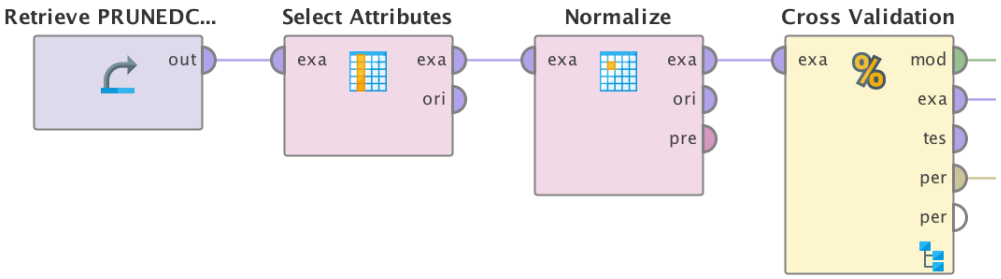
La costruzione del data set completo è stata fatta su Visual Studio Code utilizzando il linguaggio di programmazione c#.
Alla suddetta tupla sono state aggiunte tutte le feature precedentemente elencate, che vanno quindi a completare le informazioni usate successivamente per l'apprendimento da parte dell'algoritmo di machine learning utilizzato.
Il popolamento del data set è avvenuto in due differenti fasi. Nella prima sono state calcolate le feature per ogni fotogramma, quindi per ogni istante t della partita, e si è creato un data set contenente le tuple di ogni frame di tutti gli eventi che si è potuto analizzare (Figura 4.8). AttraversoAttraverso un programma creato sempre su Viual Studio Code si è proceduto, in una seconda fase, al pruning randomico. Per ovviare infatti ai problemi di overfitting e sbilanciamento per quanto riguarda le etichette, data l'enorme differenza tra numero di fotogramma etichettati come passaggio e ricezione, rispetto a quelli etichettati come altro, si è usato un tool appositamente creato per pulizia dei dati.
Dopo le due fasi, il data set che è stato poi utilizzato su RapidMiner risulta di 171 tuple ugualmente ripartite in tuple etichettate come passaggio, ricezione o altro.

Figura 4.8: Esempio di una tupla dopo la fase di annotazione

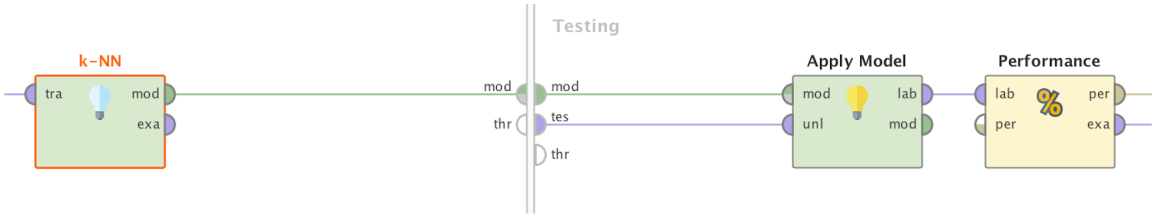
4.5 Rilevazione degli eventi

Per le funzionalità di machine learning è stato utilizzato il programma RapidMiner. L'algoritmo utilizzato in questo lavoro è il KNN in quanto è quello con cui si sono ottenuti i risultati migliori. Il valore di K è stato impostato ad 1.
Per ovviare alla limitata quantità di dati disponibili, è stato usato il cross validation con un numero di partizioni uguale a 20 ed un campionamento lineare per la costruzione delle partizioni.
In figura 4.9 è possibile vedere come è stato configurato RapidMiner ed i nodi utilizzati in questa fase.

Il nodo di selezione degli attributi è stato utilizzato poiché sono state fatti diversi tentativi con diverse feature per valutare le diverse prestazioni dell'algoritmo e l'influenza di quelle ogni volta utilizzate. Anche per quanto riguarda la normalizzazione è stata fatta una selezione delle feature in quanto non tutte, dopo questa operazione, contribuivano positivamente all'apprendimento dell'algoritmo.
Nel capitolo successivo verranno analizzate le diverse prove effettuate con le diverse feature e normalizzazion, e discussi i risultati ottenuti.



(a)



(b)

Figura 4.9: Nodi utilizzati per la selezione, normalizzazione e cross validation (a). Nodi utilizzati per l'esecuzione dell'algoritmo e la rilevazione delle sue prestazioni (b).

Capitolo 5

Valutazione

5.1 Analisi dei risultati

In questo capitolo si sono valutati i risultati ottenuti dall'applicazione dell'algoritmo di machine learning considerato, variando di volta in volta le feature utilizzate al fine di individuare quelle in grado di offrire le prestazioni migliori.
La misura delle prestazioni è stata effettuata tenendo in considerazione le metriche di seguito elencate:

- Accuracy;
- Precision;
- Recall;
- F-measure;

Per ogni prova effettuata sono elencate le feature utilizzate e quelle normalizzate, la tabella rappresentante la matrice di confusione e le suddette misure. Le sezioni che segueno aggiungono, di volta in volta, una o più feature o variano la normalizzazione di alcune delle già presenti. Si sono prese in considerazione, per la prima analisi, le stesse feature presenti in [7] e ne sono state aggiunte progressivamente delle altre.

5.1.1 Utilizzo delle feature 2D

Nel primo tentativo effettuato si sono utilizzate solamente le feature 2D che sono state adoperate nel lavoro adoperato in [7].
Esse sono elencate di seguito:

- Acc_{2D};
- $a_{max_{2D}}$;
- $a_{min_{2D}}$;
- $ctr_{orizzontale}$;
- cd_{2d};
- $Dist_{2D}toBasketTarget$

- Vel_{2D};
- p_x;
- p_y.

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	23	18	20	0,38	0,40	0,39	33,68
PASSAGGIO predetto	16	17	20	0,32	0,30	0,31	
RICEZIONE predetto	18	22	17	0,30	0,30	0,30	

Tabella 5.1: Risultati del KNN usando le feature 2D e senza alcuna normalizzazione

Come si evince dalla tabella 5.1, senza la normalizzazione, i risultati risultano essere piuttosto scadenti, con un'accuratezza del 33.68%.

5.1.2 Utilizzo delle feature 2D con normalizzazione

Si è provveduto successivamente alla normalizzazione di alcune delle feature precedentemente utilizzate. Negli algoritmi basati sulla misura della distanza come il KNN, normalizzare permette di scalare le features in modo da evitare che alcune di esse, con una scala maggiore rispetto ad altre, vengono favorire nel calcolo della distanza. Le feature normalizzate sono qui sotto elencate:

- Acc_{2D};
- $a_{max_{2D}}$;
- $a_{min_{2D}}$;
- $ctr_{orizzontale}$;
- $Dist_{2D}toBasketTarget$
- Vel_{2D};

Si può notare, in tabell 5.2, come l'esecuzione della normalizzazfione influisca positivamente sulle prestazione del KNN portando l'accuratezza al 68,4%.

5.1.3 Utilizzo della distanza dal giocatore più vicino

Partendo da questo risultato si è aggiunta e normalizzata, assieme a quelle sopra elencate, la feature relativa alla distanza dal giocatore più vicino alla palla, DistfPC. Si è percepito in questo caso un aumento dell'accuratezza di 1.1 punti percentuali, mentre i valori di Precision, Recall e F-Measure sono rimasti pressoché invariati

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	54	2	3	0,92	0,95	0,93	68,40
PASSAGGIO predetto	0	32	22	0,59	0,56	0,57	
RICEZIONE predetto	3	23	32	0,55	0,56	0,55	

Tabella 5.2: Risultati del KNN usando le feature 2D ed effettuando la normalizzazione

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	54	2	3	0,92	0,95	0,93	69,51
PASSAGGIO predetto	0	32	22	0,59	0,56	0,57	
RICEZIONE predetto	3	23	32	0,55	0,56	0,55	

Tabella 5.3: Risultati del KNN aggiungendo la distanza dal giocatore più vicino, DistfPC.

5.1.4 Utilizzo della varianza del cambio di direzione 2D

Si è guadagnato circa un altro punto percentuale in accuratezza aggiungendo la varianza del cambio di direzione 2D nel secondo precedente e la varianza del cambio di direzione 2D in quello successivo (tabella 5.4):

- $var - cd_{2d} - after$;
- $var - cd_{2d} - Before$.

Oltre all'Accuracy, anche Precision e Recall hanno evidenziato un leggero miglioramento. In questo caso, non è stata applicata una normalizzazione alla nuova feature inserita.

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	2	3	0,92	0,98	0,95	70,62
PASSAGGIO predetto	0	33	21	0,61	0,58	0,59	
RICEZIONE predetto	1	22	33	0,59	0,58	0,58	

Tabella 5.4: Risultati del KNN aggiungendo la varianza del cambio di direzione 2D nel secondo precedente e quella nel secondo successivo.

5.1.5 Utilizzo della varianza della distanza dal giocatore più vicino

Le varianze della distanza dal giocatore più vicino un secondo prima e un secondo dopo portano invece l'accuratezza dal 70.62% al 73.19% (tabella 5.5).

- $var - DistfPC - After$;
- $var - DistfPC - Before$;

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	2	4	0,90	0,98	0,94	73,19
PASSAGGIO predetto	0	33	16	0,67	0,58	0,62	
RICEZIONE predetto	1	22	37	0,62	0,65	0,63	

Tabella 5.5: Risultati del KNN aggiungendo la varianza del cambio di direzione 2D nel secondo precedente e quella nel secondo successivo.

5.1.6 Utilizzo della squadra del giocatore più vicino

Aggiungendo la feature relativa alla squadra a cui appartiene il giocatore più vicino, TeamPlayerCloser, l'accuratezza acquista ancora mezzo punto percentuale. Anche in questo caso, sulla feature aggiunta non si è effettuata alcuna normalizzazione (tabella 5.6).

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	2	3	0,92	0,98	0,95	73,68
PASSAGGIO predetto	0	36	19	0,65	0,63	0,64	
RICEZIONE predetto	1	19	35	0,64	0,61	0,62	

Tabella 5.6: Risultati del KNN aggiungendo la squadra del giocatore più vicino alla palla.

5.1.7 Utilizzo di Z

La possibilità di utilizzare la terza dimensione e il suo successivo utilizzo tout court ha evidenziato come l'inserimento di questa feature non porti alcuni risultato positivo, ma faccia, invece, abbassare le prestazioni dell'algoritmo riducendo l'accuratezza dal 73,68% al 71,32%.

Come mostrato in tabella 5.8, essa va ad influenzare maggiormente la misura di Recall relativa al passaggio, che passa dallo 0,63 al 0,56.

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	3	3	0,90	0,98	0,94	71,32
PASSAGGIO predetto	0	32	19	0,63	0,56	0,59	
RICEZIONE predetto	1	22	35	0,60	0,61	0,60	

Tabella 5.7: Risultati del KNN aggiungendo solo la componente Z.

5.1.8 Utilizzo di Velocità e Accelerazione verticale

Risultano utili invece le misure derivate dalla terza dimensione e calcolate scindendo questa dalle altre due. In particolare, la velocità e l'accelerazione verticale della palla migliorano l'accuratezza del 3% rispetto a quella ottenuta con la sola componente verticale z (tabella X) .

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	2	2	0,93	0,98	0,95	74,31
PASSAGGIO predetto	0	35	18	0,66	0,61	0,63	
RICEZIONE predetto	1	20	37	0,64	0,65	0,64	

Tabella 5.8: Risultati del KNN aggiungendo la velocità e l'accelerazione verticale.

5.1.9 Sostituzione del 3D al 2D

Si intravede ancora un piccolo miglioramento di 0.6 punti percentuale, andando a sostituire tutte le feature 2D con le corrispettive feature 3D (tabella 5.9). In questo modo quindi, velocità, accelerazione, picchi di accelerazione e cambio di direzione vengono calcolate tenendo in considerazione tutte le tre dimensioni della posizione nello spazio della palla.
Le feature quindi selezionate sono:

- Acc_{3D};
- $a_{max_{3d}}$;
- min_{3d};
- $ctr_{orizzontale}$;

- cd_{3d};
- $Dt_{3D}toBasketTarget$;
- $DistfPC$;
- $TeamPC$;
- Vel_{3D};
- A_y;
- V_y;
- p_x;
- p_y;
- $avg - cd_{2d} - after$;
- $avg - cd_{2d} - Before$;
- $avg - DistfPC - After$;
- $avg - DistfPC - Before$;

di cui, normalizzate:

- Acc_{3D};
- $a_{max_{3d}}$;
- min_{3d};
- $ctr_{orizzontale}$;
- $Dt_{3D}toBasketTarget$;
- $DistfPC$;
- Vel_{3D};
- A_y;
- V_y.

5.1.10 Utilizzo delle medie

La costruzione di alcune feature in grado di considerare una finestra temporale offre la possibilità di sfruttare alcune informazioni che gli algoritmi di machine learning non riescono a rilevare autonomamente. La media temporale, in particolare, porta dei vantaggi notevoli in quanto va ad accentuare le differenze di un evento rispetto ad un altro nel corso del tempo. Ad eccetto della varianza relativa al cambio di direzione 2D e quella relativa alla distanza dal giocatore più vicino, le altre, invece, non hanno apportato alcun miglioramento al KNN. Aggiungendo alcune di esse, si è portata l'accuratezza al massimo trovato in questo lavoro, con una percentuale del 76.67% (tabella 5.10) Le medie che sono state aggiunte sono:

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	56	2	2	0,93	0,98	0,95	74,93
PASSAGGIO predetto	0	35	18	0,66	0,61	0,63	
RICEZIONE predetto	1	20	37	0,64	0,65	0,64	

Tabella 5.9: Risultati del KNN dopo la sostituzione delle feature 2D con quelle 3D e alla loro normalizzazione.

	ALTRO	PASSAGGIO	RICEZIONE	Precision	Recall	F-measure	Accuracy %
ALTRO predetto	57	2	2	0,93	1	0,96	76,67
PASSAGGIO predetto	0	37	17	0,69	0,65	0,67	
RICEZIONE predetto	0	18	38	0,68	0,67	0,67	

Tabella 5.10: Risultati del KNN ottenuti aggiungendo le medie temporali di cambio di direzione 2D, distanza dal giocatore più vicino e posizione verticale della palla.

5.2 Generazione dell'animazione in un'applicazione di Realtà Virtuale

In questa sezione si spiega in che modo si possono sfruttare (e sono stati in effetti sfruttati) i risultati ottenuti grazie al machine learning, applicando le informazioni di classificazione alla generazione di animazioni di eventi di tipo passaggio in un'applicazione di Realtà Virtuale. Lo sviluppo di questa applicazione ha occupato la prima parte di questo e di altri due lavori di tesi collegati realizzati da Giuseppe Ministeri e Marco Musto. Essa consiste nel ricreare una partita di basket in cui sono state acquisiti i dati di tracking dei giocatori e della palla e permettere di rivederla e riviverla attraverso l'utilizzo di un visore di Realtà Virtuale. L'applicazione fa parte di un progetto più ampio pensato per supportare gli atleti e l'allenatore nello svolgimento degli allenamenti. Nella realizzazione dell'applicazione si è utilizzato il game engine Unity3D.
L'identificativo utilizzato per la costruzione del data set di training, ovvero il numero del fotogramma, è stato utilizzato per la creazione dell'entità base utilizzata per la ricostruzione della partita. Un fotogramma su Unity3D diventa quindi un oggetto (Frame) contenente un identificativo incrementale che indica il tempo in millisecondi dall'inizio della partita, una lista con le informazioni di posizione di tutti i giocatori, la posizione della palla, una lista con le feature calcolate precedentemente e la previsione ottenuta in output da RapidMiner (Figura 5.1).

Figura 5.1: Oggetto Frame di Unity3D contenente tutte le informazioni per la creazione di un istante di gioco.

L'insieme di tutti i Frame formano la Timeline. Essa rappresenta l'intera partita di basket all'interno dell'applicazione ed è possibile "muoversi" avanti e indietro per spostarsi temporalmente all'interno della partita (Figura X) .

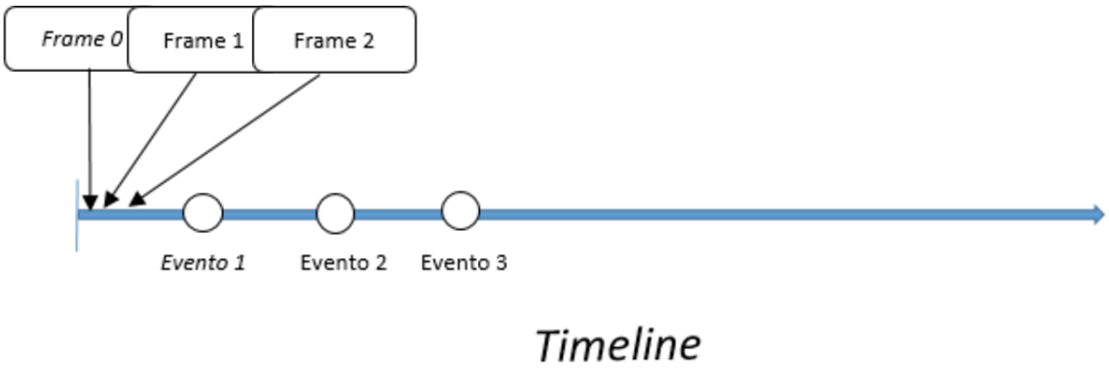


Figura 5.2: Composizione della Timeline dell'applicazione. I Frame forniscono le informazioni sulle posizione che assumono i giocatori e la palla ad ogni passo temporale. I cerchi indicano invece l'azione in cui ci si trova. Ogni azione è un evento con un proprio identificativo univoco.

L'informazione di previsione all'interno di ogni frame permette di attivare la relativa animazione quando ci si trova in quel particolare momento. La figura X mostra le differenze nella visualizzazione dell'applicazione senza e con l'utilizzo del machine learning nell'evento del passaggio. Nel caso (a), non essendoci nessuna informazione oltre a quella di posizione, non è possibile ricreare l'animazione adatta a simulare l'azione fisica del giocatore. Il risultato è che la palla si muove da un giocatore all'altro (dato che sono presenti i dati spaziali della palla), senza però alcuna interazione col giocatore (che nel frattempo esegue sempre l'animazione della corsa).

Nel secondo caso, grazie alle previsioni effettuate dall'algoritmo di machine learning, è possibile invece identificare il momento esatto in cui bisogna attivare l'animazione relativa all'evento predetto, in questo caso, il passaggio. Essa comporta, per l'evento in esame, il movimento degli arti superiori da parte del giocatore in possesso della palla nel momento in cui questo prova a passarla ad un compagno.

(a)

(b)

Figura 5.3: Esempio di sequenza di un evento senza l'uso del machine learning (a) e la stessa sequenza dello stesso evento con l'animazione del passaggio predetta dall'algoritmo di machine learning (b).

Capitolo 6

Conclusioni

In questo capitolo vengono presentate le conclusioni e alcune considerazioni sul lavoro svolto. Si accennano anche alcune idee per spunti futuri e migliorie attuabili su quanto prodotto nel presente lavoro di tesi.

6.1 Considerazioni finali

Questo lavoro di tesi è stato sviluppato per la creazione di un sistema in grado di generare automaticamente contenuti 3D mediante tecniche di machine learning. Esso permette, in contesti dove la generazione dei contenuti 3D si serve delle annotazioni,come in [5], di evitare il lavoro manuale che risulta essere dispendioso in termini di tempo.
Oggi, lo sport fa un uso sempre maggiore della tecnologia come strumento di supporto alle prestazioni degli atleti. Sia a livello agonistiche che non, è possibile ormai trovare delle soluzioni per l'acquisizione di alcune informazioni utili per l'analisi delle partite giocate e della condizione fisica degli atleti.
Nella realizzazione del lavoro sono stati utilizzati i dati di tracking delle partite NBA in [6]. In particolare, i dati impiegati descrivono i movimenti dei giocatori e della palla in una partita tra i San Antonio Spurs e Minnesota. Rispetto al lavoro presentato in [7], le informazioni utilizzate nel caso in esame hanno permesso l'utilizzo della componente di altezza nei dati posizionali della palla. Si è provato, quindi, ad analizzare quanto questa componente potesse influire nel riconoscimento degli eventi in una partita di basket.
La fase di acquisizione e trasformazione dei dati è stata eseguita utilizzando il linguaggio di programmazione c# all'interno dell'ambiente di sviluppo Visual Studio Code.
La fase di apprendimento e di classificazione dei fatti ha coinvolto tecniche di machine learning (in particolare, l'algoritmo KNN) attraverso il programma RapidMiner e la sua interfaccia a nodi, mentre per la creazione effettiva delle animazioni all'interno di un'applicazione di Realtà Virtuale si è usato il game engine Unity3D.
I risultati ottenuti indicano che la possibilità di utilizzare le informazioni sulla coordinata verticale può portare degli effettivi vantaggi, migliorando l'accuratezza della classificazione.
Si è in particolare mostrato come l'utilizzo tout court di questa componente non offra un incremento sensibile dei risultati. Essa è tuttavia essenziale, poiché permette la creazione di diverse feature dalle quali è possibile l'estrazione di informazioni utili

per l'algoritmo in fase di apprendimento.
Lo studio, evidenzia come anche la creazione di feature che sfruttano una finestra temporale possa aiutare l'algoritmo di machine learning. Esse permettono di racchiudere un evento che si consuma nel tempo all'interno di una sola informazione.
La scelta di utilizzare l'algoritmo KNN è dovuta alla sua semplicità e alla sua velocità nella fase di apprendimento. La figura 6.1 riassume il miglioramento apportato all'accuratezza dalla terza dimensione e dallo sviluppo delle nuove feature, rispetto a quello dovuto all'impiego delle sole feature 2D.

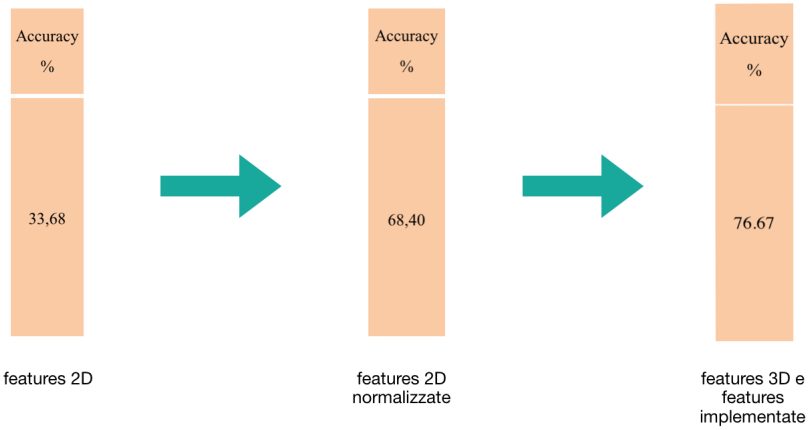


Figura 6.1: Accuratezza di partenza del KNN con le feature 2D non normalizzate a sinistra. Accuratezza dell'algoritmo normalizzando le feature 2D al centro. Accuratezza finale con le nuove feature implementate a destra.

La possibilità di generare contenuti 3D sfruttando il machine learning porta a pensare ad un futuro in cui sarà più facile, grazie appunto all'intelligenza artificiale, estendere il mondo reale all'interno di quello virtuale. Questo potrà comportare una immedesimazione maggiore da parte dell'utente e, di conseguenza, un incremento della presenza all'interno dell'ambiente virtuale. Una verosimiglianza sempre più raffinata aiuterà a nascondere la tecnologia utilizzata durante la fase di fruizione e ad ingannare il giudizio degli utenti su ciò che è reale e su quello che non lo è, obiettivo primario della Realtà Virtuale.

6.2 Sviluppi futuri

In questa sezione vengono forniti alcuni spunti per un eventuale sviluppo futuro di questo lavoro di tesi. L'analisi ulteriore dei dati in possesso potrebbe portare all'estrazione di altre informazioni nel riconoscimento del passaggio. Si potrebbe pensare, ad esempio, alle feature come segnali da mappare in un altro dominio, come quello di frequenza, per cercare di ricavare altre tipi di correlazioni e migliorare il suo riconoscimento.
Il riconoscimento stesso potrebbe essere esteso alla rilevazione di altri tipi di eventi oltre a quello del passaggio. Questo comporterebbe uno studio ulteriore delle feature e un'analisi su come quelle pensate per un determinato evento potrebbero influire su quelle pensate per un evento diverso.

Si potrebbero utilizzare le informazioni di posizione dei giocatori per la creazione di alcune feature ad hoc con lo scopo di integrarle con quelle relative alla palla. Questo potrebbe comportare la necessità di sfruttare i dati di altre partite di basket per cercare di generalizzare l'apprendimento e non adattarlo eccessivamente alle caratteristiche di pochi giocatori. Infatti, integrare diverse partite eviterebbe al sistema di machine learning di farsi condizionare eccessivamente dalla soggettività dei movimenti, caratterizzando ogni evento in base a delle caratteristiche peculiari dell'evento stesso.
La tecnologia odierna permette l'acquisizione dei dati di tracking quasi in tempo reale. La possibilità di estendere l'applicazione per consentire una fruizione contemporanea alla partita reale potrebbe portare all'implementazione a run-time dell'algoritmo di machine learning. Per far ciò, sarebbe necessario abbandonare RapidMiner come tool per il machine learning e implementare un algoritmo di classificazione capace di propagare gli errori all'indietro per auto migliorarsi. A questo scopo, si potrebbe utilizzare python come linguaggio di programmazione attraverso l'utilizzo del framework di Google TensorFlow.
Utilizzare i dati di tracking per riconoscere la modalità di esecuzione di un evento potrebbe essere un ulteriore passo avanti. La possibilità di ottenere le informazioni su quale evento riprodurre accompagnate da come effettivamente tale evento debba essere ricreato virtualmente potrebbero un maggiore grado di accuratezza nell'esecuzione della relativa animazione, contribuendo quindi ad aumentare ulteriormente la verosimiglianza dell'applicazione.

Bibliografia

[1] *Sito di STATS*. URL: https://www.stats.com/.

[2] Suat Gedikli et al. *An Adaptive Vision System for Tracking Soccer Players from Variable Camera Settings*.

[3] Metulini R., Manisera M. e Zuccolotto P. *Space-Time Analysis of Movements in Basketball using Sensor Data, "Statistics and Data Science: new challenges, new generations"*. 2017.

[4] *Sito di Beyond Sports*. URL: https://www.beyondsports.nl/.

[5] Daniel Faustino. *Beyond Mocap: Animating Soccer Players Based on Positional Tracking Data (Master's thesis)*. 2016. URL: https://dspace.library.uu.nl/handle/1874/338670.

[6] *Database dei dati di tracking della partita di NBA*. URL: https://github.com/sealneaward/nba-movement-data/blob/master/data/12.23.2015.SAS.at.MIN.7z.

[7] Keven Richly et al. «Recognizing Compound Events in Spatio-Temporal Football Data». In: (apr. 2016).

[8] David C Gross et al. «Report from the fidelity implementation study group». In: *Fall Simulation Interoperability Workshop Papers*. 1999.

[9] *Merriam-Webster.com*. URL: https://www.merriam-webster.com/dictionary/virtual%20reality.

[10] *Sito di Morton Heilig*. URL: http://www.mortonheilig.com/InventorVR.html.

[11] *Brevetto del Sensorama*. URL: http://www.mortonheilig.com/SensoramaPatent.pdf.

[12] Ivan E. Sutherland. «A Head-mounted Three Dimensional Display». In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686. URL: http://doi.acm.org/10.1145/1476589.1476686.

[13] Michael Naimark. «Aspen the verb: Musings on heritage and virtuality». In: *Presence: Teleoperators and Virtual Environments* 15.3 (2006), pp. 330–335.

[14] *Sito dell'università del Quèbec*. URL: http://w3.uqo.ca/cyberpsy/en/pres_en.htm.

[15] Arthur L Samuel. «Some studies in machine learning using the game of checkers». In: *IBM Journal of research and development* 44.1.2 (2000), pp. 206–226.

[16] *One-vs-All Multiclass, sito web Microsoft Azure*. URL: https://msdn.microsoft.com/en-us/library/azure/dn905887.aspx.

[17] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth et al. «Knowledge Discovery and Data Mining: Towards a Unifying Framework.» In: *KDD*. Vol. 96. 1996, pp. 82–88.

[18] Pang-Ning Tan et al. *Introduction to data mining*. Pearson Education India, 2006.

[19] Marina Sokolova, Nathalie Japkowicz e Stan Szpakowicz. «Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation». In: *Australian conference on artificial intelligence*. Vol. 4304. 2006, pp. 1015–1021.

[20] Kelly H Zou, A James O'Malley e Laura Mauri. «Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models». In: *Circulation* 115.5 (2007), pp. 654–657.

[21] Martin Erwig et al. «Spatio-temporal data types: An approach to modeling and querying moving objects in databases». In: *GeoInformatica* 3.3 (1999), pp. 269–296.

[22] *Rilascio di TensorFlow Lite*. URL: https://venturebeat.com/2017/05/17/android-launches-tensorflow-lite-for-mobile-machine-learning/.

[23] Andre Esteva et al. «Dermatologist-level classification of skin cancer with deep neural networks». In: *Nature* 542.7639 (2017), pp. 115–118.

[24] Georgios Palioura, Vangelis Karkaletsis e Constantine D. Spyropoulos. «Machine Learning and Its Applications, Springer-Verlag Berlin Heidelberg». In: *Medicine* ().

[25] I. Bratko, I. Mozetič e N. Lavrač. *KARDIO: a study in deep and qualitative knowledge for expert systems*. MIT Press, 1989. ISBN: 9780262022736. URL: https://books.google.it/books?id=uU9RAAAAMAAJ.

[26] Niyati Aggrawal et al. «Brand analysis framework for online marketing: ranking web pages and analyzing popularity of brands on social media». In: *Social Network Analysis and Mining* 7.1 (mag. 2017), p. 21. URL: https://doi.org/10.1007/s13278-017-0442-5.

[27] Yonghui Wu et al. «Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation». In: *CoRR* abs/1609.08144 (2016). URL: http://arxiv.org/abs/1609.08144.

[28] Mariusz Bojarski et al. «End to end learning for self-driving cars». In: *arXiv preprint arXiv:1604.07316* (2016).

[29] *Sito Internet di Magenta*. URL: https://magenta.tensorflow.org/welcome-to-magenta.

[30] Kelvin Xu et al. «Show, attend and tell: Neural image caption generation with visual attention». In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[31] *Sito internet di DeepDream*. URL: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

[32] Simon Tong e Daphne Koller. «Support vector machine active learning with applications to text classification». In: *Journal of machine learning research* 2.Nov (2001), pp. 45–66.

[33] Thomas Cover e Peter Hart. «Nearest neighbor pattern classification». In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[34] *Random Forest, università di Berkeley, California*. URL: `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`.

[35] *Sito di Perception Neuron*. URL: `https://neuronmocap.com/`.

[36] *Sito di Elcetron*. URL: `https://electron.atom.io/`.

[37] *Sito di Visual Studio Code*. URL: `https://code.visualstudio.com/docs`.