SAPIENZA
UNIVERSITÀ DI ROMA

Analisi e compressione di matrici di covarianza per la proiezione di ostacoli su un piano bidimensionale

Ingegneria dell'informazione, informatica e statistica

Corso di Laurea in Ingegneria Informatica e automatica

Candidato

Francesco Defulgentiis
Matricola 1950067

Relatore

Prof. Thomas Alessandro Ciarfuglia

Anno Accademico 2023/2024

Analisi e compressione di matrici di covarianza per la proiezione di ostacoli su un piano bidimensionale
Tesi di Laurea. Sapienza – Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: defulgentiis.1950067@studenti.uniroma1.it

Sommario

La RoboCup è un'organizzazione dedicata alla promozione della ricerca robotica in diversi settori, tra cui il calcio, il soccorso e l'automazione domestica. Il suo obiettivo principale è quello di definire regolamenti che stimolino lo sviluppo e il progresso delle ricerche nel campo della robotica. Il team SPQR dell'Università La Sapienza partecipa alla Open Soccer League, la quale presenta una caratteristica unica: l'utilizzo di una piattaforma universale condivisa da tutte le squadre, il NAO. Questo robot è dotato di capacità motorie eccezionali e di un sofisticato sistema sensoriale, che i team devono utilizzare per consentire al robot di giocare autonomamente.

Tuttavia, questa lega impone anche diverse limitazioni, tra cui una quantità limitata di memoria interna e processori embedded con prestazioni limitate. Queste limitazioni sollevano problematiche che raramente sarebbero considerate in un ambiente moderno caratterizzato da una potenza computazionale in costante crescita. È per affrontare questa sfida che questa tesi si propone di ottimizzare il software di identificazione degli errori, riducendo la quantità di dati memorizzati e trasmessi senza compromettere le prestazioni computazionali del robot.

La risoluzione di questa complessa problematica sarà affrontata attraverso diversi approcci risolutivi.

Nel Capitolo 1 verranno definiti gli strumenti che caratterizzano le circostanze nella quale verrà svolta la ricerca.

Nel Capitolo 2 si discuterà la metodologia con la quale è stata svolta l'analisi assieme ad alcune osservazioni sui dati.

Nel Capitolo 3 verranno presentate diverse soluzione al problema, con i relativi funzionamenti e pseudocodici.

Nel Capitolo 4 saranno mostrati estensivamente i metodi di analisi, fornendo gli strumenti adatti per le riflessioni successive.

Nel Capitolo 5 saranno mostrati attraverso tabelle e grafici i risultati ottenuti dalla ricerca, basandosi su metriche di paragone univoche.

Nel Capitolo 6 sarà effettuata un analisi soggettiva basata sui risultati ottenuti, discutendo i pro ed i contro delle diverse soluzioni proposte.

Nel Capitolo 7 verranno discussi diversi campi applicativi nei quali il prodotto di questo lavoro potrebbe essere utile.

Indice

1 Background **1**
1.1 Analisi strutturale delle matrici di covarianza 1
1.2 Proprietà della matrice di covarianza 2
1.3 proprietà dell'hardware e del software di interesse 3

2 Metodologia **4**
2.1 Raccolta ed analisi dei dati 4
2.2 Osservazioni conclusive 4

3 Diverse soluzioni **6**
3.1 Mappatura delle matrici attraverso una conversione di base 6
3.2 Mappatura mediante un dizionario ordinato 9
3.3 Uso di una rete Neurale 11

4 Valutazione **12**
4.1 valutazione conversione di base 12
4.2 valutazione del dizionario ordinato 13
4.3 valutazione della rete neurale 13

5 Risultati **15**
5.1 Tabelle 15
5.2 Grafici a barre 16

6 Conclusioni **17**

7 Altri applicativi **18**

Bibliografia **19**

Ringraziamenti **20**

Capitolo 1

Background

1.1 Analisi strutturale delle matrici di covarianza

Nell'ambito dell'analisi statistica, la varianza emerge come una misura cruciale della dispersione dei valori di una singola variabile attorno alla sua media. Formalmente, per una variabile aleatoria X con valori osservati $x_1, x_2, \ldots, x_n$ e media $\bar{x}$, la varianza $\mathrm{Var}(X)$ è calcolata come:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Parallelamente, la covarianza si concentra sulla relazione congiunta tra due variabili. Considerando due variabili aleatorie X e Y con valori osservati $x_1, x_2, \ldots, x_n$ e $y_1, y_2, \ldots, y_n$ rispettivamente, la covarianza $\mathrm{Cov}(X, Y)$ è definita come:

$$\mathrm{Cov}(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

La matrice di covarianza è invece un'entità matematica che rappresenta le relazioni di varianza e covarianza tra le diverse variabili presenti in un insieme di dati multivariati. Formalmente, considerando un insieme di n variabili aleatorie $X_1, X_2, \ldots, X_n$ con valori osservati $x_1, x_2, \ldots, x_n$, la matrice di covarianza C è una matrice simmetrica $n \times n$ il cui elemento nella riga i e nella colonna j è definito come:

$$C_{ij} = \mathrm{cov}(X_i, X_j) = \frac{1}{N}\sum_{k=1}^{N}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)$$

Dove $\mathrm{cov}(X_i, X_j)$ rappresenta la covarianza tra le variabili X_i e X_j, N è il numero di osservazioni nel dataset e $\bar{x}_i$ e $\bar{x}_j$ sono le medie campionarie delle variabili X_i e X_j rispettivamente.

La diagonale principale della matrice di covarianza contiene le varianze delle singole variabili, mentre gli elementi al di fuori della diagonale forniscono le covarianze tra le coppie di variabili.

Nell'ambito di questa tesi, varianza e covarianza sono usate per dare una misura di incertezza su posizione ed orientamento di un ostacolo in un piano bidimensionale; le variabili del dataset saranno dunque:

- X è la posizone registrata sull'asse orizzontale (l'asse immaginario che connette il centro delle due porte).
- Y è la posizone registrata sull'asse verticale (l'asse di 'centrocampo', il qualche divide il campo in due).
- θ rappresenta l'orientamento dell'oggetto ovvero la sua rotazione.

Nella figura sotto riportata, viene definito il sistema di coordinate utilizzato, il quale pone l'origine nel centro esatto del campo, suddividendolo in 4 settori.

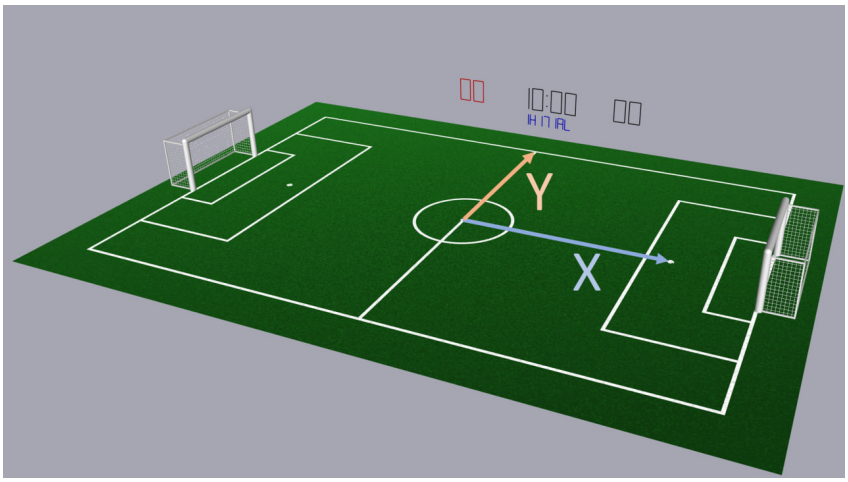

Figura 1.1. Rappresentazione digitale del campo di gioco

1.2 Proprietà della matrice di covarianza

Le matrici di covarianza presentano diverse proprietà strutturali rilevanti. Alcune delle principali sono:

- Simmetria: la simmetria della matrice di covarianza deriva direttamente dalla definizione della covarianza stessa. La covarianza tra due variabili X e Y è calcolata come:

$$\mathrm{Cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

  Notiamo che la covarianza tra X e Y è definita simmetricamente: $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$. Questo è dovuto al fatto che il calcolo della covarianza non dipende dall'ordine delle variabili ma solo dalla loro relazione con le rispettive medie.

- Diagonale principale non negativa: La diagonale della matrice di covarianza è non negativa perché contiene le varianze delle singole variabili. La varianza di una variabile aleatoria è sempre non negativa, poichè ogni termine nella sommatoria $(x_i - \bar{x})^2$ è non negativo in quanto è il quadrato di un numero reale.

1.3 proprietà dell'hardware e del software di interesse

Al centro della competizione RoboCup si trova NAO, un robot umanoide sviluppato da SoftBank Robotics. NAO rappresenta una sintesi di tecnologie all'avanguardia e design ergonomic. Equipaggiato con una serie di sensori e capacità di elaborazione avanzate, NAO è in grado di percepire il suo ambiente circostante e collaborare con altri robot per raggiungere obiettivi comuni

Per comprendere appieno il robot NAO e il suo impatto nella competizione RoboCup, è essenziale esplorare il suo complesso hardware. Una delle caratteristiche distintive di NAO è la sua testa, che ospita una serie di sensori e dispositivi essenziali per l'interazione e la percezione. Questi includono telecamere ad alta risoluzione, microfoni per il riconoscimento vocale, altoparlanti per la riproduzione del suono e sensori di profondità per la visione tridimensionale. Altri componenti hardware di NAO includono una serie di sensori di pressione distribuiti sul suo corpo, che gli consentono di rilevare il contatto con il terreno e adattare i suoi movimenti di conseguenza. Questi sensori sono particolarmente utili durante la locomozione, consentendo a NAO di mantenere l'equilibrio e evitare ostacoli mentre si muove.

La fase di sviluppo di questi robot richiede centinaia di ore di test sul campo, dove il team di ricercatori ha l'opportunità di implementare nuove strategie e migliorare quelle precedenti. Questo sarebbe un processo estremamente impegnativo, e per questo motivo è stato sviluppato un simulatore, il quale realizza un campo digitale dove è possibile posizionare robot e ostacoli per testare il comportamento dei robot, è grazie a questo strumento che successivamente in questo testo sarà possibile ottenere dati a sufficenza sulle matrici generate in una partita reale.

Capitolo 2

Metodologia

2.1 Raccolta ed analisi dei dati

Prima di sviluppare uno specifico approccio risolutivo, è sicuramente necesario svolgere un'analisi approfondita dei dati che verranno successivamente manipolati. A questo scopo è stato impiegato un software di simulazione chiamato SimRobot, fornito dal team SPQR della Sapienza. Questo strumento è capace di simulare la percezione e le interazioni dei robot in un campo da calcio, ponendoli nelle condizioni più reali possibili. Questo strumento risulta fondamentale per l'ottenimento di dati reali.

Per ricavare le matrici generate durante la simulazione, è importante innanzitutto creare le condizioni adatte; dunque un programma che descriva al simulatore quali e quanti robot sono presenti sul campo, le capacità percettive di cui essi dispongono e molto altro. Successivamente, è stato sviluppato un modulo per raccogliere le matrici generate dal robot per registrarle in un file di testo. Quest'ultimo viene poi letto da uno script Python, il quale si occupa di generare il dataframe che sarà oggetto di analisi nel capitolo successivo.

Grazie all'uso della libreria python "pandas" la quale semplifica notevolmente l'analisi di dati, è possibile ottenere le seguenti statistiche riguardanti il dataframe generato

	varX	covXY	covXTh	covYX	varY	covYTh	covThX	covThY	varTh
count	37517.000000	37517.000000	37517.000000	37517.000000	37517.000000	37517.000000	37517.000000	37517.000000	37517.0
mean	36322.371272	11.824320	-0.012528	11.824320	5275.922755	-0.026894	-0.012528	-0.026894	0.0
std	125352.214154	39.984769	0.222190	39.984769	18087.266887	0.351980	0.222190	0.351980	0.0
min	7.000000	-208.000000	-5.000000	-208.000000	24.000000	-5.000000	-5.000000	-5.000000	0.0
max	495548.000000	441.000000	4.000000	441.000000	250520.000000	3.000000	4.000000	3.000000	0.0

Figura 2.1. Informazioni sui dati raccolti

Effettuando una rapida analisi, diventa evidente che la varianza associata a theta si avvicina mediamente a zero, con una deviazione standard approssimabile anch'essa a zero. Inoltre, la covarianza di theta con le altre variabili risulta essere trascurabile. Questi dati indicano che theta, l'orientamento dell'oggetto, presenta una variazione minima o quasi inesistente all'interno del dataset considerato.

2.2 Osservazioni conclusive

Le conclusioni tratte fino a questo punto mirano a ridurre al minimo la quantità di dati da comprimere e trasmettere, basandosi sulle caratteristiche delle matrici di

covarianza raccolte:

- Ogni termine A_{ij} in una matrice $n \times n$, sarà uguale al corrispondente termine A_{ji} per ogni $j, i \in [0, n]$. Ciò implica che una matrice $n \times n$ può essere descritta utilizzando solo $n - \frac{n^2-n}{2}$ valori unici.
- I valori delle matrici di covarianza appartengono all'insieme dei numeri reali, mentre quelli sulla diagonale principale corrispondono a varianze e sono pertanto reali positivi, consentendo l'utilizzo di variabili senza segno (unsigned) per rappresentarli.
- Nel caso specifico analizzato, è possibile trascurare i valori di varianza e covarianza relativi alla variabile theta, riducendo la matrice a soli 2×2 valori.

Utilizzando queste tecniche, è possibile rappresentare una matrice di covarianza con soli 2×2 valori, descritti completamente attraverso solamente 3 valori. Di questi, due (corrispondenti alla diagonale principale) appartengono all'insieme dei reali positivi, consentendo una compressione significativa dei dati senza perdere informazioni fondamentali per l'analisi.

Capitolo 3

Diverse soluzioni

In questo capitolo, si esploreranno e si valuteranno diverse prospettive e strategie risolutive che sono emerse nel corso della ricerca. Attraverso l'analisi di diversi approcci, si mira a evidenziare le loro peculiarità, punti di forza e limitazioni. Questo permetterà di fornire una panoramica completa delle opzioni disponibili e di determinare quali possano risultare più adatte alle specifiche esigenze.

3.1 Mappatura delle matrici attraverso una conversione di base

La matrice di covarianza, essendo per definizione simmetrica, può essere espressa come:

$$\text{mat} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix}$$

Questa rappresentazione matriciale equivale a una tupla di 3 elementi: (x_1, x_2, x_3). L'obiettivo è trattare ogni valore della tupla come una cifra in un sistema di conteggio il più ampio possibile, ciò ci permette di calcolare quanti byte occuperanno tutte le possibili permutazioni dei 3 valori.

Con 3 elementi, ci sono x^3 possibili permutazioni con ripetizione, dove x equivale a quanti diversi valori ciascun elemento della tupla può assumere. Quindi, indipendentemente dalla quantità di byte disponibili, la conversione di base sarà la radice cubica di $2^{\text{num\_bits-1}} - 1$.

Uno dei valori appartenenti a questa conversione (Covarianza tra X ed Y), può essere negativo, e richiede quindi l'uso separato di un bit di segno Ad esempio, nel caso in cui si vogliano usare 16 bit per la trasmissione della matrice compressa, i possibili valori sarebbero la radice cubica di $2^{16-1} = 32768$, ovvero 32. , La base del sistema sarà quindi 32. Per convertire la tupla (considerata come un numero a 3 cifre in un sistema di base 32) in base 10, possiamo utilizzare la seguente formula:

$$x_1 \times 32^2 + x_2 \times 32^1 + x_3 \times 32^0 = \text{Risultato decimale}$$

Dove x_1, x_2 e x_3 possono avere un valore massimo di 31.

In sintesi, per mappare tutte le possibili matrici di covarianza utilizzando uno short senza sovrapposizioni, ciascun elemento della tupla (x_1, x_2, x_3) può assumere solo 31 valori distinti, e il bit più significativo rappresenta il segno della covarianza.

Per ridurre il numero di bit necessari per memorizzare la tupla, è possibile evitare di mappare tutti i possibili valori della matrice e approssimarli ad un certo grado.

Minore è l'approssimazione, maggiore sarà la precisione della conversione, ma più bit saranno richiesti per memorizzare la tupla.

Ad esempio, si potrebbe considerare l'utilizzo solo di valori pari e arrotondare per eccesso ogni valore dispari. Questo approccio comporta una perdita di precisione di 1mm, ma raddoppia l'intervallo dei valori considerati della matrice.

Il funzionamento di queste funzioni di encoding/decoding, in grado di comprimere la matrice data in input in uno short, o di decomprimere uno short dato in input in una matrice.

Sono state inoltre scritte una funzione di normalizzazione ed una di denormalizzazione, le quali permettono di mappare valori compresi in modulo tra 0 e il valore più grande trovato nel dataset, su una quantità arbitraria di valori predefiniti.

Algorithm 1 Funzione NORMALIZE

function NORMALIZE(matrice, maxX, maxY, maxCov)
    Crea una nuova matrice vuota
    Assegna alla posizione $(0, 0)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore nella posizione$(0, 0)$della matrice in input permaxX
        −Moltiplica il risultato per il valore massimo prefissato(MAX_VALUE)
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(0, 1)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore nella posizione$(0, 1)$della matrice in input permaxCov
        −Moltiplica il risultato per il valore massimo prefissato(MAX_VALUE)
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(1, 0)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore nella posizione$(1, 0)$della matrice in input permaxCov
        −Moltiplica il risultato per il valore massimo prefissato(MAX_VALUE)
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(1, 1)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore nella posizione$(1, 1)$della matrice in input permaxY
        −Moltiplica il risultato per il valore massimo prefissato(MAX_VALUE)
        −Tronca il risultato a un numero intero
    Restituisci la nuova matrice normalizzata
end function

Algorithm 2 Funzione REVERSE

function REVERSE(matrice, maxX, maxY, maxCov)
    Crea una nuova matrice vuota
    Assegna alla posizione $(0, 0)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore massimo prefissato(MAX_VALUE)per il valore nella posizione$(0, 0)$
        −Moltiplica il risultato permaxX
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(0, 1)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore massimo prefissato(MAX_VALUE)per il valore nella posizione $(0, 1)$
        −Moltiplica il risultato permaxCov
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(1, 0)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore massimo prefissato(MAX_VALUE)per il valore nella posizione$(1, 0)$
        −Moltiplica il risultato permaxCov
        −Tronca il risultato a un numero intero
    Assegna alla posizione $(1, 1)$ della nuova matrice il risultato dell'operazione:
        −Dividi il valore massimo prefissato(MAX_VALUE)per il valore nella posizione$(1, 1)$
        −Moltiplica il risultato permaxY
        −Tronca il risultato a un numero intero
    Restituisci la nuova matrice invertita
end function

un esempio di output a seguito dell'uso corretto di entrambe le funzioni è:
Matrice originale:

$$\begin{bmatrix} 1050 & 0 \\ 0 & 1530 \end{bmatrix}$$

Matrice normalizzata:

$$\begin{bmatrix} 18 & 0 \\ 0 & 27 \end{bmatrix}$$

Valore unico: 0100 0011 1010 1101
Matrice decodificata:

$$\begin{bmatrix} 18 & 0 \\ 0 & 27 \end{bmatrix}$$

Matrice denormalizzata:

$$\begin{bmatrix} 1032 & 0 \\ 0 & 1521 \end{bmatrix}$$

MSE: 130.00
In questo caso risulta evidente che il valore originale della varianza di x, ovvero 1050, è stato normalizzato al valore 18, uno dei 32 valori disponibili. questo una volta denormalizzato viene ricondotto a 1032, ottenendo una minima perdita di informazione.

3.2 Mappatura mediante un dizionario ordinato

L'approccio proposto in questa seconda strategia punta a raffinare il precedente; Invece di ricorrere ad una conversione di base che copra con precisione ideale tutte le possibili matrici con una ridotta portata, si propone di utilizzare i dati precedentemente ottenuti dalle simulazioni di partite reali per ottenere un dizionario ordinato.

L'idea è quella di analizzare e ordinare il dataset di matrici (ottenuto dalla simulazione di una partita reale), in base al numero di volte in cui ciascuna matrice appare, questo, per evidenziare i valori più comuni e frequenti, consentendo di approssimare le matrici meno comuni a quelle più frequenti con la minor differenza; diminuendo quindi l'accuratezza della compressione ma aumentando la portata dei valori contenuti nella matrice.

Al fine di potenziare l'efficacia di questo approccio, risulta vantaggioso raggruppare le matrici che presentano differenze minime(+/-5) attraverso l'organizzazione in dei bin, eseguendo così un'ulteriore approssimazione. Questa azione consente di accrescere ulteriormente la rappresentatività delle matrici selezionate.

Operativamente, è stato istanziato un dataframe utilizzando i dati generati nel capitolo precedente. In seguito, mediante l'impiego delle funzioni messe a disposizione dalla libreria Pandas, i dati sono stati aggregati ed è stata eseguita un'operazione di conteggio.

Attraverso la classificazione di tali dati in base alla loro frequenza, è stato agevolato l'impiego di una metodologia di encoding alternativa. Questa metodologia si basa sull'approssimazione della matrice originale a una delle $2^{\text{num\_bits-1}} - 1$ matrici più frequenti contenute nel dataframe. Il processo di selezione della matrice più idonea avviene attraverso un'iterazione sul dataframe, estraendo quella con la minima distanza euclidea dalla matrice originale.

Approssimando le due matrici, ai corrispettivi vettori,(come visto nei precedenti capitoli, senza perdita di informazioni), otteniamo v_1 e v_2 rappresentati come:

$$v_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

La formula per ottenere la distanza euclidea tra v_1 e v_2 è data da:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Dove x_1, y_1, z_1 sono le componenti di v_1, e x_2, y_2, z_2 sono le componenti di v_2.

La nuova funzione di encoding consiste quindi nell'ottenere l'indice della matrice più 'simile':

Algorithm 3 Funzione FindClosestArrayIndex

```
1: function FindClosestArrayIndex(data, target_data)
2:     Calcola la distanza euclidea tra ogni riga del dataset e il vettore target
3:     Inizializza una lista vuota di distanze distances for ogni riga row in data do
4:
       end
       Calcola la distanza euclidea tra row e target_data
5:     Aggiungi la distanza calcolata alla lista distances
6:
7:     Trova l'indice del minimo valore nella lista distances
8:     Assegna il valore dell'indice minimo trovato alla variabile min_index
9:     Ritorna min_index
10: end function
```

Inizialmente, la procedura prevede la lettura del file .csv e la traduzione del suo contenuto in una lista di vettori, ognuno dei quali rappresenta una famiglia di matrici. Successivamente, attraverso un'iterazione su tutte le matrici, viene individuata quella con la minore distanza euclidea. Infine, la funzione restituisce l'indice associato a tale matrice e la distanza corrispondente. Quest'ultimo passo è essenziale per valutare la quantità di perdita di precisione registrata.

3.3 Uso di una rete Neurale

La terza strategia proposta per la risoluzione del problema si basa sull'utilizzo di una rete neurale, nello specifico un autoencoder, per la compressione e la decompressione delle matrici di covarianza. L'autoencoder è un tipo di rete neurale artificiale utilizzata per apprendere la rappresentazione di un insieme di dati attraverso un processo di compressione e ricostruzione.

L'autoencoder è composto da due componenti principali: un encoder, che converte l'input in una rappresentazione compressa, e un decoder, che ricostruisce l'input originale dalla rappresentazione compressa.

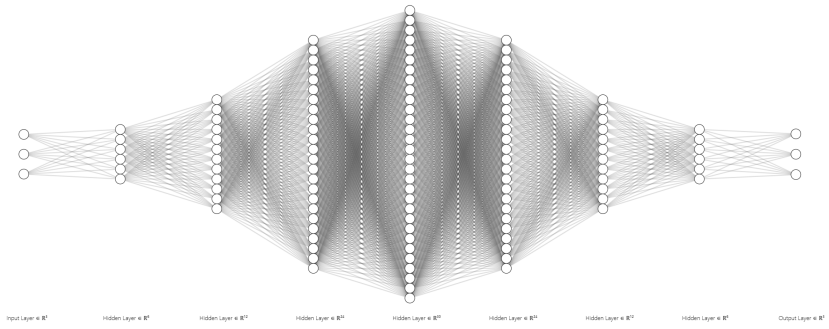

Figura 3.1. Diagramma della rete neurale utilizzata

Dalla figura soprastante è possibile comprendere la quantità di layer presenti nella rete e le loro proporzioni; Tuttavia il numero di neuroni presenti in ciascuno strato non è rappresentato correttamente in scala. Di fatto la rete utilizzata per ottenere i successivi risultati ha la seguente dimensione:

- input layer = 3 -> 64 -> 256 -> 4096 -> 16384 = encoding layer
- decoding layer = 16384 -> 4096 -> 256 -> 64 -> 3 = output layer

Durante il processo di addestramento, l'autoencoder cerca di minimizzare l'errore di ricostruzione tra l'input originale e l'output ricostruito.

Una volta addestrato, l'encoder ed il decoder possono essere utilizzati separatamente per comprimere le matrici in una rappresentazione a dimensionalità ridotta e successivamente decomprimerle per ripristinare l'input originale.

L'encoder una volta isolato ha la struttura di un classificatore, presa una matrice in input si occupa trovarne una rappresentazione adeguata e classificarla in base ad essa, maggiore la quantità di classi messe a disposizione maggiore sarà la precisione con cui la rete riesce a distinguere le diverse matrici.

Il decoder invece, una volta isolato ha il comportamento opposto all'encoder, cioè impara quali matrici sono più significative per ciascuna classe.

Essendo in questo modello presenti 16384 classi il valore dell'encoding sarà un intero compreso tra 1 e 16384, ovvero rappresentabile tramite un valore binario a 14 bit.

L'autoencoder è in grado di apprendere in modo automatico le migliori rappresentazioni per le matrici di covarianza, senza richiedere una progettazione manuale delle funzioni di compressione. Inoltre, l'autoencoder è in grado di gestire in modo efficiente le variazioni e le complessità presenti nei dati, garantendo una compressione efficace e una ricostruzione accurata delle matrici di covarianza.

Capitolo 4

Valutazione

La presenza di vari approcci risolutivi implica la necessità di condurre un confronto tra di essi. Per questo motivo, è stata introdotta una fase aggiuntiva di test, durante la quale sono state valutate tre diverse metriche di qualità: l'errore quadratico medio, la quantità di bit occupati dal dato compresso ed infine il tempo di esecuzione.

questi tre parametri sono stati estrapolati attraverso l'applicazione operativa delle funzioni viste nei capitoli precedenti, su dei campioni di dati ottenuti in una simulazione indipendente da quella usata in precedenza.

4.1 valutazione conversione di base

La valutazione di questo approccio è resa possibile dal seguente algoritmo:

Algorithm 4 Main

Ottieni i dati
Calcola i valori massimi $maxX$, $maxY$ e $maxCov$ dal dataset
Inizializza una lista vuota di perdite *losses*
Inizia a registrare il tempo **for** *ogni matrice i nel dataset di test* **do**
    end
    Estrai la matrice originale dal dataset
Normalizza la matrice utilizzando i massimi delle rispettive feature
Codifica e **normalizza** la matrice
Decodifica e **denormalizza** il risultato
Calcola il vettore $v1$ dalla matrice originale e il vettore $v2$ dalla matrice denormalizzata
Calcola l'errore quadrato medio tra $v1$ e $v2$ e aggiungilo alla lista delle perdite

Ferma il timer e calcola il tempo di inferenza
Calcola la media delle perdite

Nel metodo proposto, il tempo di inferenza viene determinato mediante l'utilizzo della funzione `time()`, che viene invocata all'inizio e alla fine dell'iterazione dei dati di test.

La precisione dell'algoritmo viene valutata calcolando l'errore quadratico medio tra i componenti delle due matrici, quella originale e quella ricostruita.

La quantità di bit necessaria per la compressione è stabilita in modo arbitrario tramite la costante NUM_BITS. I risultati ottenuti da ciascuna valutazione saranno successivamente sintetizzati in un resoconto collettivo.

4.2 valutazione del dizionario ordinato

La valutazione di questo approccio è resa possibile dal seguente algoritmo:

Algorithm 5 Script principale

1: Carica il dataset di validazione e il vettore target
2: Carica il dataset principale
3: Inizializza una lista vuota per memorizzare le perdite
4: Inizializza il tempo di inizio
5: Avvia un pool di thread esecutori
6: Crea una lista di futures per elaborare le righe del dataset di validazione **for** *ogni riga i nel range(2000) del dataset di validazione* **do**
7:
    end
    Aggiungi un futuro alla lista: il thread elaborerà la riga i del dataset
8:
9: Attendere il completamento dei futures **for** *ogni futuro completato nella lista* **do**
10:
    end
    Aggiungi il risultato alla lista delle perdite
11:
12: Calcola il tempo impiegato per l'inferenza
13: Calcola il MSE (Mean Squared Error) utilizzando le perdite
14: Stampare il tempo impiegato per l'inferenza
15: Stampare il MSE

In questo algoritmo, il tempo di inferenza viene determinato mediante l'utilizzo della funzione `time()`, che viene invocata all'inizio e alla fine dell'iterazione dei dati di test.

La precisione dell'algoritmo (come in precedenza), viene valutata calcolando l'errore quadratico medio tra i componenti delle due matrici, quella originale e quella ricostruita.

La quantità di bit necessaria per la compressione è derivata dalla dimensione del dizionario scelto, in quanto il dato compresso rappresenza un indice del dizionario, fino a rappresentarne l'indice maggiore.

4.3 valutazione della rete neurale

La valutazione della rete viene svolta su dei dati indipendenti da quelli usati per il suo training, seguendo le stesse metriche viste finora.
Come in passato, il tempo di inferenza viene determinato mediante l'utilizzo della funzione `time()`, che viene invocata all'inizio e alla fine dell'iterazione dei dati di test.

La precisione dell'algoritmo equivale alla funzione di loss usata per il training, per fini comparativi è stata scelto di usare l'errore quadratico medio, così da rendere questo dato paragonabile alle altre soluzioni proposte.

Algorithm 6 Valutazione del Modello

Carica i dati da 'validationData.txt' in un formato appropriato per il dataset
Inizializza una lista vuota per memorizzare la funzione di loss
Inizia il cronometro per calcolare il tempo impiegato per l'inferenza **for** *ogni voce nel dataset* **do**

    end
    Ottieni l'output dell'encoder applicando la voce del dataset
Ottieni l'output del decoder applicando l'encoding calcolato
Calcola la funzione di perdita tra l'output del decoder e l'input reale
Aggiungi la perdita alla lista delle perdite

Ferma il cronometro e calcola il tempo totale impiegato per l'inferenza

La quantità di bit necessaria per la compressione è uguale alla quantità di neuroni che si scelgono di inserire nell'ultimo strato di encoding della rete.

Capitolo 5

Risultati

5.1 Tabelle

I dati ottenuti dalle precedenti rilevazioni sono stati schematizzati organizzati e riassunti nelle seguenti tabelle. Per ogni approccio vengono analizzate diverse varianti della soluzione, ciascuna con le sue caratteristiche.

	conversione a 10 bit	conversione a 16 bit	conversione a 32 bit
mse	2677	643	502
inference time	0.31	0.33	0.29
dati trasmessi	10	16	31

Figura 5.1. Risultati del primo approccio

	dizionario a 10 bit	dizionario a 16 bit	dizionario a 32 bit
mse	1805	7.46	0
inference time	0.077	5.7	16.05
dati trasmessi	10	16	32

Figura 5.2. Risultati del secondo approccio

	rete neurale a 10 bit	rete neurale a 14 bit	
mse	10453	13	
inference time	0.2	5.2	
dati trasmessi	10	14	

Figura 5.3. Risultati del terzo approccio

5.2 Grafici a barre

Con l'intenzione di riassumere i dati raccolti, e trarne delle conclusioni valide, sono riportati qui sotto dei grafici a barre contenti i risultati e le prestazioni delle diverse soluzioni.

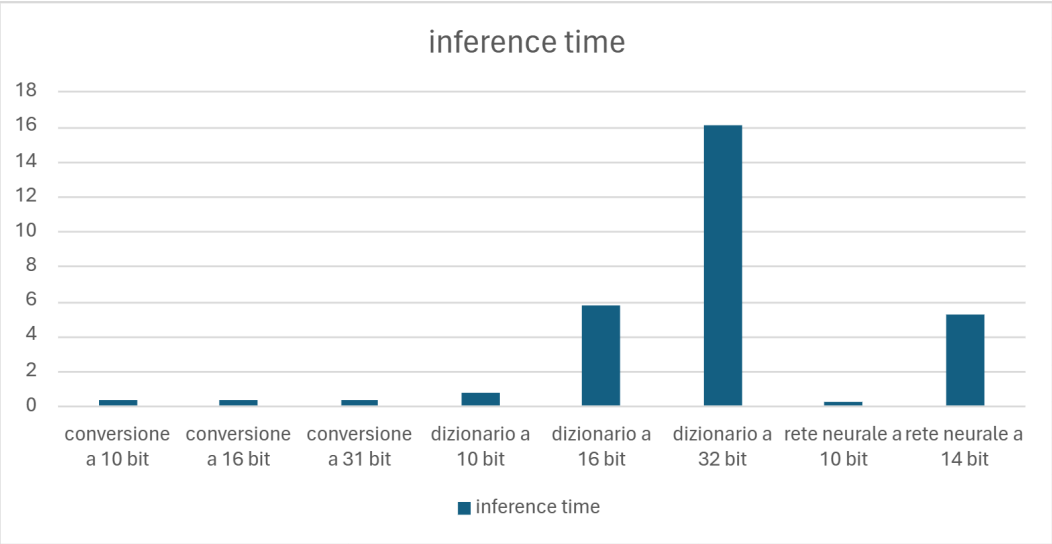

Figura 5.4. Tempo impiegato per processare il test set

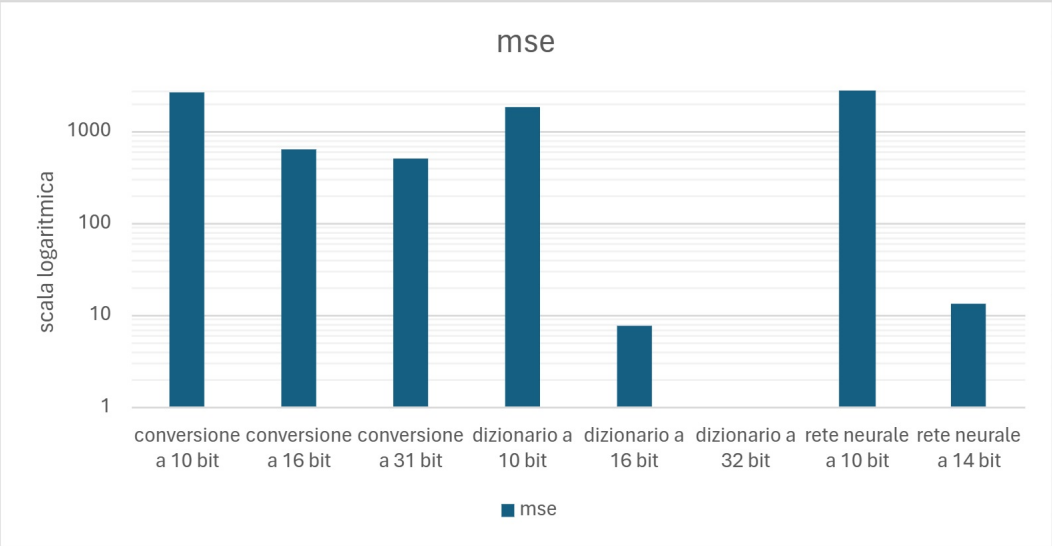

Figura 5.5. Errore commesso sul test set

Capitolo 6

Conclusioni

In seguito ad un'analisi dei grafici appena forniti, si può fare una distinzione tra i diversi approcci, in particolare uno studio dei punti deboli e punti di forza di ciascuno di essi:

- **Conversione di base:**
  - Pro: Approccio più rapido, richiede un impegno computazionale minore.
  - Contro: Per ottenere un indice di perdita utilizzabile richiede la trasmissione di più informazioni rispetto ad altri approcci
- **Dizionario ordinato:**
  - Pro: Approccio più accurato, permette di ottenere un errore rasente allo zero con la trasmissione di meno informazioni
  - Contro: Richiede che nella memoria sia salvata una matrice di dimensioni non trascurabili, e che questa venga ogni volta analizzata totalmente, rendendo questo algoritmo computazionalmente esigente.
- **Rete Neurale**
  - Pro: Similarmente al Dizionario ordinato, si tratta di un approccio più accurato, permette di ottenere un errore molto basso attraverso la trasmissione di pochi bit. A differenza del dizionario ordinato La rete neurale è in grado di adattarsi a qualunque genere di dataset senza un analisi specifica.
  - Contro: Richiede l'istanziamento nella memoria di una rete neurale con i relativi pesi, le dimensioni di questi dati non sono trascurabili.

Capitolo 7

Altri applicativi

Oltre al ruolo che questa ricerca svolge sul campo da calcio per il team S.P.Q.R. è possibile definire altri possibili applicativi, che sfruttino a pieno l'uso delle matrici di covarianza necessitando delle dimensioni ridotte.

- Comunicazioni wireless e reti di sensori: Nelle reti di sensori e nelle applicazioni di comunicazione wireless, è essenziale trasmettere dati in modo efficiente a causa delle limitazioni di banda e di energia. La compressione delle matrici di covarianza potrebbe essere utile per ridurre il carico di trasmissione dei dati senza compromettere la qualità delle informazioni trasmesse.
- Reti neurali e apprendimento automatico: Nei modelli di reti neurali e nell'apprendimento automatico, le matrici di covarianza possono essere utilizzate per modellare la correlazione tra le caratteristiche di input. La compressione delle matrici di covarianza potrebbe portare a una riduzione della complessità computazionale e dei requisiti di memoria durante l'addestramento e l'inferenza dei modelli.
- Edge computing: Nell'ambito dell'edge computing, l'analisi dei dati viene eseguita direttamente sui dispositivi periferici anziché su server remoti, al fine di ridurre la latenza e migliorare l'efficienza delle applicazioni. Le matrici di covarianza possono essere utilizzate per analizzare i dati provenienti da sensori distribuiti in una rete edge. La compressione delle matrici di covarianza potrebbe essere impiegata per ottimizzare l'uso delle risorse computazionali e ridurre i costi di trasmissione.

Bibliografia

[1] Michelucci, Umberto. *An Introduction to Autoencoders*. arXiv preprint arXiv:2201.03898 (2022).

[2] Antonioni, Emanuele, et al. *Game Strategies for Physical Robot Soccer Players: A Survey*. IEEE Transactions on Games, vol. PP, pp. 1-1, Apr. 2021, doi: 10.1109/TG.2021.3075065.

Ringraziamenti

Certo! Ecco un testo di ringraziamento per la tua tesi:

—

Vorrei ringraziare con tutto il cuore un gruppo speciale di persone che hanno reso possibile il completamento di questo viaggio accademico, un percorso fatto di sfide, ma anche di incredibili momenti di crescita e soddisfazione.

Innanzitutto, non posso che iniziare con un ringraziamento speciale alla mia amata Miriam. Grazie per essere stata la mia roccia, il mio faro luminoso durante i momenti bui e la mia più grande fonte di ispirazione. Senza il tuo amore incondizionato e il tuo sostegno costante, questo traguardo sarebbe stato molto più difficile da raggiungere. Sei il mio amore eterno.

Ai miei fantastici amici Federico, Matteo, Luca, Dennis e Aleida: voi siete il tesoro più prezioso che la vita mi ha donato. Grazie per avermi sostenuto, incoraggiato e fatto ridere quando ne avevo più bisogno. Le risate condivise, le avventure folli e le lunghe chiacchierate sono state la linfa vitale di questi anni universitari.

Un ringraziamento speciale va anche a Camilla, Federica, Carlo, Lorenzo, Chiara, Alessandro, Claudio, Karen, Jack e Pietro, i miei compagni di università. Insieme abbiamo affrontato esami difficili, progetti complicati e notti insonni, ma anche creato ricordi indelebili e stretto legami che vanno ben oltre il mondo accademico.

Non posso dimenticare di menzionare mia sorella Federica, che con il suo sostegno incondizionato e il suo esempio di determinazione mi ha sempre spronato a dare il meglio di me stesso.

Infine, un grazie va a mia madre Mariateresa. Grazie per essere sempre stata la mia più grande sostenitrice, per aver creduto in me quando nemmeno io ci credevo e per avermi insegnato il valore del sacrificioa.

Vi porto nel mio cuore con infinita gratitudine.

[Tuo nome]

—

Se hai bisogno di ulteriore assistenza o di altro supporto, non esitare a chiedere.