POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Gestionale

Tesi di Laurea Magistrale

Cinema e Intelligenza Artificiale: metodologia non supervisionata per l'analisi di sequenze di inquadrature cinematografiche

Relatori
Prof.ssa Tania Cerquitelli
Ing. Bartolomeo Vacchetti

Candidato
Simone Magnafico

Anno Accademico 2020/2021

Indice

Abstarct

Il presente lavoro di ricerca ha lo scopo di analizzare sequenze di inquadrature cinematografiche mediante l'utilizzo di algoritmi e metodologie tipiche dell'Unsupervised Learning. In particolare, diversi algoritmi di Clustering sono stati utilizzati al fine di individuare pattern ricorrenti all'interni del dataset appositamente creato per lo studio. Prendendo il via dai risultati di un precedente lavoro di ricerca nell'ambito della classificazione di inquadrature cinematografiche presente in letteratura, per ciascun frammento di film sono state individuate le scene salienti ed estrapolati i frame, che una volta classificati, hanno fornito per ogni clip cinematografica una sequenza di inquadrature di vario tipo. Le classi di inquadrature considerate per lo studio delle sequenze sono otto: campo lungo, campo medio, figura intera, piano Americano, mezza figura, mezzo busto, primo piano e primissimo piano. Il dataset, così creato, è stato poi integrato con informazioni circa la durata delle clip, la durata media di ciascun frame e il genere del film in esame, scelto tra Drama, Action, Comedy ed Horror. L'obiettivo ultimo dell'analisi è stato, dunque, individuare tra le sequenze eventuali pattern significativi in grado di caratterizzare un determinato genere e, ancora, valutare eventuali correlazioni tra le altre metriche riportate e genere stesso del film di riferimento. Vengono riportartati, infine, i risultati sperimentali dell'analisi.

Capitolo 1

Introduzione

Il seguente lavoro di tesi è nato dalla felice collaborazione con la Prof.ssa Tania Cerquitelli, docente del corso di laurea magistrale in Ingegneria Gestionale, e con l'Ing. Bartolomeo Vacchetti; rispettivamente Relatrice e Correlatore dell'elaborato. Lo studio rappresenta la naturale continuazione di un precedente progetto di ricerca nell'ambito della classificazione delle inquadrature cinematografiche svolto dall'Ing. Bartolomeo Vacchetti e dalla Prof.ssa Tania Cerquitelli. In esso ci si è posto l'obiettivo di analizzare sequenze di inquadrature cinematografiche secondo un approccio Unsupervised. Nello specifico, diversi algoritmi di clustering sono stati testati con lo scopo di ottenere gruppi di sequenze, o clusters, omogenei rispetto al genere cinematografico di appartenenza. Inoltre, al fine di estrapolare una maggiore conoscenza utile dai dati, si è tentato di caratterizzare i risultati del clustering alla ricerca di regole in grado di evidenziare pattern ricorrenti all'interno delle sequenze, riconducibili ad una determinata tecnica narrativa o ad un preciso stile di racconto.

L'elaborato si articola in sei Capitoli.

Nel Capitolo 2 viene fornita, in maniera sintetica, una panoramica sulle diverse metodologie del Machine Learning, con particolare focus sulle classi di algoritmi esistenti e sull'intuizione che si cela dietro di esse. Segue, poi nel Capitolo 3, una più profonda trattazione degli algoritmi e dei metodi di Clustering utilizzati per lo studio in questione. Il quarto capitolo, invece, introduce il lettore alla classificazione delle inquadrature cinematografiche nelle otto classi maggiormente utilizzate nel mondo

del cinema, ossia: campo lungo, campo medio, figura intera, piano Americano, mezza figura, mezzo busto, primo piano e primissimo piano. Si entra, poi, nel vivo dell'analisi all'interno del Capitolo 5, nel quale viene snocciolata nel dettaglio la pipeline di analisi dei dati, partendo con le tecniche e le fonti che hanno reso possibile la costruzione del dataset e procedendo con i diversi approcci utilizzati in fase sperimentale per analizzare le sequenze. La trattazione degli esperimenti viene accompagnata dall'interpretazione dei risultati. L'elaborato si conclude con il Capitolo 6 in cui vengono sintetizzati i risultati e vengono proposti alcuni spunti di analisi future da implementare.

Capitolo 2

Intelligenza Artificiale e Machine Learning

2.1 AI e Rivoluzione Digitale

Negli ultimi decenni le innovazioni in ambito tecnologico hanno portato a quella che oggi gli studiosi definiscono "Rivoluzione Digitale" o "Quarta Rivoluzione Indutriale". Uno dei più importanti campi alla base di tale Rivoluzione è sicuramente quello dell'Intelligenza Artificiale o AI (dall'inglese Artificial Intelligence).

Le radici culturali dell'AI sono antichissime; addirittura l'idea di un primo automa compare nel mito greco di Talo, un gigantesco automa di Bronzo mandato da Zeus a sorvegliare l'isola di Creta. Il dibattito e la speculazione filosofica sulle "macchine intelligenti" ha attraversato, poi, anche tutto il periodo moderno. Su di esse, infatti, si erano interrogati filosofi come Cartesio, Hobbes e, soprattutto, Pascal, il quale realizzò diversi prototipi di calcolatori meccanici.

Il ricco dibattito scientifico e filosofico sulla possibilità di costruire o creare macchine in grado di replicare l'intelligenza umana raggiunse il suo culmine nel 1950, quando il pionere dell'informatica moderna Alan Turing, riprendendo il "Cogito" di Cartesio, iniziò a porre la domanda "Le macchine possono pensare?". Per rispondere a tale interrogativo Turing elaborò il suo celebre test descritto nell'articolo *Computing machinery and intelligence*, pubblicato nel 1950 sulla rivista *Mind*. Il

test di Turing si rifa ai cosiddetti "giochi dell'imitazione", nei quali un soggetto, definito "giudice" deve cercare di stabilire il sesso di due altri interlocutori, un uomo e una donna, ponendo loro alcune domande. In tali giochi, poi, il giudice è tenuto separato dai due partecipanti e non disponde di alcun indizio. Dal canto loro, i due interlocutori hanno il compito di ingannare il soggetto giudice, il primo, e di aiutarlo, l'altro. Sulla falsa riga dei giochi dell'imitazione Turing teorizza che se si sostituisse una macchina ad uno dei due interlocutori e se la percentuale di volte in cui il soggetto stabilisce correttamente il sesso dei due interlocutori è simile sia prima che dopo la sostituzione di uno di essi con una macchina, allora la macchina è da considerarsi intelligente, in quanto indistinguibile da un umano [1]. Da ciò si evince come per Turing l'intelligenza e la capacità di riprodurre funzioni umane siano da ricercarsi nella complessità computazionale del software. Negli anni tale test è stato contestato da numerosi membri della comunità scientifica a causa dell'imprecisione nella formulazione originale del concetto di macchina intelligente, in quanto anche programmi semplici sembravano soddisfare i criteri di Turing (in tal senso è celebre il caso di ELIZA, un software che emula il comportamento di uno psicoterapeuta). La più famosa di queste rivisitazioni è stata elaborata dal filosofo John Searle, il quale riteneva inattendibili i criteri di Turing poichè incapaci di fornire prova sufficiente per dimostrare la vera intelligenza di una macchina o di un sistema informatico che, a detta di Searle, non potranno mai replicare la "coscienza" umana.

L'acceso dibattito scientifico, tecnologico e filosofico ha portato, poi, a metà degli anni '50, gli informatici John McCarthy, Marvin Minsky, Herbert Simon e AlLen Newell a coniare per la prima volta l'espressione "Artificial Intelligence" nell'ambito di una conferenza accademica al Dartmouth College negli Stati Uniti e a dare un forte impulso alla disciplina informatica e tecnologica che è arrivata fino ai giorni nostri.

Ora, come visto, dare una definizione univoca e generale di AI è molto complesso. Tuttavia, possiamo guardare all'intelligenza artificiale come un insieme di tecnologie differenti che interagiscono per consentire alle macchine di percepire, comprendere, agire e apprendere con livelli di intelligenza simili a quelli umani. L'idea

di fondo, quindi, è quella di sviluppare delle macchine dotate di capacità autonome di apprendimento e adattamento che siano ispirate ai modelli di apprendimento umani. Dal punto di vista informatico e tecnologico possiamo definire l'AI come il ramo della computer science che studia lo sviluppo di sistemi Hardware e Software dotati di specifiche capacità tipiche dell'essere umano (interazione con l'ambiente, apprendimento e adattamento, ragionamento e pianificazione), capaci di perseguire autonomamente una finalità definita, prendendo decisioni che fino a quel momento erano solitamente affidate alle persone.

Il concetto di AI muove da due teorie distinte, affermatesi a partire dagli anni '80. I ricercatori e gli studiosi distinguono tra Intelligenza Artificiale "forte" e "debole" a seconda che vengano riprodotte tutte le funzioni e le facoltà tipiche di una mente umana o solamente alcune di esse. La maggior parte della applicazioni che vediamo nella nostra vita quotidiana rientrano nel concetto di AI debole. Tale filone ritiene possibile creare macchine in grado di svolgere operazioni complesse riproducendo una o più funzioni umane al fine di risolvere problemi specifici senza, però, sviluppare una coscienza. L'intelligenza artificiale generale, invece, è più simile a ciò che si vede nei film di fantascienza, dove macchine senzienti emulano l'intelligenza umana, pensando in modo strategico, astratto e creativo, con la capacità di gestire una serie di compiti complessi. Secondo questa corrente, quindi, l'obiettivo è costruire macchine in grado di sviluppare coscienza di sè. Ad oggi,nonostante le macchine possano svolgere alcune attività meglio degli esseri umani (per esempio l'elaborazione dei dati), questa visione pienamente realizzata dell'AI generale nella realtà non esiste ancora. Ecco perché la collaborazione uomo-macchina è cruciale: nel mondo di oggi, l'intelligenza artificiale rimane un'estensione delle capacità umane, non un sostituto.

2.2 Machine Learning

Il Machine Learning, o in italiano Apprendimento Automatico, rappresenta una delle più floride branche dell'Intelligenza Articifiale. Grazie alle nuove tecnologie che offrono ingenti capacità computazionali e alla grande mole di dati disponibile

oggi giorno, esso sta rivoluzionando il modo in cui lo sviluppo di software viene pensato e realizzato. Tale disciplina affonda le proprie radici nella seconda metà degli anni '50, quando Arthur Lee Samules, un ricercatore presso IBM, coniò per la prima volta l'espressione "Macine Learning" [2]. Oggi il campo dell'Apprendimento Automatico raccoglie una varietà di tecniche e di algoritmi provenienti da diversi ambiti scientifici, come ad esempio la Statistica Computazionale, il Data Mining, Image Processing e così via.

Nello specifico, il Machine Learning è una forma di AI, capace di rendere sistemi in grado di apprendere dai dati forniti in input piuttosto che dalla programmazione esplicita, come avveniva per gli algoritmi tradizionali [3].

Il valore aggiunto del Machine Learning, dunque, risiede nella capacità di tali tecniche ed algoritmi di imparare costantemente dai dati al fine di risolvere uno specifico task o di fare una predizione sul futuro senza l'intervento umano.

Gli algoritmi di Machine Learning, dunque, per poter produrre modelli precisi da utilizzare per risolvere task specifici o fare previsioni accurate necessitano di una grande quantità di dati, che oggi nell'era dei Big Data, sono sempre più accessibili e alla portata. Maggiori sono i dati di training, ossia dati sui quali tali algoritmi sono "addestrati", e più è possibile produrre modelli precisi in grado di compiere previsioni accurate.

Le applicazioni del Machine Learning sono molteplici e molte di esse sono già entrate stabilmente a far parte della nostra vita quotidiana. Tra le più note e familiari anche ai non addetti ai lavori abbiamo indubbiamente il ranking delle pagine web o il collaborative filtering in cui un motore di ricerca, nel primo caso, o e-commerce e piattaforme di streaming, nel secondo, mirano ad ottenere una lista ordinata di pagine web o prodotti da consigliare all'utente/cliente senza effettuare manualmente una query ma sfruttando decisioni o acquisti passati per predire visioni future o abitudini di acquisto. In questi casi, le informazioni chiave vengono dedotte dalle scelte di utenti "simili", da cui quindi la natura "collaborativa" di tali processi. [4]

Questi due esempi mostrano , dunque, come gli algoritmi di Machine Learning

siano in grado di apprendere mediante l'esperienza e migliorare le proprie perfomances nel risolvere compiti specifici o nell'effettuare previsioni all'aumentare di essa.

Da qui scaturisce, quindi, la differenza chiave tra un algortimo tradizionale ed uno di Machine Learning. Se, infatti, il primo viene programmato per restituire un determinato output definito un determinato input, il secondo, prendendo in input dati e risposte al problema in esame, è in grado di apprendere dai dati e individuare le regole e i pattern sconosciuti che li caratterizzano al fine poi di effettuare una previsione o portare a termine un dato task.

2.3 Principali tipologie di Machine Learning

Gli algoritmi di Machine Learning vengono suddivisi in diverse macro-aree a seconda dei task e degli obiettivi perseguiti. In particolare, i modelli vengono raggruppati in tre categorie base: *Supervised Learning* , *Unsupervised Learning* , *Reinforcement Learning*. Una menzione particolare meritano, poi, l'*Ensemble Learning* ed il *Deep Learning*, i quali combinano algoritmi e metodi tipici delle prime tre categorie per poter risolvere problemi complessi.

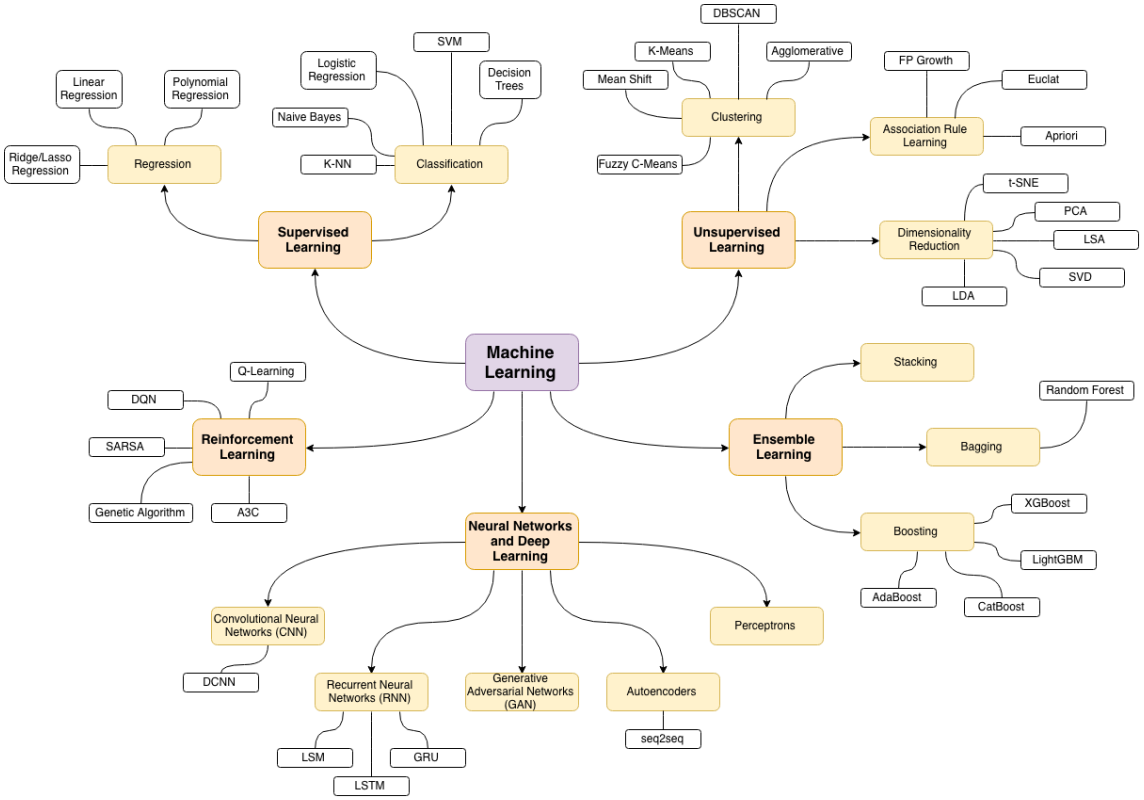

Figura 2.1: Diverse tipologie di Machine Learning

2.4 Supervised Learning

Nel Supervised Learning, o anche Apprendimento Supervisionato, l'analisi viene effettuata a partire da un set di dati dei quali viene fornita all'algoritmo la classificazione. Ciò significa che l'algoritmo conosce le etichette di classe dei dati forniti in input nella fase di addestramento. Gli algoritmi di Supervised Learning, dunque, infieriscono la funzione matematica tale da mappare gli input negli output per ciascuna coppia input-output [5]. Una volta individuata la funzione matematica sottostante i dati forniti in input durante la fase di training, viene utilizzato il modello addestrato per effettuare previsioni su dati non etichettati di test in modo da valutare le performance dell'algoritmo mediante, principalmente, l'*accuracy*. Occasionalmente, alcuni patterns che vengono identificati in piccoli dataset di training potrebbero non essere identificati in porzioni di più grandi di dataset. Quando ciò

avviene, l'algoritmo ricalca troppo fedelmente i pattern individuati nel dataset di training finendo per introdurre errore quando applicato a dati in fase di test. Questo fenomeno prende il nome di *Overfitting*. Al fine di evitare di sfociare nell'*Overfitting* è, quindi, fondamentale testare il modello su dati non etichettati.

Nel campo dell'Apprendimento Supervisionato si possono individuare due gruppi di applicazione: Classificazione e Regressione. Quando le etichette sono continuee si effettua una Regressione mentre quando l'output su cui si vuole fare la previsione è discreto allora si parla di Classificazione. Una Regressione, quindi, effettua una previsione su un output continuo. In particolare, essa può essere utile per valutare la correlazione tra più variabili. Alcuni esempi di applicazione di una regressione nell'ambito di problemi di Supervised Learning possono essere la previsione del meteo dati i pattern storici e le attuali condizioni, o ancora cercare di stabilire se vi è una relazione tra numero di alunni in una classe scolastica e risultato nei test ecc.

Nei problemi di Classificazione, invece, per ciascuna osservazione si cerca di predire una risposta qualitativa (o output discreto). Esempi di task di Classificazione sono i filtri spam delle mail o ancora la fraud detection, operata dalle banche per individuare le transazioni fraudolente, e perfino l'individuazione di mutazioni del DNA capaci di causare malattie ecc.

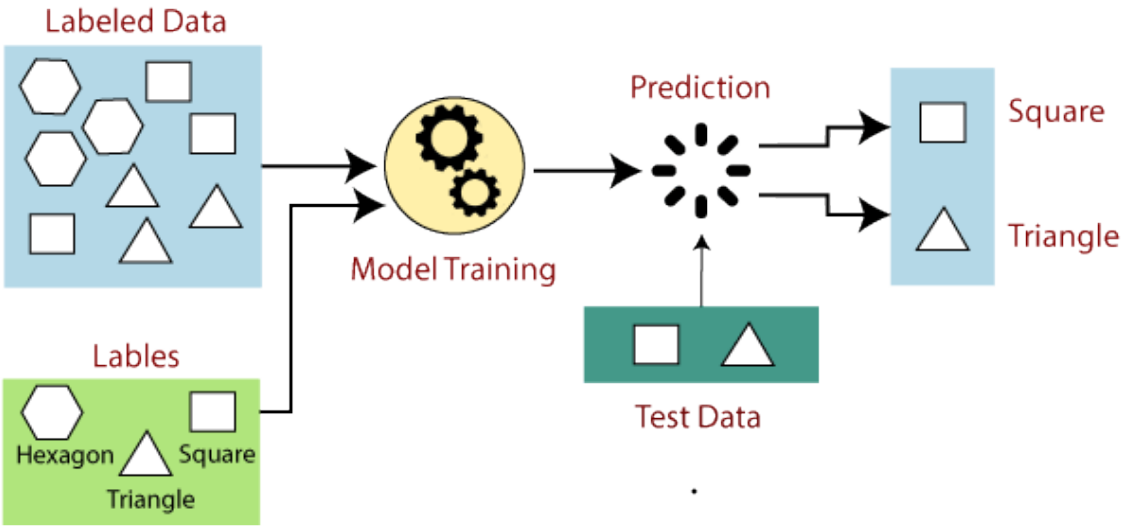

Figura 2.2: Esempio di processo di Supervised Learning

2.5 Unsupervised Learning

Nell'Unsupervised Learning, a differenza di quanto avviene nel Supervised Learning, l'algoritmo riceve in input, durante la fase di training, dati non etichettati. E' l'algortimo stesso, quindi, che dovrà individuare pattern ricorrenti o clusters all'interno dei dati in modo tale da poter classificare i dati sulla base di essi. Ciò implica che per questo tipo di modelli non si possano valutare le performance in termini, ad esempio, di accuracy come invece avveniva nel Supervised Learning.

I principali campi in cui si suddivide l'Unsupervised Learning sono: Clustering, analisi delle Regole di Associazione ed, infine, Dimensionality Reduction. Di seguito, verrà fornita una breve introduzione alle tecniche di Riduzione della dimensionalità e alle Regole di Associazione mentre seguirà una più dettagliata disamina degli algoritmi di Clustering in un successivo paragrafo dato l'utilizzo di tali tecniche all'interno del progetto di ricerca in esame.

Le tecniche di Dimensionality Reduction risultano fondamentali nei casi di dati con un numero elevato di attributi che risulterebbero altrimenti computazionalmente onerosi da analizzare. In particolare, tali modelli mirano ad individuare quel sottoinsieme di features che determina la maggiore variabilità all'interno dei dataset, secondo una solgia percentuale stabilita dall'analista. In altre parole, vengono utilizzati per rendere esplicite le informazioni maggiormente significative ai fini dell'analisi eliminando "rumore" presente all'interno dei dati che potrebbe generare errori o bias nei risultati delle analisi. Tali algoritmi, dunque, trovano il loro naturale utilizzo nell'ambito del preprocessing dei dati e molto spesso vengono utilizzate per trattare il dataset da mandare in input ad altri algoritmi di Machine Learning, che spesso possono essere algoritmi di Supervised Learning o Reti Neurali.

Le Regole di Associazione, o Association Rules Learning, raggruppano una serie di algoritmi di Machine Learning definiti rule-based. Essi, infatti, vengono utilizzati per esplicitare relationi tra variabili o elementi all'interno di vasti dataset contenti oggetti sotto forma di "transazioni", dove per transazione si intende un insieme di oggetti non ordinati. L'obiettivo di tali strumenti di analisi risulta si sostanzia, dunque, nell'individuazione di correlazioni o pattern frequenti all'interno di un database

transazionale. Da qui si evince la natura esplorativa delle Association Rules, le quali molto spesso, quindi, vengono utilizzate anche per estrapolare la conoscenza all'interno di clusters. Le Regole di Associazione trovano largo utilizzo nei campi della Bioinformatica, dell'intrusion detection e soprattutto della Market Basket Analysis operata negli anni sia dai grandi players della grande distribuzione (GDO) che dagli e-commerce online (Amazon) per conoscere le abitudini di consumo dei rispettivi clienti.

2.6 Reinforcement Learning

Il terzo dei tre paradigmi di base del Machine Learning è il Reinforcement Learning, o Apprendimento con Rinforzo. Esso studia come un agente intelligente possa trasformare gli stimoli ambientali in azioni massimizzando un segnale numerico di reward o ricompensa. Per questo motivo il Reinforcement Learning può essere definito un modello di apprendimento comportamentale [3]. In questo caso, l'algoritmo riceve dei feedbacks sotto forma di segnali di rewards dall'analisi dei dati in modo tale da guidare l'agente nel compiere le azioni necessarie per portare a termine un determinato compito. Sebbene anche per il Reinforcement Learning si parli di una fase di addestramento (training phase), essa assume un significato completamente diverso rispetto ai casi del Supervised e dell'Unsupervised Learning. In questo caso, infatti, l'algoritmo non impara su una specifico campione di dati di training ma tramite un processo di "trial and error". Così facendo il processo e la seguenta di azioni viengono "rinforzate" da rewards positivi in caso di scelta di azioni che risolvono in maniera soddisfacente il problema. Un'altra caratteristica fondamentale del Reinforcement Learning riesiede nel fatto che la massimizzazione della funzione di reward deve tener conto del trade-off che nasce dalle necessità da parte dell'algoritmo di "sfruttare" (*exploitation*) le scelte già compiute delle quali si conosce già il valore associato della funzione di ricompenza oppure di "esplorare" (*exploration*) nuove azioni dal payoff sconosciuto [6]. Da qui scaturisce, dunque, il processo di trial and error alla base del paradigma. L'agente, infatti, tenendo conto del trade-

off tra exploitation ed exploration compierà una serie di scelte progressive che lo condurranno inevitabilmente a fallire in alcuni istanti.

Le maggiori applicazioni dei modelli di Apprendimento con Rinforzo si hanno nella Robotica e soprattutto nel campo dei veicoli a guida autonoma. In questo campo di applicazione ad esempio, risulta evidente come l'interazione con l'ambiente sia fondamentale. Al fine di sviluppare il modello infatti occorrerà prima cercare di testare il modello in un ambiente controllato in cui tutti gli agenti in gioco siano a guida autonoma per poi pensare di immettere il veicolo su strada. E' evidente che in un ambiete controllato il processo di trial and error sia più facile da gestire in termini di costi dovuti a scelte sbagliate (incidenti).

Infine, un'ultima famosissima applicazione dei modelli di Reinforcement Learning è rappresentata da AlphaZero, un algoritmo sviluppato da DeepMind (azienda acquisita nel 2014 da Google) in grado di giocare e vincere contro campioni umani nei giochi cinese di Go, Shoji e nel tradizionale gioco degli scacchi.

2.7 Ensemble Learning

L'Ensemble Learning è un paradigma del Machine Learning nel quale vengono utilizzati modelli multipli al fine di migliorare le performance nella risoluzione di un determinato task. Le sue principali applicazioni sono nell'ambito della Classificazione e per questo molto spesso l'Ensemble Learning è ritenuto un evoluzione delle tecniche di Supervised Learning. Lo scopo fondamentale di tali tecniche è quello di migliorare le performance di un modello predittivo riducendo, quindi, la probabilità di errore o più in generale gli errori relativi a bias e varianza. Gli algoritmi di classificazione utilizzati per i modelli di Ensemble Learning possono essere MLP (Multi Layer Perceptron), Support Vector Machine o ancora algoritmi meno complessi come Decision Trees e Random Forest. Ciascuno di questi modelli, prendendo in input i dati di training, elabora una predizione con una differente probabilità di errore legata al modello. Pur fornendo previsioni diverse è molto probabile che vi siano, poi, previsioni uguali.

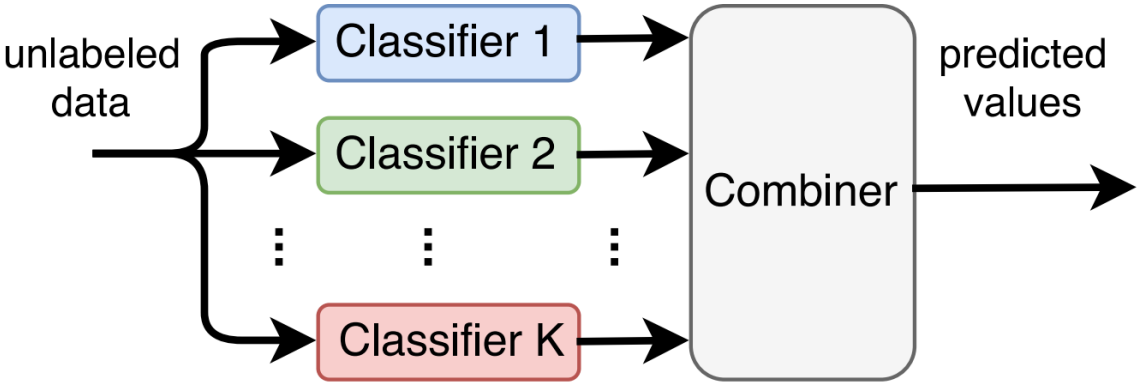

Figura 2.3: Esempio di processo di Ensemble Learning

A questo punto, dunque, il modello sceglie la soluzione più frequente o comunque quella fornita dalla maggioranza dei classificatori mediando quindi le predizioni e riducendo l'errore del processo decisionale.

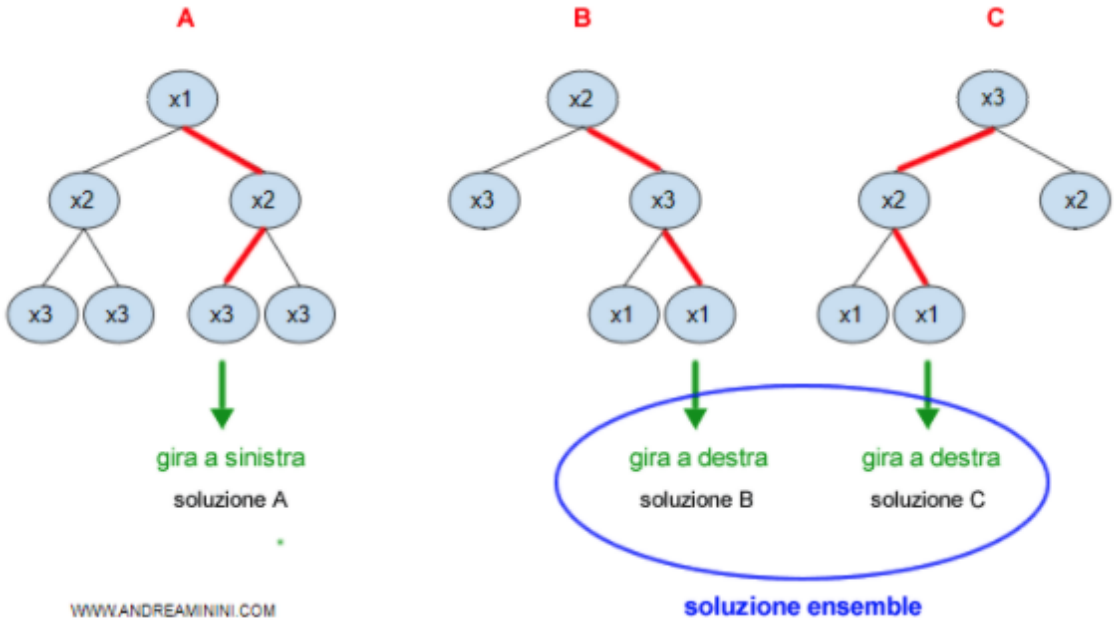

Figura 2.4: Esempio di processo decisionale di Ensemble Learning nel caso di tre Decision Trees

Una possibile criticità che può nascere nei processi di Ensemble Learning è dovuta ad eventuali bias o dati fuorvianti all'interno del dataset di training utilizzato per

tutti i modelli di Classificazione in parallelo. In questo caso, infatti, ciascun modello potrebbe produrre previsioni fuorvianti con conseguente ripercussione sul risultato finale del processo. Un metodo per arginare tale rischio è quello di dare in input a ciascun modello di classificazione un diverso dataset di training.

Gli algoritmi di Ensemble Learning vengono suddivisi in tre diverse categorie: *Bagging*, *Boosting* e *Stacking*. Nel *Bagging*, che sta per "bootstrap aggregating", vengono utilizzati diversi classificatori dello stesso tipo (in figura 2.4 viene fornito un esempio con tre Decision Tree) ai quali vengono dati in input diversi campioni di dati ottenuti a partire dal medesimo dataset di training [7]. La predizione finale viene, infine, elaborata scegliendo la soluzione proposta dalla maggioranza dei classificatori.

Il *Boosting* riprende gran parte delle strategie di analisi del *Bagging* fatta eccezione per la scelta finale prodotta dal modello Ensemble. In questo caso, infatti, il modello restituisce una media pesata delle decisioni finali prese da ciascun classificatore dove i pesi sono rappresentati dall'accuratezza della previsione elaborata da ciascun algoritmo di classificazione.

Lo *Stacking*, infine, rappresenta la forma più evoluta di Ensemble Learning. In questo caso, a differenza del *Bagging* e del *Boosting*, vengono utilizzati modelli eterogenei di classificatori. Inoltre la predizione finale dell'intero modello di Stacking non viene più elaborata sulla base di una media bensì viene restituita da un meta-classificatore (in genere una rete neurale) che prende in input le previsioni di ciascun classificatore.

2.8 Deep Learning

Con l'espressione Deep Learning ci si riferisce a quell'insieme di metodi di Machine Learning che si servono di modelli computazionali multi-layer per apprendere la rappresentazione di dati secondo livelli di astrazione multipli [8]. I modelli di Deep Learning sfruttano, in particolare, una classe di algoritmi nota come Reti Neurali Artificiali; si tratta di strutture a più livelli che cercano di apprendere dai dati in maniera iterativa senza l'intervento umano. Per questo motivo, tali tecniche

risultano particolarmente utili nell'individuazione di pattern all'interno di dati non strutturati.

Mentre i tradizionali algoritmi di Machine Learning richiedono operazioni preliminari di feature selection per permettere ai modelli di estrapolare conoscenza utile dal dataset, i modelli di Deep Learning estrapolano automaticamente la rappresentazione dei dati necessaria ai fini dell'esplicitazione di patterns o della classificazione. In tal senso, questa classe di algortimi basati su Reti Neurali, trasformano il dato di input ad ogni livello raggiungendo livelli d'astrazione sempre più alti rendendo possibile apprendere anche funzioni molto complesse dai dati. Per task di classificazione, ad esempio, maggiori livelli o layers, permettono di amplificare aspetti dei dati di input che possono determinare una maggiore accuratezza della previsione.

Da qui deriva il nome "Deep Learning" o Apprendimento Profondo che rimanda ai livelli multipli di una rete neurale che permettono di estrapolare sempre più conoscenza dal dato grezzo. L'aspetto chiave di tali metodologie, dunque, è che tali livelli di feauture non vengono selezionati da umani ma vengono apprese dall'algoritmo in fase di analisi.

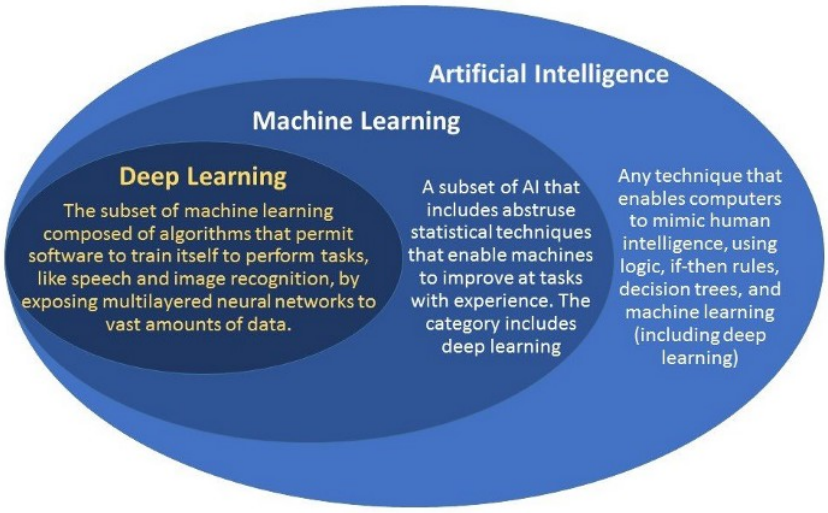

Figura 2.5: Relazione tra AI, Machine Learning e Deep Learning

L'obiettivo degli algoritmi di Deep Learning è replicare il funzionamento del cervello umano in cui ogni nodo di una Rete Neurale riproduce un neurone che funge

da tramite per l'informazione. Proprio per questo motivo, tali architetture, forniscono le migliori soluzioni a problemi in tutti quei campi in cui sarebbero richieste capacità umane come il riconoscimento di immagini, della voce, Natural Language Processing o ancora Bioinformatica e cosi via ottenendo risultati straordinariamente promettenti.

Grazie al continuo aumento dei dati disponibili e alle nuove soluzioni che permettono l'accesso a macchine dalla grandi capacità computazionali (come ad esempio le GPU mobile o le architetture Cloud) il Deep Learning sta riscuotendo sempre più successo sia in ambito scientifico che in ambito business; tanto che lo si ritrova già alla base di numerosi applicativi entrati a far parte della vita di tutti i giorni come, ad esempio, filtri spam delle mail, traduttori automatici e assitenti vocali ecc...

2.9 Reti Neurali

Con l'espressione Reti Neurali vengono indicati quei modelli computazionali che cercano di riprodurre il funzionamento del cervello umano. Essi rappresentano il fondamento del Deep Learning; il termine "Deep" (profondo) infatti, fa riferminento ai diversi livelli di nodi o neuroni che caratterizzano una Rete Neurale. Una Rete Neurale Artificiale o, più semplicemente Rete Neurale, è basata su una collezione di nodi definiti, per l'appunto neuroni, che, in maniera analoga a quanto accade con le sinapsi per il cervello umano, sono collegati tra loro mediante delle interconnesisoni.

Ciascun neurone riceve in ingresso l'informazione dal suo precedente, la processa e trasmette il dato di output, calcolato mediante funzioni non lineari, al suo successivo. Inoltre, ogni neurone e le sue interconnessioni hanno dei pesi che vengono modificati ad ogni interazione mano a mano che il processo di apprendimento si affina adattandosi ai dati di input. Tali pesi vengono chiamati "attivazione" per ciascun neurone.

Come accennato in precedenza, i neuroni di una Rete sono raggruppati in livelli o layers. La struttura base di una rete neurale prevede almeno tre diversi tipi di livelli: il layer di input, attraverso i quali vengono forniti in ingresso i dati di

training all'algoritmo, i layers intermedi o nascosti (dall'Inglese "hidden layers") che elaborano l'informazione e la trasmettono, poi, ai layers di output che restituiscono il risultato, tipicamente una previsione.

E' evidente, dunque, come tali modelli rappresentino dei potentissimi strumenti per la risoluzione di problemi legati ai più disparati ambiti. Tuttavia occorre sottolineare anche alcune criticità che emergono nell'utilizzo delle Reti Neurali. In primis, al fine di ottenere modelli accurati, è necessario addestrare le Reti su una quantità molto significativa di dati e, inoltre, tali algoritmi agiscono come delle "Black Boxes" per cui è impossibile risalire ai singoli step del processo di analisi effettuato dall'algoritmo. Ciò che si può osservare infatti, utilizzando una rete neurale, sono solamente i dati in ingresso e la previsione più o meno accurata in output, senza avere alcuna informazione sui fattori che determinano tale output. Tale criticità non è da sottovalutare, in quanto limita l'utilizzo di questi modelli in tutti quei campi nei quali è necessario fornire le motivazioni sottostanti un processo decisionale come ad esempio in campo medico o nella concessione del credito.

Principali Architetture di Reti Neurali

Come visto, una Rete Neurale si costituisce di neuroni, organizzati in layers, e interconnessioni. Ora, vi sono diversi modi per connettere tra loro i neuroni artificiali di una rete che determinano differenti architetture di Reti Neurali, ciascuna delle quali viene utilizzata per risolvere differenti classi di problemi.

Le principali architetture note in letteratura sono: le Reti Feedforward, le Reti Neurali Convoluzionali o Convolutional Neural Networks (CNN), le Reti Neurali Ricorrenti (Recurrent Neural Networks-RNN) e le Generative Adversarial Networks (GAN). In figura 2.6 viene fornita una panoramica esaustiva delle diverse architetture di Reti Neurali.

Nel seguente elaborato, sfruttando un famoso caso studio presente in letteratura, verrà approfondito il funzionamento delle Reti Feedforward, in particolare del Multilayer Perceptron. Di seguito viene riportato un piccolo excursus sul funzionamento dei modelli RNN, CNN e GAN.

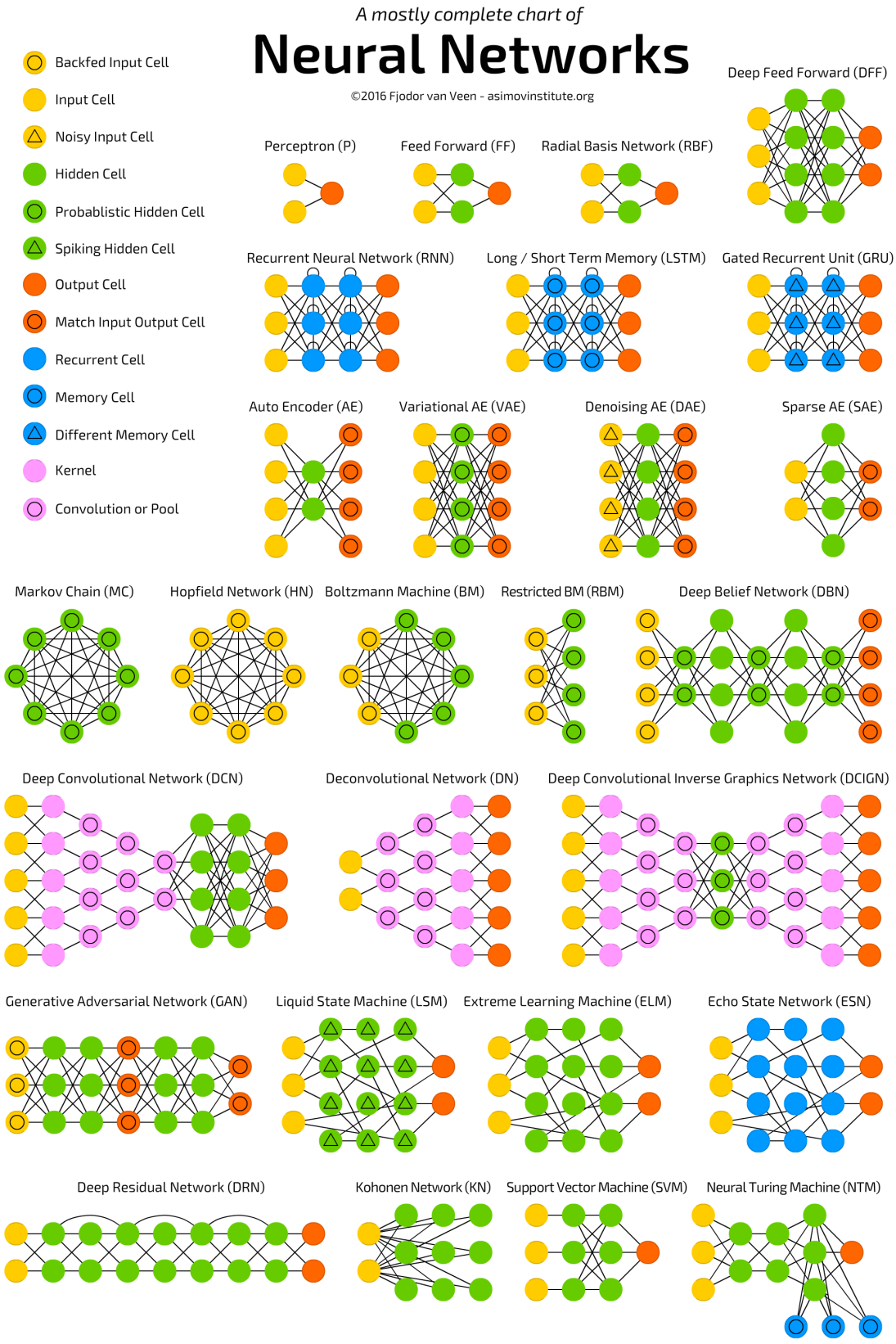

Figura 2.6: Principali architetture di Reti Neurali

Le Reti Neurali Ricorrenti (RNN) sono una classe di Reti Neurali in cui le interconnessioi tra i nodi formano un grafo ciclico. Ciò significa che l'output di un neurone viene utilizzato come input per un neurone di un layer precedente. Ciò permette, quindi, alle RNN di conservare memoria delle informazioni temporali e delle relazioni emerse nei dati sfruttando un layer di memoria di stato potendo così modellare comportamenti temporali dinamici[1]

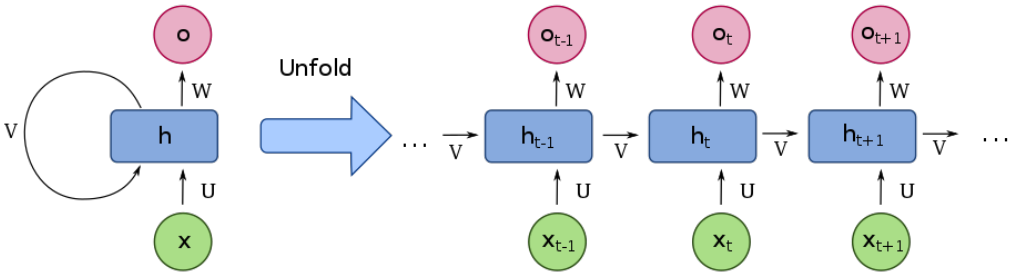

Figura 2.7: Esempio di RNN: a sinistra rappresentezaione compatta degli hidden layers, a destra visualizzazione estesa.

In virtù della loro capacità di conservare memoria delle correlazioni tra un input e gli altri e di modellare comportamenti temporali, le RNN sono utilizzate in ambiti in cui è fondamentale l'individuazione di correlazioni tra dati. Questi sono il riconoscimento vocale, l'analisi predittiva su sequenze di dati, il riconoscimento della grafia o ancora l'analisi di serie storiche.

Le CNN, Convolutional Neural Networks, sono una classe di di Reti Neurali caratterizzate da due particolari tipi di hidden layers: i convolutional layers e i pooling layers. I primi sono in grado di astrarre i dati di input restituendo la cosi detta "activation map" mentre i secondi riducono la dimensionalità dell'activation map tramite operazioni di pooling[2]. In Fig. 2.8 viene mostrato un esempio di CNN[3].

[1] https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-2

[2] https://en.wikipedia.org/wiki/Convolutional_neural_network

[3] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks

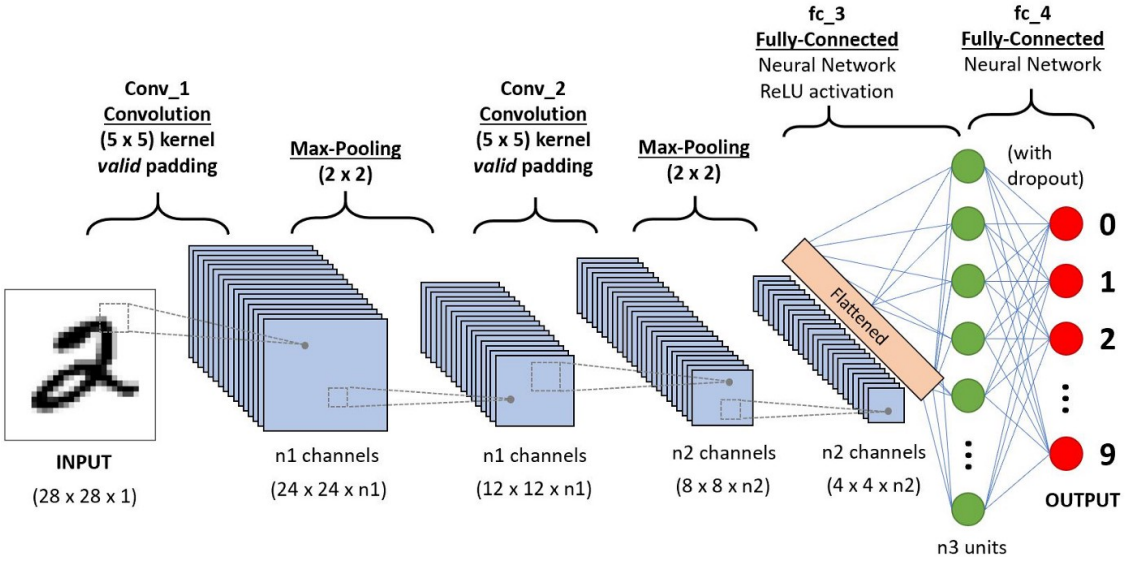

Figura 2.8: CNN utilizzata nella classificazione di numeri scritti a mano

Le GAN, Generative Adversarial Networks, rappresentano una delle classi di Reti Neurali più all'avanguardia. In tali modelli due differenti Reti Neurali vengono addestrate in maniera competitiva nella forma di un gioco a somma zero, ossia in un contesto in cui le perdite di un agente rappresentano i profitti dell'altro[4]. Dato un dataset di training, le GAN sono in grado di riprodurre con fedeltà le distribuzioni dei dati, sia che il dato in input sia una fotografia o che sia un campione musicale. Tali modelli, per questo motivo, vengono utilizzati per rigenerare vecchie foto a bassa risoluzione, per migliorare la resa di immagini astronomiche o ancora per scalare la risoluzione di videogiochi ecc.

Multilayer Perceptron e funzionamento di una Rete Neurale

Il funzionamento di una Rete Neurale, come visto, è strettamente correlato con la struttura delle interconnessioni tra i neuroni della stessa rete. Al fine di comprendere il funzionamento di base di questa classe di algoritmi, verrà utilizzato come riferimento il Multilayer Perceptron (anche chiamato in gergo "Plain Vanilla"), uno dei primi modelli elaborati che di conseguenza risulta ampiemente studiato all'in-

[4] `https://en.wikipedia.org/wiki/Generative_adversarial_network`

terno della comunità scientifica. Inoltre, per facilitare la comprensione del lettore, si procederà alla spiegazione sfruttando un esmpio noto in letteratura di classificazione di cifre numeriche scritte a mano raccolte all'interno del dataset MNIST.

Il Multilayer Perceptron è un particolare tipo di *deep feedforward neural network*, ossia una rete neurale caratterizzata da un numuro elevato di hidden layers in cui le interconnessioni tra nodi non formano cicli. Tale Rete Neurale è utilizzata soprattutto nell'ambito del Supervised Learning per cercare soluzioni a problemi di Classificazione.

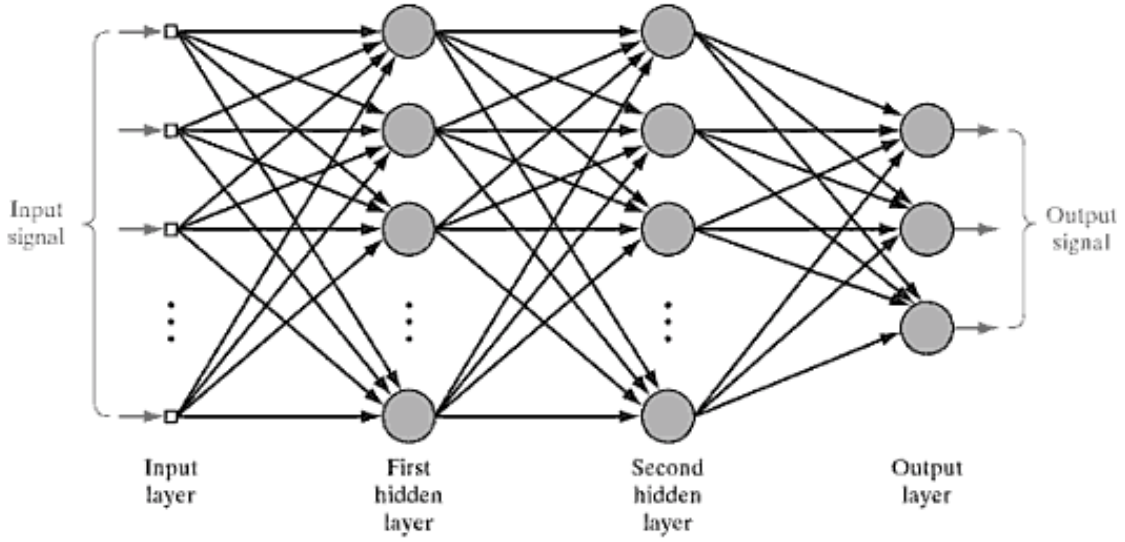

Figura 2.9: Esempio di MLP

In figura 2.9 viene fornito un esempio generico di architettura MLP caratterizzato da due hidden layers. Dall'immagine si evince la struttura di base di tale modello: il primo livello è definito *input layer* e conta un numero di nodi pari al numero di features presenti all'interno dei dati forniti in ingresso all'algoritmo. Nel caso, ad esempio, dello studio sulla classificazione di immagini cinematografiche a monte di tale progetto di ricerca [9] i dati di training sono rappresentati da immagini in formato 16:9, ossia aventi una risoluzione di 160x90 pixel e di conseguenza l'input layer implementato si caratterizza per un numero di neuroni pari a 14'400. Nel caso, invece, del riconoscimento di cifre numeriche scritte a mano il dato in ingresso consisteva in immagini 28x28 pixel per un totale di 784 neuroni nell'input layer.

Proseguendo verso destra in figura 2.9, troviamo i vari *hidden layers* che, come accennato in precedenza, rappresentano il punto centrale del modello, ossia quello in cui i dati vengono elaborati e in cui vengono poi effettuate le previsioni. Non esiste un numero predefinito di hidden layers o un numero di neuroni da utilizzare per un dato problema bensì occorre testare differenti configurazioni di nodi e livelli al fine di individuare la soluzione che garantisce le migliori performance. Infine, troviamo l'*output layer* che è caratterizzato da un numero di neuroni pari al numero di classi complessive del task di riferimento. Ad esempio, se si considera il task di classificazione di cifre numeriche scritte a mano, troveremo 10 diversi nodi di output; uno per ciascuna cifra numerica da 0 a 9.

Ora, a ciascun nodo viene attribuito un coefficiente che prende il nome di "activation" e assume valori compresi nell'intervallo [0,1]. Tale valore viene influenzato inevitabilmente dal valore di attivazione di degli altri nodi; infatti, una volta che l'informazione arriva ad un neurone esso si attiva e trasmette l'informazione a tutti i suoi successivi "attivandoli", un pò come avviene all'interno del cervello umano. Ciò implica che ciascun neurone di un livello è correlato ai neuroni degli altri livelli. Tale relazione viene espressa, nel caso di un MLP "fully connected"[5], mediante un peso scalato tra 0 e 1 assegnato a ciascuno degli archi che connettono in i nodi della rete. Tale relazione potrà essere positiva, se il neurone sul layer successivo dovrà essere attivo, o negativa nel caso opposto. Ciò porta, quindi, l'algortimo ad elaborare i valori di attivazione dei nodi dell'*output layer* fornendo la classe predetta in base al valore maggiore tra gli stessi. Considerando il caso presentato in figura, 2.9, di un MLP con due starti intermedi, il valore di attivazione di un neurone su un hidden layer è dato da:

$$a = (w_1 a_1 + w_2 a_2 + ... + w_n a_n) \tag{2.1}$$

dove $(w_1, w_2, ..., w_n)$ rappresentano i pesi associati agli archi della rete. Il risultato di somma pesata di questo tipo potrebbe essere qualsiasi numero, tuttavia, ai

[5]Per fully connected MLP si intende un Multilayer Perceptron in cui ogni neurone di un layer è interconnesso a tutti i neuroni del layer successivo

fini di contenere la complessità computazionale di tale processo occorre riscalare tali valori all'interno dell'intervallo [0,1]. Per fa ciò ci si serve di particolari funzioni matematiche, come ad esempio la funzione logistica (anche detta Sigmoid function) oppure la funzione "ReLu". Tali funzioni (Fig 2.10) prendono il nome di *Funzioni di attivazione*

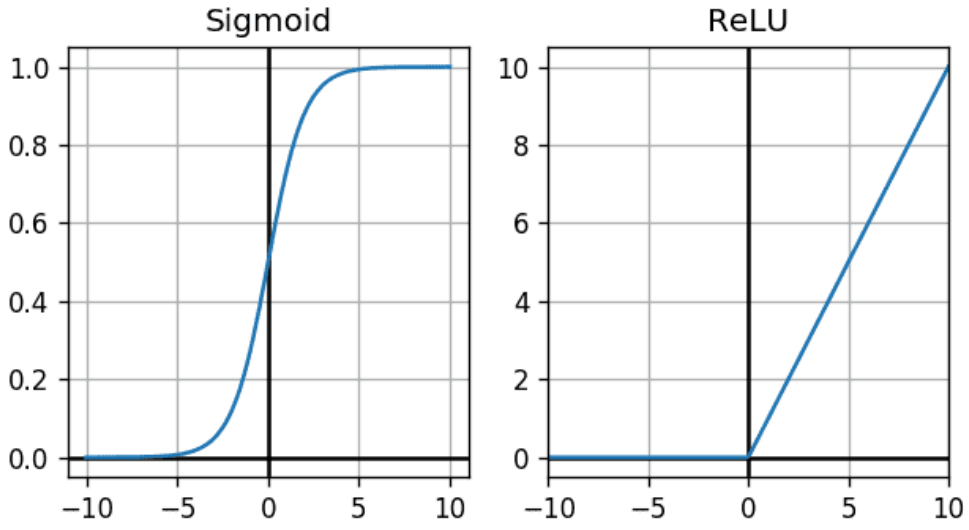

Figura 2.10: Grafico delle funzioni Sigmoid e ReLu

Prendiamo il caso della funzione Sigmoid:

$$\sigma(x) = \frac{1}{(1 + e^{-}x)} \tag{2.2}$$

L'equazione per ottenere valori di attivazione tra 0 e 1 diventa:

$$a = \sigma(w_1 a_1 + w_2 a_2 + ... + w_n a_n) \tag{2.3}$$

In alcuni casi, potrebbe risultare necessario far si che il neurone non si attivi a meno che una certa soglia non venga superata. Per modellare ciò occorre, quindi, inserire un coefficiente di distorsione anche detto *bias*. Con esso l'equazione diventa:

$$a = \sigma(w_1 a_1 + w_2 a_2 + ... + w_n a_n + b) \tag{2.4}$$

Siamo arrivati, dunque, ad introdurre la formula per calcolare l'attivazione di un singolo neurone. Tale formula può essere ulteriormente generalizzata introducendo i riferimenti ai layers e ai singoli neuroni di esso come segue:

$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + ... + w_{0,n}a_n^{(0)} + b_0) \tag{2.5}$$

Nella (2.5), gli apici dei valori di attivazione indicano il layer o livello mentre i pedici fanno riferimento alla posizione del neurone nel singolo layer (in questo caso si è mostrato l'esempio per il calcolo dei valori di attivazione nel layer 1). Ciascun peso $w_{i,j}$, indica la correlazione tra l'i-esimo nodo del layer considerato e il j-esimo nodo del layer precedente al quale è collegato. Sfruttando il calcolo matriciale e il prodotto righe per colonne possiamo, infine, generalizzare l'espressione come segue:

$$a^1 = \sigma(Wa^0 + b) \tag{2.6}$$

dove W rappresenta la matrice kxn dei pesi, mentre a^1, a^0 *e* b rappresentano i vettori dei nodi del livello 1 e 0 e il vettore dei bias.

Apprendimento di una Rete Neurale

Abbiamo visto che una Rete Neurale esegue una classificazione mediante il calcolo iterativo di pesi, attivazioni e bias. Ciò avviene durante la fase di training, ossia quando vengono dati in input all'algoritmo dati già etichettati al fine di addestrarlo a riconoscere le classi. In questa fase la rete rielabora diverse volte il dataset di training; ciascuna di queste iterazioni prende il nome di *epoca*. Durante ogni epoca l'algoritmo ricalcola e aggiusta i valori di attivazione, i pesi e i bias ogni qual volta effettua una classificazione errata; è evidente che può fare ciò in virtù del fatto che i dati di training sono già etichettati. Tale processo aiuta il classificatore ad ottenere performance migliori in termini di accuracy in fase di test, ossia quando riceve dati non etichettati di test.

E' chiaro, dunque, che il funzionamento di una rete neurale è strettamente interconnesso al processo di rivalutazione dei pesi, delle attivazioni e dei bias. Inizialmen-

te essi verranno selezionati in modo randomico e questo porta ad avere accuratezze bassissime nelle epoche iniziali.

Per capire come ricalibrare tali valori l'algoritmo si avvale di una *funzione di costo*. Ogni volta che in fase di training effettua una previsione sbagliata, la Rete calcola il "costo" di questa previsione; in genere per far ciò utilizza funzioni quadratiche come l'SSE (*Sum of Squared Error*). Maggiore sarà il costo calcolato e maggiore sarà l'errore commesso in fase di classificazione; viceversa più il costo è prossimo allo zero e più accurata sarà la previsione. Di conseguenza l'obiettivo, ora, sarà trovare pesi, attivazioni e bias tali da minimizzare la funzione di costo. Per far ciò viene utilizzata una tecnica di ottimizzazione che prende il nome di *Gradient Descent* capace di risolvere problemi di minimizzazione di funzioni multivariate.

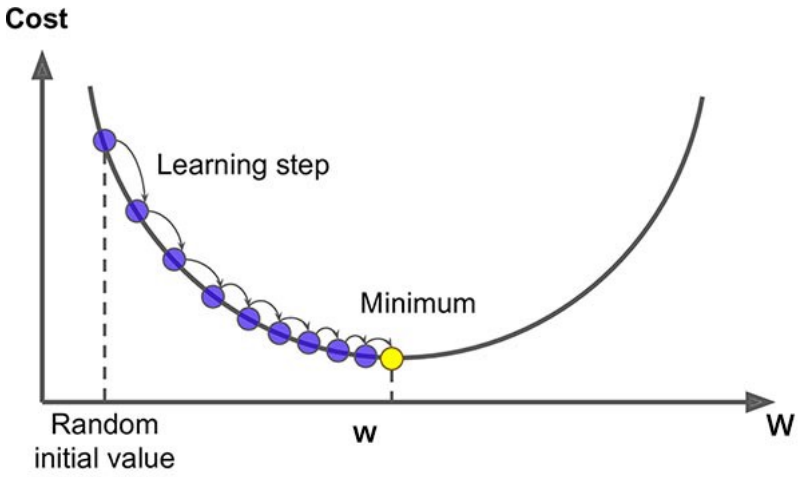

Figura 2.11: Visualizzazione esemplificativa della tecnica del Gradient Descent applicato ad una generica funzione di costo

Il *Gradient Descent* viene utilizzato dall'algoritmo per calcolare il minimo della funzione. Poichè, però, per funzioni molto complesse da milioni di variabili (come quelle che vengono spesso utilizzate per le Reti Neurali) è molto oneroso effettuare il calcolo della derivata della funzione al fine di individuare il minimo globale si procede inizializzando in modo randomico "la posizione" lungo la curva di costo, selezionando in modo casuali gli input, e analizzando quella porzione di grafico calcolando la retta tangente la curva in quel punto (derivata) e muovendosi a destra

della curva se la tangente è negativa[6] o a sinistra se la tangente risulta positiva; e così via fino ad arrivare ad un minimo locale. Ovviamente, nel caso di spazi multidimensionali, non si può più guardare al Gradiente come ad una derivata bensì occorre generalizzarlo in un vettore delle derivate parziali. La tecnica del *Gradient Descent*, dunque, consiste nel processo iteritivo di calcolo del gradiente negativo per individuare una direzione lungo la curva al fine di muoversi verso un minimo della funzione, modificando i pesi e i valori di attivazione dei nodi. Così facendo, tale tecnica definisce l'importanza e la sensitività della funzione di costo a ciascun peso, attivazione e bias. Maggiore è il peso di un arco e più la funzione di costo è sensibile alla variazione dello stesso. Il processo, infine, sarà tanto più rapido quanto più alto è il *learning rate*.

Abbiamo visto come una Rete Neurale individua le variazioni da apportare ai valori di nodi e archi della rete; ossia mediante la tecniche del *Gradient Descent*. Tuttavia, per poter comprendere nel profondo il processo di apprendimento di una Rete Neurale occorre introdurre il concetto di *Backpropagation*.

La *Backpropagation* è l'algoritmo alla base del calcolo del gradiente negativo e di consegueza, del processo di aggiustamento dei pesi, delle attivazioni e dei bias. Dal punto di vista concettuale, la backpropagation fa sì che ogni qual volta la Rete effettua una previsione sbagliata analizza i pesi di tutti gli archi che connettono l'ultimo hidden layer con l'output layer. Procedendo poi a ritroso, tiene traccia di tutte le attivazioni dei nodi appartenenti ai livelli precedenti fino ad arrivare all'input layer. Qui, in base al costo della previsione sbagliata e alla sensibilità della funzione di costo ai vari archi, modifica i pesi degli archi che collegano l'input layer al primo hidden layer facendo variare a cascata le attivazioni dei neuroni dei layer successivi e cosi via fino a ritornare all'output layer. Per quanto riguarda i bias, invece, al fine di ottenere un attivazione più elevata possibile nel neurone dell'output layer corrispondente alla classe da predire l'algoritmo tenderà ad aumentare il bias del nodo che codifica per la classe corretta e diminuire il bias degli altri nodi dell'input

[6]Il gradiente di una funzione misura la crescita di una curva; di conseguenza la tecnica del *Gradient Descent* cerca di calcolare un gradiente negativo

layer in modo da rendere più difficile l'attivazione degli stessi (dalla (2.5)).

Proviamo, ora, a formalizzare matematicamente l'intuizione che si nasconde dietro l'algoritmo di *Backpropagation*. Consideriamo ora due neuroni posti rispettivamente sui layer L ed $L-1$.

Dall'eq (2.5) abbiamo che:

$$a^L = \sigma(w^L a^{L-1} + b^L) \tag{2.7}$$

Sia ora y l'output desiderato. Il costo sarà dunque:

$$C_0 = (a^L - y)^2 \tag{2.8}$$

Definiamo ora, la somma pesata delle attivazioni di un layer come z^L. Otterremo dalla eq. (2.5):

$$\begin{aligned} z^L &= w^L a^{L-1} + b^L \\ a^L &= \sigma(z^L) \end{aligned} \tag{2.9}$$

L'obiettivo, dunque, è quello di valutare la sensitività della funzione di costo alle variazioni dei pesi. Per far ciò, calcoliamo la derivata parziale rispetto al peso w^L e dalle eq. (2.8) e (2.9), sfruttando la chain rule delle derivate parziali, avremo[7]:

[7] da `https://www.3blue1brown.com/lessons/backpropagation-calculus`

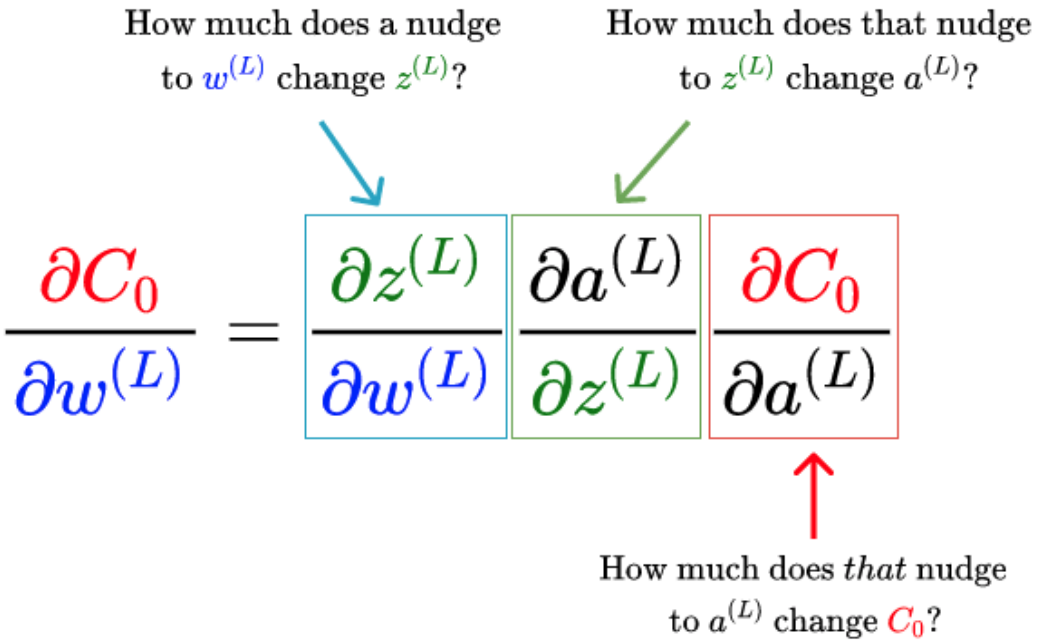

Figura 2.12: Effetto della variane del peso w^L sulla funzione di costo C_0

Dalle analisi precedenti sappiamo:

$$\frac{\partial z^L}{\partial w^L} = a^{L-1}$$

$$\frac{\partial a^L}{\partial z^L} = \sigma^{'}(z^L) \tag{2.10}$$

$$\frac{\partial C_0}{\partial a^L} = 2(a^L - y)$$

Queste tre equazioni, ci dicono che una variazione del peso w^L ha un effetto maggiore su z^L e di conseguenza sul costo quando il neurone precedente è più attivo. Inoltre, dalla $\frac{\partial C_0}{\partial a^L}$ si evince che il costo è proporzionale alla differenza tra l'output desiderato e quello ottenuto. Ciò significa che quando la previsione è di molto differente da quella desiderata anche piccole variazioni del valore di attivazione del neurone possono aumentare significativamente il costo. Mettendo tutto insieme, l'espressione in Fig 2.12 diventa:

$$\frac{\partial C_0}{\partial w^L} = a^{L-1}\sigma'(z^L)2(a^L - y) \tag{2.11}$$

Tale formula ci dice come la variazione di un singolo peso nell'ultimo layer influenzi la funzione di costo per uno specifico record di training. Al fine di ottenere il vettore del Gradiente negativo occorrerà generalizzare tali espressioni per tutti i dati di training.

Procediamo calcolando la funzione di costo totale come la media di tutti i singoli costi associati a ciascun dato di training:

$$C = \frac{1}{n}\sum_{k=0}^{n-1} C_k \tag{2.12}$$

La derivata di tale funzione ci dice come cambia l'intera funzione di costo al variare del peso in esame.

In maniera analoga a quanto fatto in precedenza, ora, possiamo valutare il gradiente rispetto ai bias e la sensitività della funzione di costo all'attivazione del neurone sul layer precedente:

$$\begin{aligned} \frac{\partial C_0}{\partial b^L} &= \sigma'(z^L)2(a^L - y) \\ \\ \frac{\partial C_0}{\partial a^{L-1}} &= w^L\sigma'(z^L)2(a^L - y) \end{aligned} \tag{2.13}$$

E' bene sottolineare come questi siano le intuizioni matematiche dell'algoritmo di *Backpropagation* nel caso di una rete neurale composta esclusivente da due nodi e un arco al quale è associato un solo peso.

Per reti neurali decisamente più complesse, in realtà, non cambia molto. A livello di notazione ciò che va introdotto in questo caso è un indice al pedice di ogni espressione per far riferimento ad uno specifico neurone di un determinato layer, che invece sarà ancora indicato ad apice.

Sia j il j-esimo neurone del layer L e k il k-esimo neurone del layer L-1. La funzione di costo per un singolo dato di training sarà:

$$C_0 = \sum_{j=0}^{n-1} (a_j^L - y_j)^2 \tag{2.14}$$

La somma pesata di pesi, attivazioni e bias diventerà:

$$z_j^L = w_{j,0}^L a_0^{L-1} + w_{j,1}^L a_1^{L-1} + w_{j,2}^L a_2^{L-1} + ... + b_j^L \tag{2.15}$$

La nuova equazione per il calcolo del gradiente, dunque, analogalmente a quanto avveniva per la semplice rete neurale con un neurone per layer; potrà essere derivata dalla *chain rule* del calcolo differenziale:

$$\frac{\partial C_0}{\partial w_{jk}^L} = (\frac{\partial z_j^L}{\partial w_{jk}^L})(\frac{\partial a_j^L}{\partial z_j^L})(\frac{\partial C_0}{\partial a_j^L}) = a_k^{L-1}\sigma'(z^L)(\frac{\partial C}{\partial a_j^{L-1}}) \tag{2.16}$$

Ciò che cambia in questo caso è la derivata della funzione di costo rispetto alle attivazioni nel layer L-1. Esse infatti influenzano in diversi modi la funzione di costo. Matematicamente possiamo esprimere tale dipendenza come somma di diversi contributi:

$$\frac{\partial C_0}{\partial a_j^{L-1}} = \sum_{j=0}^{n_L-1} (\frac{\partial z_j^L}{\partial a_k^{L-1}})(\frac{\partial a_j^L}{\partial z_j^L})(\frac{\partial C_0}{\partial a_j^L}) \tag{2.17}$$

Tale somma si riferisce a tutto il layer L. Ripetendo tale operazione per tutti i pesi e i bias di ciascun layer otteniamo:

$$\nabla C = \begin{cases} \frac{\partial C_0}{\partial w_{jk}^l} & = a_k^{l-1}\sigma'(z_j^l)(\frac{\partial C}{\partial a_j^l}) \\ & \\ \frac{\partial C}{\partial a_j^l} & = \sum_{j=0}^{n_l-1} w_{jk}^{l+1}\sigma'(z_j^{l+1})(\frac{\partial C}{\partial a_j^{l+1}}) \end{cases} \tag{2.18}$$

Questa è l'espressione del Gradiente calcolato mediante l'algoritmo di Backpropagation che permette di aggiustare pesi, attivazioni e bias al fine di minimizzare la funzione di costo.

Capitolo 3

Metodi di Clustering

Un'altra importante classe di metodi di Unsupervised Learning è rappresentata dagli algoritmi di Clustering. Essi vengono utilizzati per analizzare ed esplorare un set di dati al fine di suddividerlo in gruppi di oggetti tali per cui oggetti dello stesso gruppo (definito *cluster*) siano più simili tra loro rispetto ad oggetti di un altro cluster. In altri termini, punti o oggetti vengono inseriti all'interno dello stesso cluster poichè hanno una distanza minore tra loro; dove la distanza, o in termini opposti la similarità, viene definita sulla base di specifiche funzioni matematiche. la Cluster analysis, di conseguenza, può essere assimilatà a problemi di ottimizzazione multivariata in cui si cerca di minimizzare la distanza tra punti dello stesso cluster e massimizzare quella tra punti di clusters diversi.

In generale, gli algoritmi di clustering si possono suddividere in due macro-aree: *Hierarchical Clustering* e *Partitional Clustering*. Inoltre si possono ulteriormente classificare in algoritmi *model-based*, *density-based* e *distance-based*.

Gli algoritmi di Clustering Partizionale , o Partitional Clustering, operano una suddivisione dei dati in partizioni, ossia dei sotto insiemi o clusters che non si sovrappongono tra loro. Ciò significa che un oggetto è collocato esattamente in uno e un solo cluster. Essi sono costruiti in modo tale da minimizzare la distanza degli oggetti di uno stesso cluster (distanza *intra-cluster*) e massimizzare quella tra clusters (distanza *inter-cluster*).

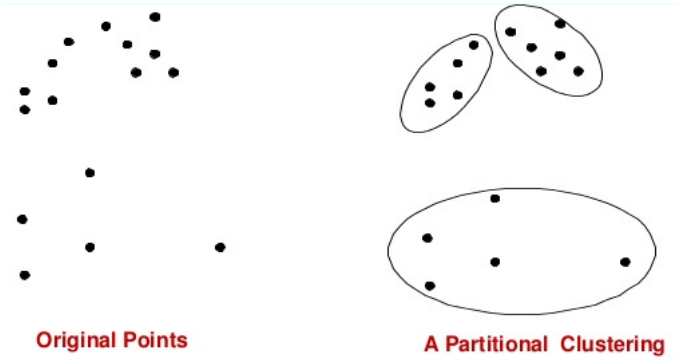

Figura 3.1: Esempio di algoritmo di Partitional Clustering

Sebbene siano numerosi gli algoritmi di Clustering facenti parte della famiglia del Partitional Clustering, in questo elaborato verranno approfonditi i due principali, ossia:

- **K-means**: Esso rappresenta un algoritmo *center-based* di Partitional clustering che mira ad individuare clusters all'interno dei dati sulla base di un parametro K specificato in input
- **DBSCAN**: Consiste in un algoritmo *density-based* che opera un clustering partizionale in cui il numero di clusters viene automaticamente generato dall'algoritmo.

La seconda categoria di modelli di clustering è rappresentata dagli algoritmi di *Hierarchical Clustering*. Questa classe di algoritmi produce un insieme di clusters nidificati e organizzati in un albero gerarchico che prende il nome di *Dendogramma*. In particolare, ci sono due differenti approcci per generare un clustering di tipo gerarchico:

- **Agglomerative**: L'algoritmo inizia considerando ciascun punto come un cluster individuale e ad ogni iterazione unisce i cluster più vicini fino ad ottenere alla fine un unico grande cluster (o k se specificato in input il numero di clusters ricercato).

- **Divisive**: al contrario del precedente approccio, inizializza un unico grande cluster e ad ogni iterazione divide, per l'appunto, i clusters fino a che non se ne ottengono tanti quanti sono i punti del dataset (oppure k clusters indicati)

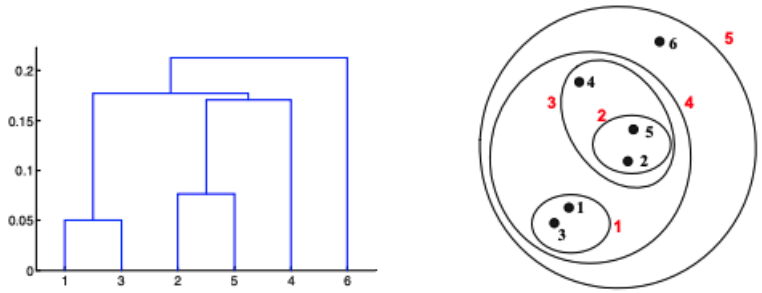

Figura 3.2: Esempio di Hierarchical Clustering e Dendogramma corrispondente

In questo elaborato verrà approfondito il funzionamento dei modelli di Agglomerative clustering in quanto essi risultano essere i più comunemente utilizzati.

3.1 K-means

Il K-means è uno dei metodi più usati nella cluster analysis. Il suo funzionamento si fonda su un parametro K specificato dall'utente che corrisponde al numero di clusters desiderati che l'algoritmo costruisce sulla base di altrettanti centroidi. Un centroide o center point, tipicamente, rappresenta la media di un gruppo di punti. Una volta inizializzati i centroidi l'algortimo assegna ciascun punto al centroide più vicino così che ogni collezione di punti assegnata ad un centroide definisce un cluster. Tale step viene ripetuto fin quando i centroidi inizializzati non rimangono i medesimi e nessun punto all'interno dei clusters cambia. Dunque possiamo semplificare l'algoritmo del k-means come:

1. Seleziona K punti come centroidi
2. Ripeti
3. Forma K clusters assegnando ciascun punto al centroide più vicino

4. Ripeti finchè i centroidi non variano

I centroidi vengono generalmente scelti in modo randomico. Questo può rappresentare un limite dell'algoritmo in quanto due diverse esecuzioni di esso potrebbero generare clusters di diversa forma e numerosità sulla base della scelta dei centroidi iniziali. Inoltre, al fine di misurare la distanza o la vicinanza di ciascun punto da un centroide occorre specificare una misura di distanza. Per misure di distanza comuni, come quelle che approfondiremo in seguito, l'algoritmo converge facilmente ad uno stato in cui i centroidi rimangono immutati tanto che la convergenza si ottiene, spesso, nelle prime iterazioni. L'algoritmo K-means, così facendo, fornisce una soluzione completa; ciò significa che assegna l'etichetta di cluster a tutti i dati di input. Questo significa che l'algoritmo non è in grado di individuare eventuali dati rumorosi o outliers all'interno del dataset che dovranno essere, di conseguenza, eliminati in fase di preprocessing.

La complessità computazionale dell'algoritmo è $O(n*k*I*d)$, dove n rappresenta il numero di punti, I le iterazioni, d il numero di attributi ed infine K il numero di clusters specificato.

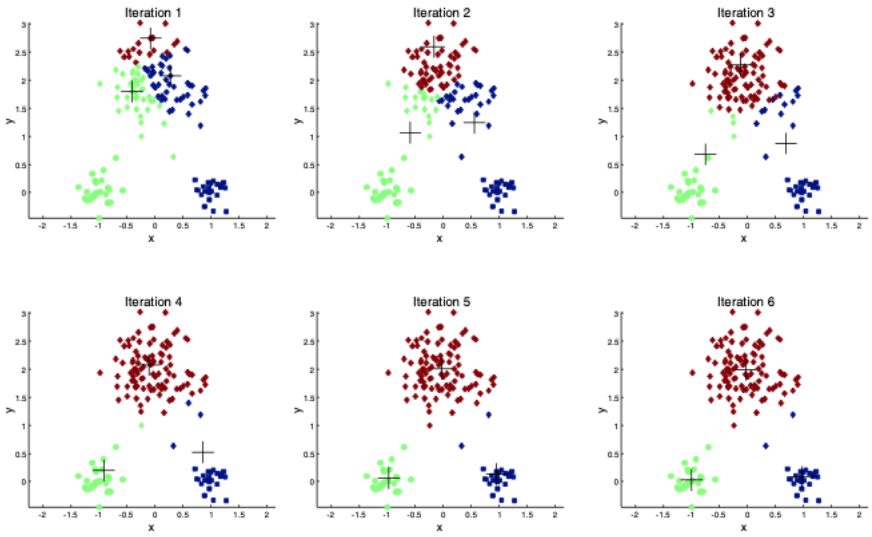

Figura 3.3: Esempio di K-means [10]

Per valutare i clusters prodotti dal K-means si usano metriche qualitative; la più comune è l'SSE (Sum of Squared Error). Ora, per ogni punto, l'errore è rappresen-

tato dalla rispetto al cluster più vicno. Per ottenere l'SSE si elevano al quadrato e si sommano prima tra loro gli errori per ciascun cluster e poi si esegue un'ulteriore somma su tutti i clusters come descritto da:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x) \tag{3.1}$$

dove x è il dato contenuto nel cluster C_i ed m_i rappresenta il centro del cluster (media o centroide). Dunque, dati due clusters o più in generale due partizionamenti, la soluzione migliore a parità di k è quella che minimizza l'SSE.

L'SSE trova un ulteriore utilizzo anche nella scelta del numero di cluster ottimale da specificare in input all'algoritmo. Nel "*Elbow Method*" o "metodo del gomito", infatti, viene valutato il trend di una misura qualitativa, tipicamente l'SSE, al variare del parametro k rappresentativo del numero di clusters. Tale euristica mira all'individuazione di eventuali "gomiti" ossia punti di massima decrescita dell'SSE, anche detti a ritorni marginali decrescenti. Tali punti, infatti, sono tali per cui il vantaggio di aggiungere un ulteriore centroide è trascurabile oppure per cui l'ulteriore riduzione della misura qualitativa non risulta essere interessante.

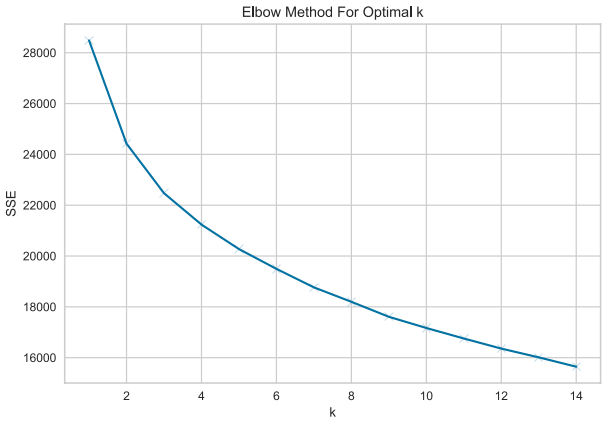

Figura 3.4: Esempio di Elbow Method per la scelta del K ottimale

E' da notare, infine, che eventuali picchi nella curva testimoniano la presenza di clusters vuoti dettati da centroidi non aggiornati. In questo caso, quindi, il numero di clusters effettivamente trovati dalla soluzione è minore dei k inizializzati.

3.2 DBSCAN

Il DBSCAN è un algoritmo di Partitional clustering *density-based*. Esso individua regioni ad alta densità che sono separate tra loro da regioni a bassa densità. Per comprendere il funzionamento dell'algoritmo è, dunque, necessario esplicitare il concetto di densità su cui si basa il DBSCAN. In questo caso si parla di approccio "center-based" in quanto la densità per un particolare punto del dataset è stimata contando il numero di punti contenuti all'interno di area definita da uno specifico raggio (*Eps*). Ciò includendo il punto stesso [10]. In questo caso, dunque, la nozione di densità dipende forteente dal raggio *Eps*; se, infatti, quest'ultimo è troppo ampio allore tutti i punti avranno densità pari all'interno numero di oggetti nel dataset. AL contrario, sei il raggio è troppo piccolo i punti avranno tutti densità unitaria.

L'approccio center-based alla stima della densità permette di classificare i punti del dataset in tre classi:

- **core points**: Sono punti che hanno più di uno specificato numero di punti (*MinPts*) all'interno dellarea individuata dal raggio *Eps*. Questo tipo di punti sono coloro che giacciono all'interno di un cluster.
- **border points**: Sono punti che hanno meno *MinPts* nell'area dettata da *Eps* ma che si trovano comunque nell'intorno di un core point.
- **noise points**: sono tutti quei punti che non possono essere classificati come core points o border points.

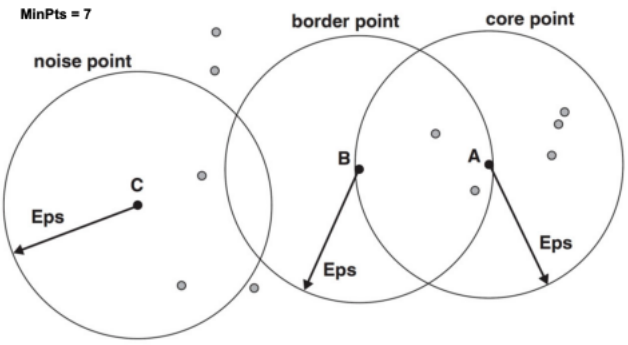

Figura 3.5: Esempio core point, border point e noise point [10]

Introdotte le nozioni di core, border e noise points possiamo studiare il funzionamento dell'algoritmo. Possiamo semplificare il funzionamento dell'algoritmo come segue:

1. Etichetta i punti come core, border o noise point
2. Elimina i noise point o punti rumorosi
3. Definisci l'*Eps-vicinato* per ciascun core point
4. Crea un cluster per ciascun gruppo di core point connessi
5. Assegna ciascun border point al cluster associato al suo core point

Il DBSCAN, dunque, fornisce una soluzione parziale poichè alcuni punti sono etichettati come rumorosi a seconda di come si scelgono i parametri *Eps* e *MinPts*. Inoltre, a differenza del K-means, il punto di partenza non influenza i clusters individuati. I clusters, poi, possono avere cardinalità e forme diverse fornendo, quindi, un partizionamento eterogeneo. Infine, è bene sottolineare come per ogni punto bisogni esplorare il suo *Eps-vicinato* e ciò rende l'algoritmo particolarmente complesso dal punto di vista computazionale. Nel peggiore dei casi, infatti, l'algoritmo impiega m steps per individuare i punti nell'*Eps-vicinato* e la complessità computazionale è pari a $O(m^2)$ dove m rappresenta il numero di punti del dataset.

Come accennato, per il DBSCAN, la scelta dei parametri *Eps* e *MinPts* influenza fortemente la classificazione dei punti e di conseguenza si pone il problema di come sceglierli. L'approccio basilare consiste nel costruire il grafico del *k-dist*, che definisce la distanza da un punto al k-esimo vicino. Per ogni oggetto nel dataset l'algoritmo calcola la distanza tra lui e gli altri punti, ordina i risultati in ordine crescente di distanza e scelglie il k-esimo. l'idea è quindi quella di sceliere il k maggiore tale per cui non cambia troppo la distribuzione. Dal grafico poi, effettuando un taglio rispetto un flesso si possono individuare rispettivamente l'*Eps* e i *MinPts*. L'algoritmo originale del DBSCAN usava un valore di k=4, in quanto appariva ragionevolmente buono per dati bidimensionali.

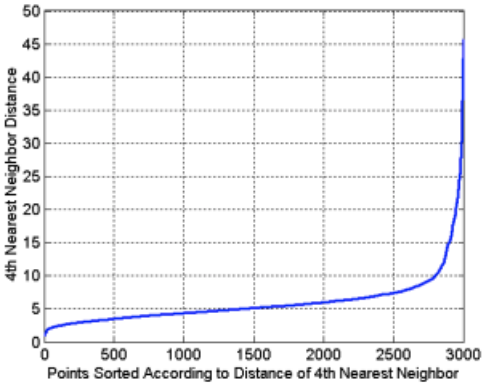

Figura 3.6: Esempio di grafico k-dist [10]

Tra i pregi del DBSCAN troviamo la sua capacità di essere resistente al rumore e di gestire clusters di numerosità e forme diverse. Questo gli permette, quindi, di rilevare clusters che sarebbe impossibile individuare mediante algoritmi come il K-means. Tuttavia, come accennato in precedenza, il DBSCAN ha problemi quando i clusters hanno densità variabili oppure nel caso di dataset che presentano un alto numero di dimension, in quanto il la definizione della densità e il calcolo dei punti all'interno del vicinato possono risultare onerosi in termini computazionali.

3.3 Agglomerative Clustering

Il funzionamento di un algoritmo che opera clustering gerarchico di tipo Agglomerative può essere riassunto come segue:

1. Calcolo della matrice di prossimità
2. Sia ogni punto del dataset un cluster
3. **Ripeti**
4. Unisci i due clusters più vicini
5. Aggiorna la matrice di prossimità
6. **Fino** a che non rimane un unico cluster

Ciò che emerge analizzando la forma schematica dell'algoritmo è che l'operazione chiave eseguita dall'algoritmo è il calcolo della matrice di prossimità ed il suo continuo aggiornamento. Questo può essere fatto in diversi modi a seconda dell'approccio che si sceglie di utilizzare per la definizione della distanza inter-cluster.

Tra i più famosi abbiamo:

- *Single linkage* o MIN
- *Complete linkage* o MAX
- *Average linkage*
- *Ward's Method*

Di seguito viene riportata una descrizione per ciascuno dei diversi approcci.

Per il *Single linkage* la prossimità tra due clusters è definita come la minima distanza (o massima similarità) ottenuta tra qualsiasi coppia di punti appertenenti ai due diversi clusters. Ciò significa che l'algoritmo calcola n misure di distanza dove n sono le coppie di punti tali per cui uno appartiene ad un cluster e il secondo all'altro. Questo tipo di approccio ha il vandaggio di riuscire a gestire cluster di forma non-ellittica ma è risulta sensibile ai dati rumorosi e agli outliers.

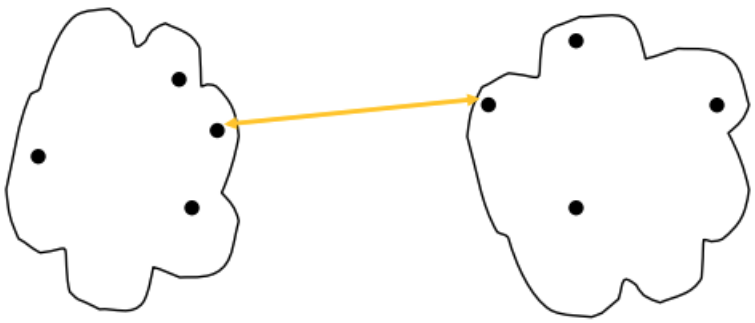

Figura 3.7: Esempio di Single linkage

Nella versione *Complete linkage* dell'Agglomerative Clustering, la matrice di prossimità viene calcolata utilizzando come misura di distanza inter-cluster il massimo della distanza (o minima similarità) calcolata tra tutte le possibili coppie formate

dai punti dei due diversi clusters. Questo approccio permette di avere soluzioni meno suscettibili al rumore agli outliers ma che tendono a suddividere clusters di maggiori dimensioni favorendo la formazione di clusters di forma globulare.

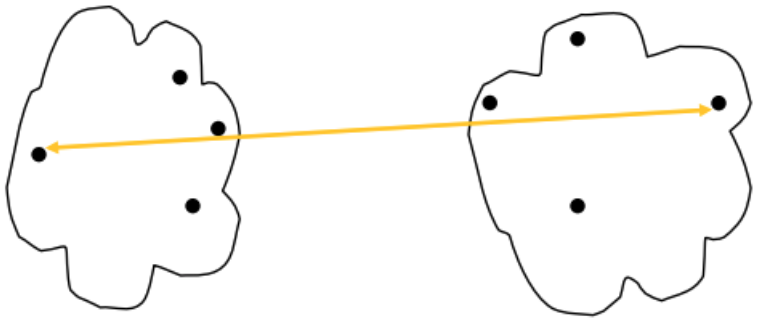

Figura 3.8: Esempio di Complete linkage

Con l'*Average linkage* o Group Average, invece, la distanza tra due clusters viene stimata come la media delle distanze calcolate per tutte le coppie di punti nei due differenti clusters. Esso rappresenta, dunque, una soluzione di compromesso rispetto ai due casi precedenti

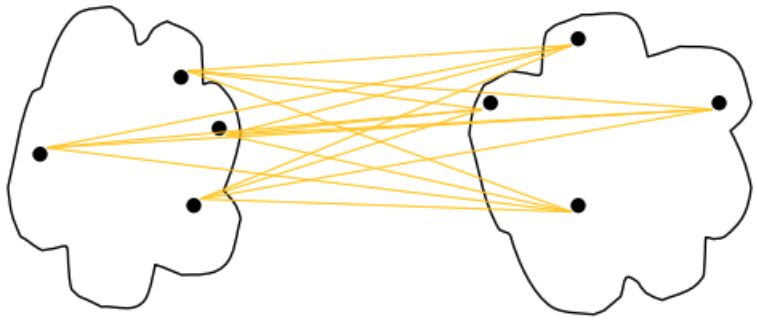

Figura 3.9: Esempio di Average linkage

Il *Ward's Method* si differenzia dai precedenti approcci in quanto stima la prossimità di due clusters sulla base dell'incremento dell'errore quadratico che si ottiene quando due clusters vengono fusi [10]. Tale metodo risulta simile all'Average linkage utilizzato considerando la distanza quadratica tra punti. Esso, inoltre, è meno suscettibile al rumore e agli outliers ma tende a creare cluster globulari.

3.4 Misure di distanza

Gli algortimi di clustering determinano la posizione di un oggetto in un cluster piuttosto che un altro in base a diversi tipi di distanza presi in considerazione. In generale, però, non si può parlare di cluster analysis senza parlare delle funzioni matematiche che gli algoritmi usano per effettuare il raggruppamento in clusters. Esistono numerosi tipi di funzioni matematiche per calcolare la distanza tra due oggetti, ognuna delle quali può risultare più o meno adatta a seconda del caso studio. Nel seguente elaborato vedremo le due principali, ossia la *Distanza Euclidea* e la *Cosine Similarity* utilizzate anche ai fini del progetto di ricerca presentato.

La *Distanza Euclidea* tra due punti in uno spazio euclideo rappresenta la lunghezza del segmento che unisce i due punti[1] La sua formulazione per distanze calcolate tra due punti (p, q) definiti in uno spazio n-dimensionale è:

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2} \tag{3.2}$$

Come metrica di distanza riprende tutte le caratteristiche tipiche degli spazi metrici. Essa, infatti, risulta:

- simmetrica, ossia dati due punti,la distanza tra di essi non dipende dal punto di partenza
- positiva, è un numero reale positivo
- rispettosa della "*Triangle inequality*", tale per cui dati tre punti distinti la somma delle distanze tra due coppie di punti è sempre maggiore della distanza calcolata per la terza coppia.

La *Cosine Similarity*, invece, misura la similarità tra due vettori in termini del coseno dell'angolo compreso tra i due. Così facendo, valuta se i due vettori puntano nella stessa direzione [11]. Essa viene largamente utilizzata nell'ambito della text

[1] `https://en.wikipedia.org/wiki/Euclidean_distance`

analysis dove un testo viene trasformato in un vettore di attributi ciascuno rappresentato da una parola. Essa viene calcolata come il rapporto fra il prodotto scalare dei due vettori e il prodotto delle loro norme.

La sua formulazione matematica è:

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{3.3}$$

dove t ed e rappresentano i due vettori di n componenti.

3.5 Metriche di valutazione

La parte più difficile di un'analisi di clusters è la validazione dei clusters stessi. Al fine di valutare la bontà dei risultati di un algoritmo di clustering si possono valutare alcune misure numeriche. Esse possono essere classificate in due macro-categorie:

- Indici Esterni: utilizzati per misurare quanto le etichette di cluster combaciano con etichette di classe esterne fornite a priori.
- Indici Interni: usati per misurare la bontà della struttura dei cluster senza far riferimento ad informazioni esterne

Nel seguente elaborato verranno approfonditi gli indici interni. Essi sono rappresentati dall'SSE, la *Coesione* dei clusters, la loro *Separazione* ed, infine, la Silhouette.

La *Coesione* dei clusters misura quanto simili risultano essere gli oggetti posti all'interno di un cluster. Essa si misura mediante l'SSE interno del cluster (WSS).

$$WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2 \tag{3.4}$$

La *Separazione*, invece, misura quanto distanti, ossia ben separati, sono i clusters tra loro. Essa si misura in termini di SSE tra clusters (BSS).

$$BSS = \sum_{i} |C_i|(m - m_i)^2 \tag{3.5}$$

L'ideale, dunque, sarebbe ottenere livelli alti di coesione e separazione minimizzando cosi la distanza degli oggetti all'interno dello stesso cluster e massimizzando la distanza inter-cluster.

Per facilitare l'analisi di coesione e separazione vi è un ulteriore indicatore sintetico definito Silhouette. Ora, per ogni oggetto i:

- sia $a(i)$ la distanza media di i rispetto agli altri elementi dello stesso cluster (misura di coesione)
- sia $b(i)$ la minima distanza media tra i e ciascun altro cluster di cui i non fa parte

Possiamo definire la Silhouette come:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3.6}$$

oppure

$$s(i) = \begin{cases} 1 - a(i)/b(i), & a(i) < b(i) \\ 0, & a(i) = b(i) \\ b(i)/a(i) - 1 & a(i) > b(i) \end{cases} \tag{3.7}$$

Guardando alla Silhouette media su tutto il dataset possiamo avere un'idea quantitativa del risultato del clustering. In particolare, più il suo valore tende ad 1 e più il partizionamento è buono.

Capitolo 4

Classificazione delle inquadrature cinematografiche

Nel presente lavoro, le tecniche di Machine Learning precedentemente presentate, nella fattispecie quelle di Unsupervised Learning, sono state utilizzate nel contesto dell'analisi di sequenze di inquadrature cinematografiche.

Per sequenza cinematografica si intende un insieme di inquadrature tali da esaurire un episodio narrativo[1]. Una sequenza, tuttavia, può essere composta anche da una sola inquadratura; in questo caso parliamo di piano-sequenza. Ogni opera cinematografica, dunque, è realizzata a partire da ciascuna sequenza, che, a sua volta, è il risultato dell'unione di diverse inquadrature la cui relazione viene esplicitata grazie al montaggio. L'inquadratura rappresenta l'elemento base di un'opera cinematografica oltre che il mezzo attraverso il quale un autore o un regista è in grado di veicolare una personale e particolare visione. Grazie, infatti, alle relazioni tra le varie inquadrature è possibile individuare un registro stilistico nel racconto che sarà in grado di suscitare nello spettatore un'emozione piuttosto che un'altra.

In questo Capitolo analizzeremo la divisione delle diverse tipologie di inquadrature soffermandoci sulle otto scelte per il nostro caso studio. A tale scopo occorre premettere che un'inquadratura prende il nome di *piano*, quando la ripresa vede una figura umana protagonista, mentre viene definita *campo*, quando raffigura un

[1] `https://it.wikipedia.org/wiki/Sequenza_(cinema)`

ambiente o un paesaggio. Le tipologie di inquadrature scelte per il presente studio sono, come accennato, otto: *Campo Lungo (CL), Campo Medio (CM), Figura Intera (FI), Piano Americano (PA), Mezza Figura (MF), Mezzo Busto (MB), Primo Piano (PP) e Primissimo Piano (PPP)*. Nel Campo Lungo o Long Shot, l'ambiente è dominante rispetto alla figura umana della quale si percepisce esclusivamente la presenza. A differenza di quanto avviene nel Campo Lunghissimo (un tipo di campo non considerato nell'analisi svolta in questo elaborato), in cui non vi è alcuna traccia della figura umana, nel Campo Lungo lo spazio individuato dalla camera è ridotto e la figura umana assume dei contorni riconoscibili rendendo lo spettatore in grado di collocare la figura all'interno dell'ambiente (tipicamente esterno) rappresentato.

Figura 4.1: Esempio di Campo Lungo

Il Campo Medio (CM) o Medium Shot, pur rappresentando ancora l'ambiente circostante in maniera preponderante, vede un progressivo aumento di importanza della figura umana che risulta abbstanza vicina da poter riconoscere il soggetto e la sua azione.

Figura 4.2: Esempio di Campo Medio

Avvicinandosi ulteriormente alla figura umana, le inquadrature definite dalla camera prendono, come detto, il nome di piani. La prima tipologia di piano, risulta essere la Figura Intera (FI) o Full Figure. In essa la figura umana viene inquadrata nella sua interezza, dai piedi alla testa. Si tratta della prima inquadratura in cui il personaggio acquista predominanza sull'ambiente e, in virtù di ciò, viene utilizzata per mettere in risalto la fisicità dell'attore[2].

Figura 4.3: Esempio di Figura Intera tratto dal film *Joker di Todd Philips*

La tipologia di piano subito successiva alla Figura Intera è definita Piano Americano (PA) o American Shot, in quanto largamente usata nella cosiddetta "età d'oro di Hollywood". Qui, la figura umana risulta inquadrata dalle ginocchia in su. Tale tipo di inquadratura serviva in passato per enfatizzare le movenze dei protagonisti

[2]`https://www.dreamvideo.it/articoli/221/l-inquadratura-campi-e-piani`

dei film Western, inquadrandoli dalla testa alla pistola posta poco al di sotto della cintura.

Figura 4.4: Esempio di Piano Americano tratto dal film *The Hateful Eight di Quentin Tarantino*

Riducendo ulteriormente la distanza tra la camera e il soggetto umano troviamo il Piano Medio o Mezza Figura (MF). Questa inquadratura si concentra sui personaggi e l'ambiente in cui agiscono perde quasi di significato. Comprende la parte superiore della figura tagliata alla vita e viene utilizzata nei casi in cui si riprende l'interazione di due personaggi in stretta vicinanza enfatizzando la relazione che si sta instaurando tra loro[3].

[3] `https://www.griffithduemila.com/art/tipi-di-inquadrature-cinematografiche-campi-e-piani.html`

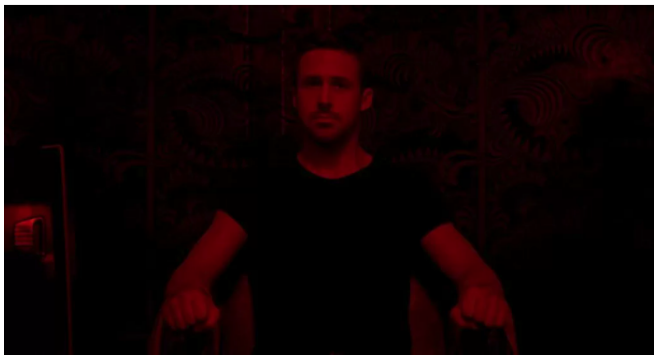

Figura 4.5: Esempio di Mezza Figura tratto dal film *Solo Dio perdona di Nicolas Winding Refn*

Il Mezzo Busto (MB) o Half Torso, è una variante della Mezza Figura nella quale il protagonista della scena è ripreso dalla testa fino al petto, dando ancora più risalto al protagonista.

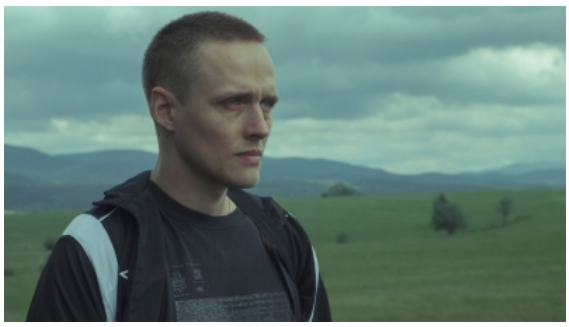

Figura 4.6: Esempio di Mezzo Busto

A seguire abbiamo il Primo Piano (PP) o Close Up che si caratterizza per la ripresa della testa e delle spalle del personaggio. Quest'inquadratura viene utilizzata per dare risalto allo stato d'animo del personaggio, sottolineandone la psicologia ed è tipica delle scene di dialoghi tra personaggi.

Figura 4.7: Esempio di Primo Piano tratto dal film *Lei di Spike Jonze*

Infine, vi è Primissimo Piano (PPP). Tale tipo di inquadratura rivela esclusivamente il volto del personaggio isolandolo dal corpo e dall'ambiente mostrando nel dettaglio anche le più piccole espressioni del soggetto.

Figura 4.8: Esempio di Primissimo Piano tratto dal film *Shining di Stanley Kubrick*

Ai fini dell'analisi le otto inquadrature sono state etichettate con alltrettanti numeri da 0 a 7 in ordine decrescente di distanza della camera dal soggetto dell'inquadratura (0 per il Campo Lungo fino a 7 per il Primissimo Piano)

Capitolo 5

Pipeline di analisi

5.1 Creazione del dataset

Il Dataset utilizzato per l'analisi è stato creato a partire da video cinematografici presenti nel dataset YoutubeM8[1] e in altre risorse video online, tra le quali, la più importante è stata il canale Youtbe *Movieclips*[2]. In particolare, l'obiettivo principale è stato quello di ottenere per ciascun film una serie di immagini, ciascuna rimandante ad una scena nella clip, da poter classificare in una delle otto differenti classi presentate nel capitolo precedente. Così facendo, si è ottenuta una sequenza di inquadrature associata a ciascuna clip.

Per la prima fase del processo di creazione del Dataset, ossia quella del download di frae cinematografici, è stato realizzato un apposito script Python in grado di automatizzare il lavoro. Le librerie utilizzate sono state Pandas, Numpy, PySceneDetect, cv2, os, shutil e Pafy. Pandas rappresenta una delle librerie più usate nel campo della Data Science in quanto fornisce strumenti per ad hoc per gestire dati in forma tabulare e serie ed creare statistiche sui dati; essa si basa su una struttura dati vettoriale, ossia gli array di Numpy, libreria, invece, che implementa numerose funzioni matematiche e di algebra lineare. Per quanto riguarda l'effettiva gestione dei video le librerie utilizzate sono state PySceneDetect, tool di analisi video in grado

[1] `https://research.google.com/youtube8m/explore.html`

[2] `https://www.youtube.com/user/movieclips`

di dividere video in clip ed individuarne il contenuto e/o le scene, e Pafy, utilizzata per download diretto dei frame tramite link della clip. Infine, i frame scaricati e salvati su pc sono stati organizzati in directory e collezionati grazie alle librerie cv2, os e shutil.

Lo script per la creazione del database di immagini cinematografiche prevede, dapprima, l'importazione delle librerie appena descritte e gli URLs delle clip cinematografiche opportunamente organizzati in un file Excel come segue:

```
import pandas as pd
import numpy as np

# Standard PySceneDetect imports:
from scenedetect import VideoManager
from scenedetect import SceneManager

# For content-aware scene detection:
from scenedetect.detectors import ContentDetector
import cv2
import os
import shutil
import pafy
```

Nello specifico, i dati relativi l'URLs e il titolo delle clip cinematografiche sono state importate in un DataFrame di Pandas specificando la posizione del file Excel all'interno del computer.

```
df_test=pd.read_excel('/Users/simonemagnafico/Desktop/Test.xlsx')
```

Si è passati poi all'individuazione delle scene all'interno di ciascuna clip, tenendo traccia dei frame di inizio e di fine di ogni scena oltre che del frame mediano per ciascuna di essa. La funzione utilizzata per identificare le diverse scene è stata realizzata partendo da alcune risorse open source trovate online[3] mostranti l'utilizzo dei

[3] https://github.com/Breakthrough/PySceneDetect/

metodi della libreria PySceneDetect. Tale funzione sfrutta i metodi VideoManager() e SceneManager() per caricare il video, individuare le scene ed, infine, restituire una lista di tuple contenenti i timecodes di inizio e di fine di ciascuna scena rilevata. Gli unici parametri da specificare in ingresso alla funzione sono il path del video da analizzare ed il parametro *threshhold* attraverso il quale si modifica la sensitività della rilevazione delle scene.

```
def find_scenes(video_path, threshold=30.0):
    # Create our video & scene managers, then add the detector.
    video_manager = VideoManager([video_path])
    scene_manager = SceneManager()
    scene_manager.add_detector(
        ContentDetector(threshold=threshold))

    # Improve processing speed by downscaling before processing.
    video_manager.set_downscale_factor()

    # Start the video manager and perform the scene detection.
    video_manager.start()
    scene_manager.detect_scenes(frame_source=video_manager)

    # Each returned scene is a tuple of the (start, end) timecode.
    return scene_manager.get_scene_list()
```

Ora, una volta specificata la directoy in cui salvare i frame di ciascuna clip, sono state utilizzate le funzioni new() e getbestvideo() di Pafy per poter dare in input alla funzione find_scene() l'url del video. Ciò è stato, inoltre, implementato in un ciclo for tale da iterare l'operazione di individuazione delle scene su tutti gli URLs dei video presenti nel file Excel. Inoltre, tramite un ulteriore ciclo for annidato questa volta operato sulle scene individuate per ciascun film, sono stati salvati in un DataFrame di Pandas i frame di inizio e fine scena oltre che il frame mediano, calcolato come il frame intermedio di ogni scena. Proprio quest'ultimo frame, infatti, verrà scaricato. Una volta fatto ciò, tramite la libreria os, viene creata iterativamente una nuova cartella per ciascun film, mano a mano che essi viengono processati. Un ciclo while

infinito è stato in seguito implementato per scorrere i frame uno ad uno tra quelli intermedi di ogni scena in modo tale da eseguire il download all'interno della cartella precedentemente creata e specificata nella variabile newpath. Ciascun frame è stato, inoltre, salvato nella directory con nome del film e numero di frame corrispondente. Esauriti i frame intermedi di ogni scena del film in esame il ciclo while si interrompe, le finestre aperte vengono chiuse e il ciclo for iniziale inizializza un nuovo video da processare. Segue il codice implementato.

```
frame_start=[]
frame_end=[]
frame_inter=[]

dest='/Users/simonemagnafico/Desktop/Video_frame'
current_frame=0

for i in range(0,len(df_test)):
    url=df_test.iloc[i,0]
    title=df_test.iloc[i,1]

    vPafy=pafy.new(url)
    play=vPafy.getbestvideo(preftype='webm')
    scenes = find_scenes(play.url)

    for scene in scenes:
        frame_start.append(scene[0].frame_num)
        frame_end.append(scene[1].frame_num)
        if (scene[0].frame_num+scene[1].frame_num)%2==0:
            frame_inter.append(int((scene[0].frame_num+scene[1].
                                    frame_num)/2))
        else:
            frame_inter.append(int((scene[0].frame_num+scene[1].
                                    frame_num+1)/2))

    df_start=pd.DataFrame(frame_start, columns=['Start Frame'])
    df_end=pd.DataFrame(frame_end, columns=['End Frame'])
    df_inter=pd.DataFrame(frame_inter, columns=['Inter. Frame'])
```

```
df_frame=pd.concat([df_start, df_end,df_inter],axis=1)

video=cv2.VideoCapture(play.url)

newpath = r'/Users/simonemagnafico/Desktop/Video_frame/{}'.
                                    format(title)

try:
# creating a folder named data
    if not os.path.exists(newpath):
        os.makedirs(newpath)

# if not created then raise error
except OSError:
    print ('Error: Creating directory of data')

while(True):
    ret, frame=video.read()

    if ret:
        if current_frame in df_frame['Inter. Frame'].values:
            name=title+str(current_frame)+'.jpg'
            print("Creating..." + name)
            cv2.imwrite(name, frame)
            shutil.move(name, newpath)
            current_frame+=1
        else:
            current_frame+=1
    else:
        break

video.release()
cv2.destroyAllWindows()
```

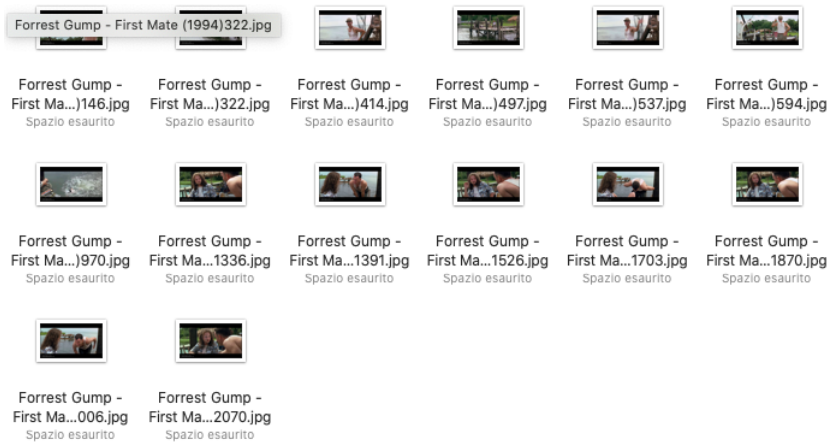

Figura 5.1: Directory Forrest Gump con i rispettivi frames individuati per ciascuna scena

Ora, una volta aver salvato per ciascuna clip cinematografica i frame relativi ad ogni scena individuata, è stato necessario un ulteriore step al fine di ottenere il Dataset necessario per l'analisi in questione. I frames cinematografici sono stati, infatti, classificati nelle otto diverse classi di inquadrature cinematografiche prese in esame mediante un classificatore basato su CNN realizzato dall'Ing. Bartolomeo Vacchetti e il suo gruppo di ricerca [9]. Date in input le immagini estrapolate dalle clip, è stato possibile ottenere per ciascun video una sequenza di inquadrature tra le otto differenti tipologie prese in esame (discretizzate con etichette numeriche da 0 a 7) la cui lunghezza è pari al numero di frame salvati per ogni clip cinematografica.

Il Dataset è stato, poi, completato aggiungendo il frame finale di ogni clip. Grazie ad esso è stato possibile calcolare il *runtime*, ottenuto dal rapporto tra frame finale della clip e frame rate del video, ed il *mean frame time*, risultato del rapporto tra frame finale e numero di inquadrature all'interno della sequenza. In particolare questo secondo indicatore, è stato analizzato al fine di esplicitare evntuali relazioni tra genere e tecniche di montaggio. Infine, il dataset è stato completato con le etichette del genere di appartenza di ciascuna clip scelte tra: Drama, Action, Comedy ed Horror.

	sequence	movie	end frame	runtime	Genre	mean frame time
0	[4, 2, 6, 6, 6, 2, 2, 2, 6, 6, 5, 5, 3, 5, 5, ...	Dirty Dancing - Hungry Eyes (1987)	4320	144.0	Drama	154
1	[5, 6, 6, 5, 5, 6, 5, 5, 2, 5, 5, 6, 5, 6, 3, ...	10000 BC (2008)	3270	109.0	Action	68
2	[6, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4]	12 Monkeys (1955)	2820	94.0	Drama	176
3	[5, 5, 1, 2, 5, 4, 4, 2, 5, 5, 5, 2, 5, 3]	12 Years A Slave (2013)	2716	91.0	Drama	194
4	[5, 5, 5, 5, 5, 3, 1, 5, 5, 5, 2, 1, 2, 5, 2, ...	1941 (1979)	2184	73.0	Action	114

Figura 5.2: Visualizzazione dei primi 5 records del Dataset

5.2 Analisi esplorativa

Il dataset così creato si compone di 937 records suddivisi nei quattro diversi generi come da Tabella 5.1.

Drama	**282**
Action	**278**
Comedy	**294**
Horror	**83**
Totale	**937**

Tabella 5.1: Composizione del dataset

Come si evince dalla Figura 5.2, il dataset è caratterizzato da 6 diversi attributi. Il primo passo dello studio è stato rappresentato, dunque, dall'analisi esplorativa delle features. In particolare, sono state indagate le proprietà e le distribuzioni del runtime e del mean frame time, il cui calcolo è stato esplicitato nel paragrafo precedente, oltre che della lunghezza delle sequenze di inquadrature.

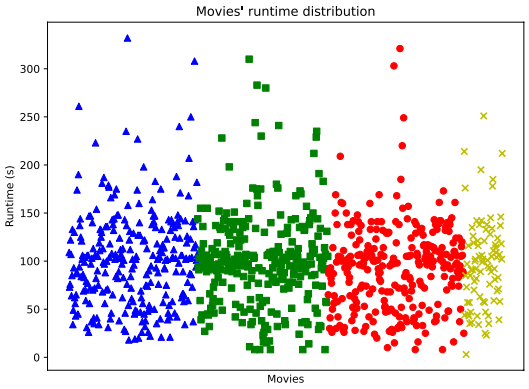

Figura 5.3: Distribuzione del runtime suddivisa per generi

Il runtime misura la durata di ciascuna clip cinematografica presente all'interno del dataset. La Figura 5.3 mostra come i records presentino un runtime medio di 125 secondi e si distribuiscano nella maggior parte dei casi tra i 30 e i 150 secondi, con pochi elementi che arrivano a durare fino a 300 secondi. I records sono stati raggruppati nel grafico per genere; in blu abbiamo le clip di genere Drama, in verde i film Action, in rosso i Comedy ed, infine, il giallo è stato associato agli elementi di genere Horror.

Il Mean frame time, invece, misura la durata media di ciscuna scena (rappresentata dall'inquadratura corrispondente) all'interno della sequenza. Per ottenere il suo valore in secondi, tale indicatore, calcolato come descritto nella sezione precedente, è stato diviso per il frame rate delle clip, pari a 30 $frame/s$. La Figura 5.4 mostra come la durata media delle scene all'interno delle clip cinematografiche si concentri nell'intervallo tra i 2 e i 10 secondi, con pochi outliers che superano i 20 secondi e arrivano fino ad un mean frame time di 80 secondi. La media del Mean frame time è pari a circa 5 secondi. Anche in questo caso vale la suddivisione in generi adoperata per il runtime.

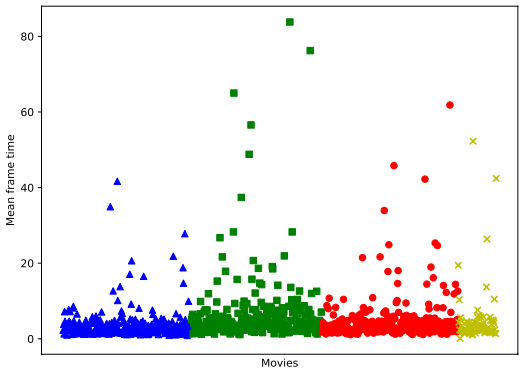

Figura 5.4: Distribuzione del Mean frame time suddivisa per generi

Una volta analizzati il runtime ed il mean frame time si è proceduto con l'esplorazione delle sequenze di inquadrature. La Figura 5.5 permette di visualizzare la distribuzione della lunghezza delle sequenze. In particolare si osserva che gran parte di esse presentino una lunghezza compresa tra le 10 inquadrature e le 30. Poichè trattare sequenze di diversa lunghezza avrebbe rappresentato un problema in fase di analisi si è scelto, in fase di preprocessing, di ricondursi ad un problema definito in uno spazio dimensionale ridotto scegliendo sequenze aventi tutte la stessa lunghezza. A tale fine l'analisi è stata effettuata sia con sequenze di lunghezza pari a 10, ottenute scegliendo le prime 10 inquadrature di tutte le sequenze aventi lunghezza superiore o pari ad essa, sia con sequenze composte da 20 inquadrature, ottenute in modo analogo.

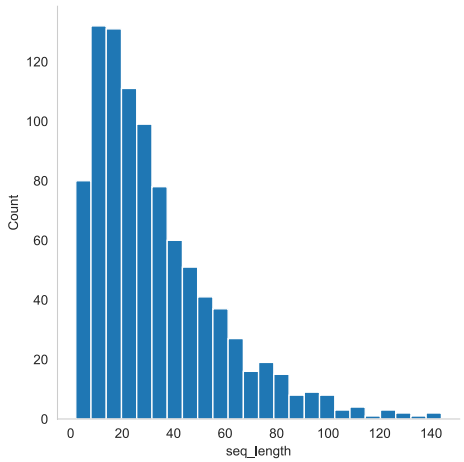

Figura 5.5: Distribuzione della lunghezza delle sequenze

Di seguito sono riportate le suddivisioni in termini di genere delle sequenze aventi lunghezza ripsettivamente pari a 10 e 20 inquadrature. In totale, dunque, si è scelto di effettuare l'analisi unsupervised sulle 821 sequenze di 10 inquadrature e sulle 594 sequenze di lunghezza pari a 20, andando dapprima ad eseguire il clustering imponendo un numero di clusters pari a 4 per cercare di isolare il genere e, in seguito, scegliendo un numero variabile di clusters tale per cui potessero emergere pattern ricorrenti all'interno delle sequenze del dataset.

Drama	**232**
Action	**259**
Comedy	**253**
Horror	**77**
Totale	**821**

Tabella 5.2: Sequenze di 10 elementi

Drama	**149**
Action	**204**
Comedy	**183**
Horror	**58**
Totale	**594**

Tabella 5.3: Sequenze di 20 elementi

5.3 Clustering su sequenze di dieci inquadrature

Il primo approccio di analisi adoperato ci ha visto testare diversi algoritmi di clustering sulle sequenze di 10 inquadrature (Tab. 5.2) al fine di isolare il genere delle

sequenze, creando cosi gruppi o clusters composti, idealmente, di sequenze dello stesso genere. Per far ciò è stato creato un apposito dataframe di Pandas caratterizzato da tante righe quante sono le sequenze di 10 inquadrature e un numero di colonne pari al numero di inquadrature all'interno di ciascuna sequenza, ossia dieci (come si osserva in Figure 5.6).

	element0	element1	element2	element3	element4	element5	element6	element7	element8	element9
0	4	2	6	6	6	2	2	2	6	6
1	5	6	6	5	5	6	5	5	2	5
2	6	4	5	5	5	5	5	5	5	5
3	5	5	1	2	5	4	4	2	5	5
4	5	5	5	5	5	3	1	5	5	5
...	...	...	...	...	...	...	...	...	...	...
816	0	6	5	0	5	1	6	2	6	2
817	5	5	6	6	6	5	7	6	5	2
818	6	6	6	2	6	4	6	5	6	6
819	1	2	5	5	1	2	5	4	2	0
820	5	5	5	4	5	6	5	5	5	5

Figura 5.6: Dataframe contenente le sequenze di 10 inquadrature

La singola sequenza è stata, dunque, trattata come un record del dataframe caratterizzato da 10 features.

Una volta preparati i dati per l'analisi si è proceduto con la scelta degli algoritmi di clustering da testare su di essi. Per il seguente studio sono stati utilizzati sia algoritmi di tipo partizionale e sia algoritmi di tipo gerarchico, come descritto nel Capitolo 1. In particolare essi sono:

- *K-means*: algoritmo partizionale di tipo center-based
- *DBSCAN*: algoritmo partizionale di tipo density-based
- *Agglomerative Clustering*: algoritmo di clustering gerarchico di tipo agglomerativo
- *Spectral Clustering*: algoritmo di clustering che fa uso dello spettro della matrice di similarità dei dati per eseguire la riduzione della dimensionalità prima del raggruppamento in meno dimensioni.

Inoltre, ciascun algoritmo è stato testato con due differenti misure di distanza in base alla quale determinare l'appartenenza di una sequenza ad un cluster piuttosto che ad un altro. Esse sono:

- *Distanza Euclidea*:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{5.1}$$

- *Cosine similarity*:

$$\cos(\mathbf{t},\mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{5.2}$$

Per l'implementazione dei test sono state utilizzate le implementazioni dei diversi algoritmi di clustering fornite dalla libreria *Scikit-Learn*[4]. Per ciascun algoritmo è stato specificato il numero di clusters desiderato, che nel seguente caso è stato scelto pari a 4 (ossia pari al numero dei generi cinematografic), la metrica di distanza o "affinity" e, per il caso del K-Means, il numero di inizializzazioni dei centroidi oltre al numero massimo di iterazioni. Nel caso dell'Agglomerative Clustering è stato, invece, necessario il linkage da utilizzare, essendo quest'ultimo un algoritmo di Hierarchical Clustering. Tramite poi la funzione *fit_predict()* di ciascun algoritmo è stato possibile dare in input il dataframe contenente le sequenze sulle quali operare il clustering. Infine, le etichette di cluster sono state assegnate come una nuova feature del dataset in modo tale da tenerne traccia.

Numerosi esperimenti sono stati effettuati con ciascuna combinazione di algoritmi e metriche di distanza. Nel seguente elaborato verranno presentati i risultati migliori ottenuti in ciascun caso, ponendo l'attenzione sull'interpretazione di essi. Inoltre, al fine di evitare di appesantire la trattazione, saranno tralasciati i test effettuati con il *DBSCAN*, in quanto, attraverso di esso non è stato mai possibile individuare più di un cluster, fatta eccezione per il caso di due cluster di cui uno rappresentativo dei dati ritenuti dall'algoritmo rumorosi presenti all'interno del dataset in input. Ciò è stato dovuto alla variabilità e alla sparsità dei dati presenti

[4] `https://scikit-learn.org/stable/modules/clustering.html`

nel dataset che non hanno permesso di individuare cluster distinti in maniera netta mediante l'algoritmo density-based.

Nel caso in esame, ossia di clustering sulle sole sequenze di lunghezza pari a dieci inquadrature, con un numero di clusters imposto pari a quattro (poichè quattro sono i generi scelti) i migliori risultati sono stati ottenuti con il K-means, con entrambe le metriche si distanza, con lo Spectral Clustering, scegliendo la cosine similarity come *"affinity"* ed, infine, con l'Agglomerative Clustering, con parametri cosine similarity e linkage ward.

Per ciascun test vengono riportati i risultati in tabella. In particolare, in essa si ritrovano il numero di clip, classificate secondo il genere di appartenza, presenti in ciascun cluster. Infine, nell'ultima riga della tabella sono presenti le sequenze totali raggruppate in ciascun cluster ed il numero totale di elementi che compongono il dataset di sequenze di 10 inquadrature.

K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	64	83	46	39	232
Action	50	97	52	60	259
Comedy	65	93	53	42	253
Horror	13	30	19	15	77
TOT	**192**	**303**	**170**	**156**	821

Tabella 5.4: K-means con distanza euclidea

Cosine K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	35	38	68	91	232
Action	56	43	51	109	259
Comedy	44	37	79	93	253
Horror	16	16	18	27	77
TOT	**151**	**134**	**216**	**320**	821

Tabella 5.5: K-means con Cosine Similarity

Spectral Clustering	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	86	72	37	37	232
Action	92	58	50	59	259
Comedy	84	80	48	41	253
Horror	28	19	19	11	77
TOT	**290**	**229**	**154**	**148**	821

Tabella 5.6: Spectral Clustering con Cosine Similarity

Agglomerative Clustering	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	96	39	72	25	232
Action	88	62	73	36	259
Comedy	99	49	77	28	253
Horror	33	17	18	9	77
TOT	**316**	**167**	**240**	**98**	821

Tabella 5.7: Agglomerative Clustering con distanza euclidea e linkage Ward

Come si evince dalle tabelle sopra riportate, in nessun esperimento è stato possibile isolare il genere delle sequenze. E' evidente, infatti, che in nessun test si è riusciti ad ottenere clusters con una prevalenza netta di un genere rispetto all'altro, anzi. Ciò che traspare è che all'interno di ciascun cluster vengano riprodotte le proporzioni di clip di generi differenti presenti già all'interno del dataset di sequenze di 10 inquadrature. Per questo tipo di sequenze è, dunque, chiara l'impossibilità di riprodurre mediante algoritmi di clustering la suddivisione delle clip in base al genere.

Valutazione dei clusters

Si è proceduto, dunque, alla valutazione e alla caratterizzazione dei risultati del clustering. Per la valutazione delle varie tecniche di clustering è stata utilizzata la Silhouette. Come accennato in precedenza, la Silhouette rappresenta una metrica di

validazione e interpretazione di quanto simile è un oggetto rispetto agli altri elementi del suo stesso cluster (coesione) e quanto risulta distante da oggetti di altri clusters (separazione). Tanto più il valore della Silhouette media tende a 1, tanto più i clusters saranno coesi e ben separati. I valori del Silhouette score medio per ciascun algoritmo sono riportati in Tabella 5.8.

Algoritmo	**Silhouette score**
K-Means	0.11
Cosine K-Means	0.16
Spectral Clustering	0.15
Agglomerative Clustering	0.05

Tabella 5.8: Silhouette score medio per ciascun algoritmo

I valori di Silhouette ottenuti per ciascun algoritmo risultano molto bassi. I punteggi migliori si ottengono mediante il K-means utilizzando la cosine similarity come metrica di distanza e lo Spectral Clustering con medesima misura. Ciò avviene in quanto la Cosine similarity riesce meglio della distanza Euclidea a definire la distanza di due elementi su spazi n-dimensionali. Se due oggetti, infatti, risultano molto distanti in termini di distanza euclidea potrebbero, comunque, risultare simili se la loro orientazione è simile; ossia se l'angolo sotteso dai rispettivi versori risulta acuto.

Ora, sebbene la Silhouette fornisca una valutazione oggettiva dei risultati di una cluster analysis in termini di coesione e separazione, non si può prescindere dal valutare i clusters in termini qualitativi effettuando una caratterizzazione degli stessi.

Per fare ciò si è cercato di valutare qualitativamente la distribuzione dei dati rispetto alle etichette di cluster mediante la PCA. La Principal Component Analysis è una tecnica di riduzione della dimensionalità che produce un cambio di base all'interno dei dati calcolando le componenti principali, ossia quel ridotto numero di componenti che spiega la maggior parte della variabilità all'interno dei dati. Nel caso in esame, sebbene le prime due PC riescano ad esplicitare poco meno del 40% (Fig.

5.7), la PCA è stata utilizzata per visualizzare la distribuzione dei dati rispetto alle due principal components confrontando le etichette prodotte dalla cluster analysis con il genere associato a ciascuna sequenza di 10 inquadrature.

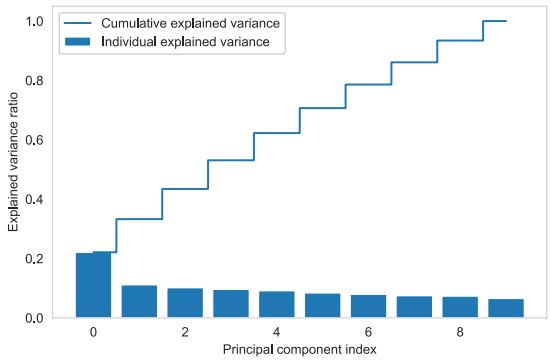

Figura 5.7: Varianza cumulativa esplicitata dalle Principal Components

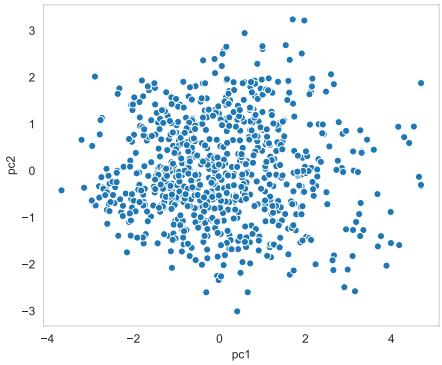

Figura 5.8: Distribuzione dei dati rispetto alle prime due PC

Di seguito, vengono riportati i risultati del confronto nei casi sperimentali analizzati in precedenza. A sinistra, nei grafici, si osserva la distribuzione dei dati rispetto le due PC etichettati secondo genere, a destra secondo i risultati prodotti dal clustering.

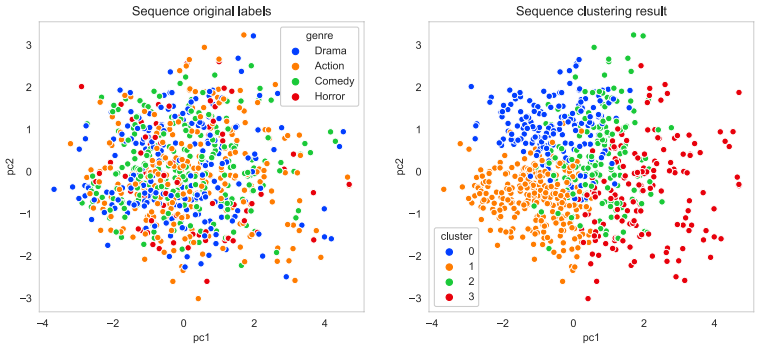

Figura 5.9: K-Means con Distanza Euclidea

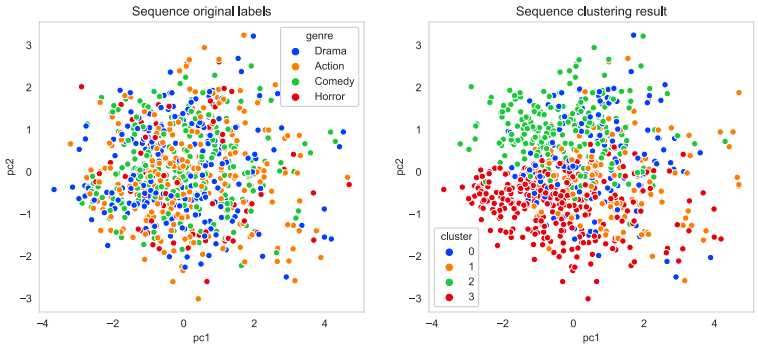

Figura 5.10: K-Means con Cosine Similarity

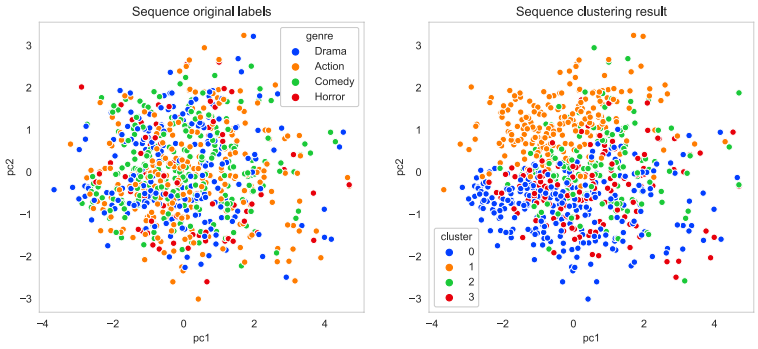

Figura 5.11: Spectral Clustering con Cosine Similarity

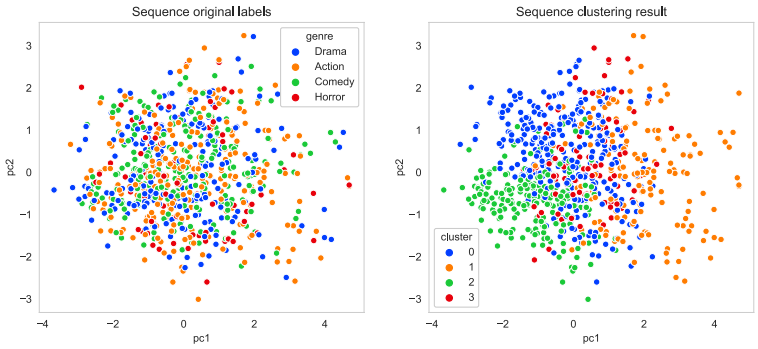

Figura 5.12: Agglomerative clustering con distanza euclidea e linkage Ward

Nonostante alcuni algoritmi sembrino riprodurre clusters ben separati e poco sovrapposti, come accade ad esempio nel caso del K-means con distanza euclidea (Fig. 5.9), nessun algoritmo riesce a replicare un raggruppamento dei dati simile a

quello che si ottiene selezionando le etichette di genere operando sulle sole sequenze di 10 elementi.

Non potendo, dunque, isolare il genere delle sequenze si è tentato di indagare gli elementi sulla base dei quali ciascun algoritmo raggruppasse i dati in clusters, cercando di individuare evntuali caratteristiche o pattern ricorrenti all'interno dei clusters.

Caratterizzazione dei clusters

L'interpretazione dei risultati e la comprensione dei cluster prodotti rappresentano il punto chiave di un processo di cluster analysis. Come visto, la Silhouette e le visualizzazioni effettuate sfruttando tecniche di riduzione della dimensionalità come la PCA, forniscono una valutazione quantitativa la prima, e qualitativa la seconda, che tuttavia poco dicono circa le caratteristiche comuni delle quali godono gli oggetti appartenenti ad uno stesso cluster.

Essendo impossibile estrarre conoscenza osservando i risultati prodotti dagli algoritmi di clustering per via dell'elevato numero di features e di records, l'interpretazione e la caratterizzazione di clusters, nel seguente studio, sono state eseguite mediante una funzione di report in grado di estrarre delle regole da ciascun cluster prodotto in output.

La funzione *cluster_report*[5] utilizzata genera automaticamente delle regole addestrando un modello di Decision Tree (o Albero di Decisione) sulle features originali e sfruttando i risultati del clustering come etichette di classe. Di seguito viene riportato il codice Python che implementa la funzione di report. Le librerie utilizzate per la realizzazione della funzione sono principalemente quelle contenenti i modelli di Decision Tree in Scikit learn e i metodi di display ed HTML presenti in IPython che permettono, invece, di visualizzare il report contenente le regole estratte.

```
from IPython.display import display, HTML
from sklearn.tree import _tree, DecisionTreeClassifier


def pretty_print(df):
```

[5] da `towardsdatascience.com/the-easiest-way-to-interpret-clustering-result-8137e488a127`

```
    return display( HTML( df.to_html().replace("\\n","<br>") ) )

def get_class_rules(tree: DecisionTreeClassifier, feature_names:
                                    list):
  inner_tree: _tree.Tree = tree.tree_
  classes = tree.classes_
  class_rules_dict = dict()

  def tree_dfs(node_id=0, current_rule=[]):
    # feature[i] holds the feature to split on, for the internal
                                        node i.
    split_feature = inner_tree.feature[node_id]
    if split_feature != _tree.TREE_UNDEFINED: # internal node
      name = feature_names[split_feature]
      threshold = inner_tree.threshold[node_id]
      # left child
      left_rule = current_rule + ["({} <= {})".format(name,
                                          threshold)]
      tree_dfs(inner_tree.children_left[node_id], left_rule)
      # right child
      right_rule = current_rule + ["({} > {})".format(name,
                                          threshold)]
      tree_dfs(inner_tree.children_right[node_id], right_rule)
    else: # leaf
      dist = inner_tree.value[node_id][0]
      dist = dist/dist.sum()
      max_idx = dist.argmax()
      if len(current_rule) == 0:
        rule_string = "ALL"
      else:
        rule_string = " and ".join(current_rule)
      # register new rule to dictionary
      selected_class = classes[max_idx]
      class_probability = dist[max_idx]
      class_rules = class_rules_dict.get(selected_class, [])
      class_rules.append((rule_string, class_probability))
```

```
      class_rules_dict[selected_class] = class_rules

  tree_dfs() # start from root, node_id = 0
  return class_rules_dict

def cluster_report(data: pd.DataFrame, clusters, min_samples_leaf=
                                    50, pruning_level=0.01):
    # Create Model
    tree = DecisionTreeClassifier(min_samples_leaf=min_samples_leaf
                                        , ccp_alpha=pruning_level)
    tree.fit(data, clusters)

    # Generate Report
    feature_names = data.columns
    class_rule_dict = get_class_rules(tree, feature_names)

    report_class_list = []
    for class_name in class_rule_dict.keys():
        rule_list = class_rule_dict[class_name]
        combined_string = ""
        for rule in rule_list:
            combined_string += "[{}] {}\n\n".format(rule[1], rule[0
                                                ])
        report_class_list.append((class_name, combined_string))

    cluster_instance_df = pd.Series(clusters).value_counts().
                                        reset_index()
    cluster_instance_df.columns = ['class_name', 'instance_count']
    report_df = pd.DataFrame(report_class_list, columns=['
                                        class_name', 'rule_list'])
    report_df = pd.merge(cluster_instance_df, report_df, on='
                                        class_name', how='left')
    pretty_print(report_df.sort_values(by='class_name')[['
                                        class_name', 'instance_count'
                                        , 'rule_list']])
```

La funzione sopra descritta prende in input il dataset, le etichette risultanti dal

clustering e i parametri di *min_samples_leaf* e *pruning_level* che invece determinano la complessità dell'albero decisionale. Essa può essere richiamata come segue:

```
cluster_report(features, clustering_results , min_samples_leaf=25,
                                        pruning_level=0.05)
```

E' importante notare che più alti vengono scelti i parametri dell'albero e più generali saranno le regole estratte.

Applicando la funzione *cluster_report* ai risultati prodotti dagli esperimenti precedentemente descritti sono stati ottenuti i seguenti reports.

	class_name	instance_count	rule_list
1	0	192	[0.8209606986899564] (element0 <= 3.5) and (element4 > 2.5)
0	1	303	[0.8733333333333333] (element0 > 3.5) and (element4 > 3.5) and (element2 > 2.5)
2	2	170	[0.6739130434782609] (element0 <= 3.5) and (element4 <= 2.5) [0.7226277372262774] (element0 > 3.5) and (element4 <= 3.5)
3	3	156	[0.6349206349206349] (element0 > 3.5) and (element4 > 3.5) and (element2 <= 2.5)

Figura 5.13: K-Means con Distanza Euclidea

	class_name	instance_count	rule_list
2	0	151	[0.5333333333333333] (element0 <= 2.5) and (element1 <= 2.5) [0.8018867924528302] (element0 > 2.5) and (element4 > 3.5) and (element1 <= 3.5)
3	1	134	[0.7171052631578947] (element0 > 2.5) and (element4 <= 3.5)
1	2	216	[0.9057591623036649] (element0 <= 2.5) and (element1 > 2.5)
0	3	320	[0.9225589225589226] (element0 > 2.5) and (element4 > 3.5) and (element1 > 3.5)

Figura 5.14: K-Means con Cosine Similarity

	class_name	instance_count	rule_list
0	0	290	[0.7933579335793358] (element0 > 3.5) and (element4 > 3.5) and (element1 > 3.5)
1	1	229	[0.6791277258566978] (element0 <= 3.5)
2	2	154	[0.656934306569343] (element0 > 3.5) and (element4 <= 3.5)
3	3	148	[0.6413043478260869] (element0 > 3.5) and (element4 > 3.5) and (element1 <= 3.5)

Figura 5.15: Spectral Clustering con Cosine Similarity

	class_name	instance_count	rule_list
0	0	316	[0.8810810810810811] (element4 > 3.5) and (element0 <= 4.5) and (element8 > 2.5) [0.6891891891891891] (element4 > 3.5) and (element0 > 4.5) and (element1 <= 2.5) [0.5111111111111111] (element4 > 3.5) and (element0 > 4.5) and (element1 > 2.5) and (element2 <= 3.5)
2	1	167	[0.7681159420289855] (element4 <= 3.5) and (element8 <= 2.5) [0.5918367346938775] (element4 > 3.5) and (element0 <= 4.5) and (element8 <= 2.5)
1	2	240	[0.8727272727272727] (element4 > 3.5) and (element0 > 4.5) and (element1 > 2.5) and (element2 > 3.5)
3	3	98	[0.48044692737430167] (element4 <= 3.5) and (element8 > 2.5)

Figura 5.16: Agglomerative clustering con distanza euclidea e linkage Ward

Ciascun report è organizzato in tre colonne. Da sinistra verso destra troviamo l'attributo *class_name*, che indica l'etichetta dei clusters, seguito poi dall'indicatore *instance_count* che riporta il numero di oggetti contenuti all'interno del cluster corrispondente, ed, infine, sotto la colonna *rule_list* troviamo le regole estratte dalla funzione. Il numero tra parentesi quadre indica la proporzione di elementi della *class_name* che soddisfano la regola. Per esempio, $[0.821](element0 <= 3.5) and (element4 > 2.5)$, che troviamo come prima regola estratta per il cluster 0 nel caso del k-means con distanza euclidea, significa che tra tutte le sequenze che cominciano con un campo come prima inquadratura e presentano un'inquadratura che tende ad un piano nel quinto elemento della sequenza, l'82% di esse sono raggruppate nel cluster 0.

Dall'analisi dei report generati per i quattro esperimenti riportati emerge che gli elementi principali sulla base dei quali vengono definite le regole sono l'element0, l'elemento1, l'elemento2 e l'elemento4. Solo nel caso dell'Agglomerative clustering con distanza euclidea e linkage ward emerge in aggiunta ai precedenti anche l'element8. Emergono, inoltre, risultati interessanti ottenuti dalla cluster analysis effettuata con il K-Means con Cosine similarity. In particolare, andando ad analizzare le regole estratte per il cluster 2 e per il cluster 3 ci si rende conto di come l'algoritmo riesca a raggruppare il 91% circa delle sequenze che iniziano con campi lunghi che introducono poi dialoghi o scene di azione, nel cluster 2 (Fig 5.17), e il 92% di clip cinematografiche che iniziano con dialoghi che tendono poi a proseguire per tutta la prima metà della sequqenza, nel cluster 3 (Fig 5.18).

Mississippi Burning (1988)	[1, 5, 6, 6, 6, 4, 3, 6, 4, 6]
Moka (2017)	[2, 5, 5, 5, 5, 5, 5, 5, 6, 5]
Mortal Kombat Annihilation (1997)	[2, 6, 5, 6, 6, 6, 6, 4, 5, 6]
Mr Deeds (2002)	[2, 6, 4, 6, 6, 5, 1, 6, 3, 5]
My Best Friends Wedding (1997) - Scene 1	[0, 5, 6, 6, 5, 6, 5, 5, 6, 6]
My Girl (1991) - Scene 1	[2, 5, 4, 5, 5, 6, 5, 5, 6, 5]
My Girl (1991) - Scene 5	[1, 5, 5, 6, 5, 5, 3, 6, 4, 6]
My Spy (2020)	[2, 5, 5, 5, 3, 4, 5, 4, 5, 4]
Mystery Science Theater 3000 (1996) - Scene 1	[0, 6, 5, 6, 5, 5, 5, 4, 6, 5]
National Lampoons European Vacation (1985)	[1, 5, 5, 5, 3, 5, 5, 2, 5, 2]

Figura 5.17: Cluster 2

Ghostbusters - This Man Has No Dick (1984)	[5, 5, 6, 5, 6, 5, 6, 6, 6, 6]
Girls Trip (2017) - Grapefruiting	[5, 5, 5, 4, 5, 5, 5, 5, 2, 4]
Going the Distance - Take Me to Berlin (2010)	[5, 6, 5, 6, 6, 5, 6, 5, 5, 5]
Green Lantern - The Ring Chose You (2011)	[6, 6, 6, 6, 6, 6, 2, 7, 6, 7]
Gremlins - Billy Meets Gizmo (1984)	[5, 5, 5, 5, 6, 5, 3, 2, 5, 5]
Hancock (2008) - Drunk Heroism	[6, 5, 5, 5, 6, 2, 6, 0, 6, 5]
Hanna - Just Find Her (2011)	[5, 5, 5, 5, 6, 5, 5, 5, 5, 5]
Happy Texas - Harry Dumps Chappy (1999)	[5, 6, 5, 5, 7, 5, 6, 6, 6, 6]
Hereafter - Can I Ask You a Question? (2010)	[5, 5, 5, 6, 5, 5, 5, 5, 5, 5]
How High (2001) - The Liberty Bong	[5, 4, 6, 5, 6, 3, 5, 5, 5, 5]

Figura 5.18: Cluster 3

5.4 Clustering su sequenze di venti inquadrature

In maniera del tutto analoga a quanto visto per le sequenze di 10 inquadrature, si è ripetuta l'analisi sulle sequenze di 20 inquadrature. Anche in questo caso, scegliendo un numero di clusters pari a quattro in input a ciascun algoritmo, si è cercato di raggruppare i dati al fine di isolarne il genere. In tale caso, il dataframe dato in input agli algoritmi di clustering conta venti features, una per ogni inquadratura della sequenza, e 594 records, ossia pari al numero delle sequenze di 20 inquadrature ricavate dal dataset (vedi Tab. 5.3).

Analogalmente a quanto avventuo per il caso delle sequenze di lunghezza pari a dieci, gli algoritmi più performanti sono stati il K-Means, con entrambe le misure di distanza, lo Spectral Clustering, con Cosine similarity ed, infine l'Agglomerative clustering, utilizando il linkage Ward e la distanza euclidea come metrica di distanza. Di seguito vengono riportati i risultati ottenuti.

K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	34	59	27	29	149
Action	55	51	44	54	204
Comedy	53	74	23	33	183
Horror	20	21	6	11	58
TOT	**162**	**205**	**100**	**127**	594

Tabella 5.9: K-means con distanza euclidea

Cosine K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	52	31	37	29	149
Action	65	53	49	37	204
Comedy	65	36	57	25	183
Horror	19	15	15	9	58
TOT	**201**	**135**	**158**	**100**	594

Tabella 5.10: K-means con Cosine similarity

Spectral Clustering	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	28	50	40	31	149
Action	46	50	58	50	204
Comedy	30	53	56	44	183
Horror	5	21	18	14	58
TOT	**109**	**174**	**172**	**139**	594

Tabella 5.11: Spectral Clustering con Cosine similarity

Agglomerative Clustering	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	80	24	19	26	149
Action	91	41	29	43	204
Comedy	103	33	17	30	183
Horror	30	15	5	8	58
TOT	**304**	**113**	**70**	**107**	594

Tabella 5.12: Agglomerative Clustering con distanza euclidea e linkage Ward

In maniera del tutto simile a quanto avveniva per le sequenze di dieci inquadrature, anche per le sequenze di lunghezza pari a venti non è stato possibile ottenere,

mediante il clustering, raggruppamenti o clusters di oggetti omogenei rispetto al genere. Anche in questo caso, pur avendo un numero ridotto di records, i clusters tendono a replicare le proporzioni tra i vari generi presenti all'interno del dataset delle 594 sequenze di lunghezza venti, con una prevalenza all'interno di ciascun insieme di film di tipo Action o Comedy.

Valutazione dei clusters

Analizzati i risultati prodotti dagli algoritmi di clustering si è provveduto, sulla falsa riga dell'analisi effettuata sulle sequenze di dieci inquadrature, a compiere una valutazione prima quantitativa, mediante il Silhouette score, e poi qualitativa, studiando la distribuzione dei dati secondo etichette di cluster rispetto alle prime due Principal Components restituite dalla PCA.

In Tabella 5.13 si possono osservare i valori di Silhouette score per ciascun algoritmo utilizzato per l'analisi delle sequenze di venti inquadrature.

Algoritmo	**Silhouette score**
K-Means	0.065
Cosine K-Means	0.043
Spectral Clustering	0.061
Agglomerative Clustering	0.076

Tabella 5.13: Silhouette score medio per ciascun algoritmo

In questo caso i valori di Silhouette media per ciascun algoritmo risultano ampiamente inferiori a quelli ottenuti per le sequenze di lunghezza dieci. In particolare, nessun algoritmo ottiene un punteggio pari allo 0.1 ed il punteggio massimo lo si raggiunge utilizzando l'Agglomerative Clustering in unione con la distanza euclidea ed il linkage Ward. Subito dopo troviamo il K-Means con distanza euclidea e lo Spectral Clustering con Cosine similarity con valori poco al di sopra dello 0.06 . Come abbiamo visto in precedenza, però, la sola valutazione quantitativa in termini di coesione e separazione di un'analisi di clustering può risultare riduttiva. Per tale

motivo si è deciso, come nel caso delle sequenza di dieci inquadrature, di ripetere la valutazione qualitativa dei clusters tramite PCA prima di procedere con una più profonda caratterizzazione degli stessi estraendo le regole.

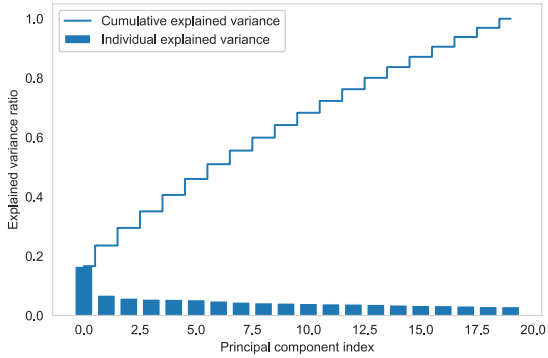

Figura 5.19: Varianza cumulativa esplicitata dalle Principal Components

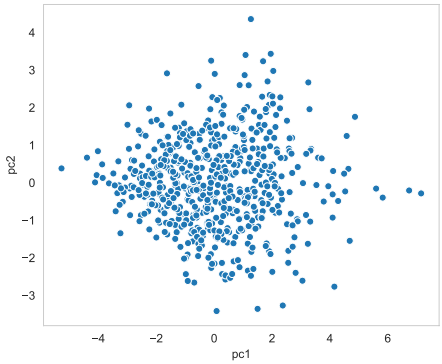

Figura 5.20: Distribuzione dei dati rispetto alle prime due PC

La PCA produce venti Principal Components combinando linearmente le venti features caratterizzanti le sequenze. Si nota che nel caso di sequenze di venti inquadrature le prime due principal components riescono ad esplicitare solo poco più del 20% della varianza all'interno dei dati (Figura 5.19). In Figura 5.20 troviamo poi la distribuzione dei dati rispetto alle prime due principal components. Come nell'analisi precedente, sono state confrontate le distribuzioni dei dati rispetto le prime due componenti fornite dalla PCA sia raggruppando per genere (grafico di sinistra) sia per etichette prodotte dai diversi algoritmi di clustering (grafico di destra). Di seguito sono riportati i confronti.

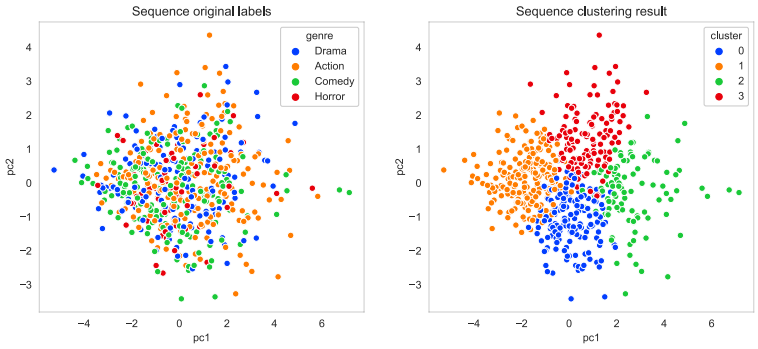

Figura 5.21: K-Means con Distanza Euclidea

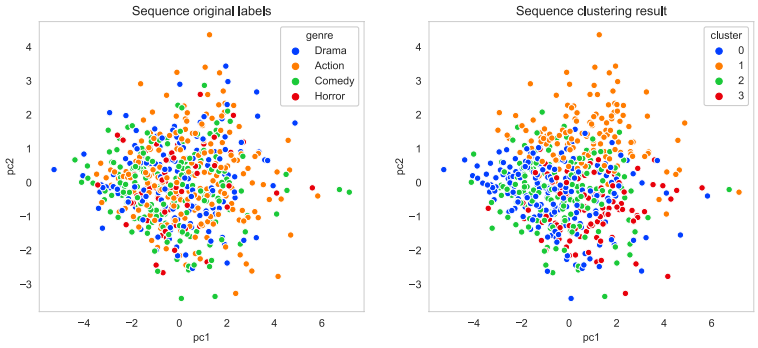

Figura 5.22: K-Means con Cosine Similarity

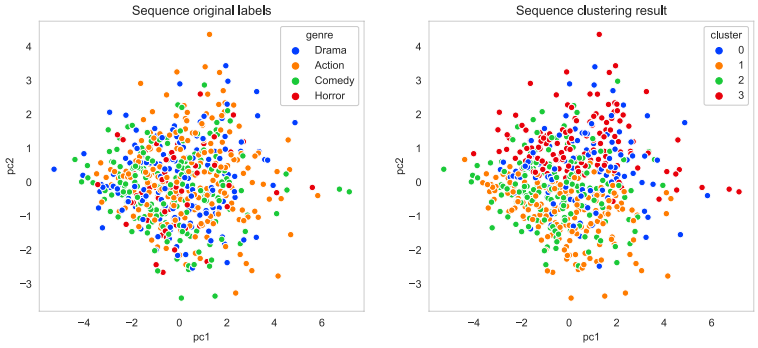

Figura 5.23: Spectral Clustering con Cosine Similarity

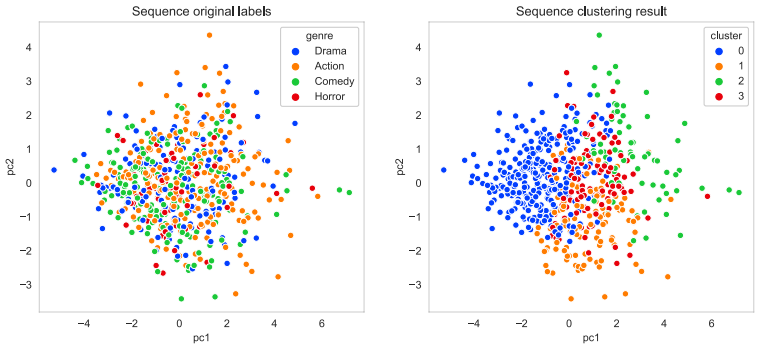

Figura 5.24: Agglomerative clustering con distanza euclidea e linkage Ward

Anche in questo caso, seppur si osservino clusters bene separati e definiti in modo netto come nel caso del K-Means con distanza euclidea 5.21, nessun algoritmo di clustering riesce a riprodurre la distribuzione dei dati etichettati secondo il genere rispetto alle prime due componenti principali definite dalla PCA.

Caratterizzazione dei clusters

Una volta valutati i clusters tramite Silhouette e comparative di grafici delle distribuzioni rispetto le componenti principali, si è proceduto alla caratterizzazione dei clustaers secondo le stesse modalità utilizzate per l'analisi delle sequenze di dieci inquadrature. Anche in questo caso, infatti è stata sfruttata la medesima funzione di report per estrarre regole circa le features prese in considerazione dai vari algoritmi nella formazione dei clusters. Vengono riportati i report forniti dalla funzione *cluster_report()* per ciascuna delle configurazioni utilizzate.

	class_name	instance_count	rule_list
1	0	162	[0.5934959349593496] (element19 > 2.5) and (element4 <= 3.5) [0.5098039215686274] (element19 > 2.5) and (element4 > 3.5) and (element13 > 3.5) and (element8 <= 3.5)
0	1	205	[0.7853107344632768] (element19 > 2.5) and (element4 > 3.5) and (element13 > 3.5) and (element8 > 3.5)
3	2	100	[0.32142857142857145] (element19 > 2.5) and (element4 > 3.5) and (element13 <= 3.5)
2	3	128	[0.51875] (element19 <= 2.5)

Figura 5.25: K-Means con Distanza Euclidea

	class_name	instance_count	rule_list
0	0	201	[0.7142857142857143] (element0 > 2.5) and (element4 > 2.5) and (element18 > 4.5)
2	1	136	[0.6636363636363637] (element0 > 2.5) and (element4 > 2.5) and (element18 <= 4.5)
1	2	158	[0.7988826815642458] (element0 <= 2.5)
3	3	100	[0.75] (element0 > 2.5) and (element4 <= 2.5)

Figura 5.26: K-Means con Cosine Similarity

	class_name	instance_count	rule_list
3	0	109	[0.5304347826086957] (element0 > 3.5) and (element3 <= 3.5)
0	1	174	[0.3484848484848485] (element0 <= 3.5) and (element7 <= 3.5) [0.5649717514124294] (element0 > 3.5) and (element3 > 3.5) and (element18 > 4.5)
1	2	172	[0.8618421052631579] (element0 <= 3.5) and (element7 > 3.5)
2	3	140	[0.6941176470588235] (element0 > 3.5) and (element3 > 3.5) and (element18 <= 4.5)

Figura 5.27: Spectral Clustering con Cosine Similarity

	class_name	instance_count	rule_list
0	0	304	[0.45054945054945056] (element10 > 2.5) and (element11 > 2.5) and (element2 <= 3.5) [0.8157894736842105] (element10 > 2.5) and (element11 > 2.5) and (element2 > 3.5)
1	1	113	[0.43956043956043955] (element10 > 2.5) and (element11 <= 2.5)
3	2	70	[0.6296296296296297] (element10 <= 2.5) and (element18 <= 1.5)
2	3	108	[0.4666666666666667] (element10 <= 2.5) and (element18 > 1.5)

Figura 5.28: Agglomerative clustering con distanza euclidea e linkage Ward

Dall'analisi dei report sopra riportati si scopre che nel caso di sequenze di lunghezza pari a venti inquadrature il numero di features rispetto alle quali vengono determinate le regole aumenta. Le più ricorrenti sono l'element0, l'element4, l'element7 e il suo successore element8, gli element10 e 11 ed infine le ultime due inquadrature della sequenza rappresentate dall'element18 e dall'element19. In generale, pur ottenendo buone percentuali rispetto a gruppi di elementi all'interno di singoli clusters, come ad esempio avviene nel caso del cluster 1 prodotto dal K-means con metrica di distanza euclidea (Fig. 5.25), i gruppi di sequenze isolati dai clusters presentano caratteristiche ricorrenti definite su features sparse e che, dunque, nulla ci dicono circa un eventuale stile di racconto comune nei film assocciati alle sequenze o, ancora, su possibili tecniche narrative espresse in termini di combinazioni di inquadrature contigue.

5.5 Clustering su sequenze e mean frame time

Avendo ormai constatato l'impossibilità di selezionare il genere delle sequenze mediante l'implementazione di metodologie di clustering operate sulle sole sequenze, sia di lunghezza dieci e sia di venti inquadrature, si è ripetuta l'analisi utilizzando oltre alle inquadrature che compongono ciascuna sequenza anche un'ulteriore feature, ossia il mean frame time. Come descritto nella sezioni relative alla creazione del dataset e all'analisi esplorativa, il mean frame time è stato utilizzato come misura di durata media delle scene, ognuna rappresentata da un'inquadratura, all'interno della sequenza.

Nella fattispecie, dunque, si è tentato di studiare se aggiungendo altre features diverse dalle inquadrature in ciascuna sequenza ma comunque caratterizzanti la sequenza stessa il risultato del clustering risultasse migliore e in grado, con numero di clusters scelto pari a quattro, di raggruppare le sequenze in clusters riproducendo la divisione per genere. A tale scopo l'analisi è stata eseguita su sequenze di dieci inquadrature con l'aggiunta, per l'appunto, del mean frame time, per un totale di 821 records e 11 features per ognuno di essi.

Tuttavia essendo il mean frame time definito su una scala diversa rispetto alle singole features rappresentative delle inquadrature e potendo raggiungere valori pari anche ad 80 secondi, è stato necessario standardizzare i dati in modo tale da non lasciare che gli alti valori assunti dal mean frame time influenzassero i risultati del clustering dominando le altre features in fase di creazione dei clusters tramite i vari algoritmi. Nella fattispecie, si è scelto di utilizzare la *z-score normalization* o *standardizzazione statistica* in modo da ottenere distribuzioni a media nulla e varianza unitaria calcolando i valori secondo la formula:

$$Z = \frac{X - \mu}{\sigma} \tag{5.3}$$

dove μ rappresenta la media della distribuzione e σ la deviazione standar. L'implementazione della standardizzazione dei dati è stata effettuata mediante la funzione

StandardScaler() contenuto nei metodi di preprocessing della libreria Scikit-learn[6]. Una volta preparati i dati per l'analisi sono stati effettuati diversi test mediante i diversi algoritmi di clustering imponendo il numero di clusters a quattro.

Di seguito vengono riportate le tabelle contenenti i risultati ottenuti attraverso l'utilizzo del K-Means, sia con distanza euclidea che con Cosine similarity, e dell'Agglomerative Clustering.

K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	93	62	55	22	232
Action	81	71	59	48	259
Comedy	97	60	59	37	253
Horror	25	27	14	11	77
TOT	**296**	**220**	**187**	**118**	821

Tabella 5.14: K-means con distanza euclidea

Cosine K-Means	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	73	50	56	53	232
Action	82	72	45	60	259
Comedy	82	62	60	49	253
Horror	23	14	17	23	77
TOT	**260**	**198**	**178**	**185**	821

Tabella 5.15: K-means con Cosine similarity

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

Agglomerative Clustering	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Totali
Drama	43	66	36	87	232
Action	71	59	38	91	259
Comedy	56	68	39	90	253
Horror	21	18	12	26	77
TOT	**191**	**211**	**125**	**294**	821

Tabella 5.16: Agglomerative Clustering con distanza euclidea e linkage Ward

Già da una prima lettura delle tabelle presentate si può notare come l'aggiunta del mean frame time come feature non produce significativi cambiamenti nella numerosità dei clusters e degli elementi di genere diverso all'interno di ogni cluster. Ancora una volta, infatti, le sequenze si distribuiscono all'interno dei clustrs riproducendo approssimativamente la distribuzione dei diversi generi caratterizzante il dataset. Fanno eccezione pochi clusters in cui vi è una leggera prevalenza dei film di genere Comedy.

Valutazione dei clusters

Sebbene non emergano sostanziali differenti paragonando i risultati con l'aggiunta del mean frame time con quelli registrati nel caso di clustering sulle sole sequenze di dieci inquadrature, elementi interessanti di discontinuità sono stati scorti nella valutazione del Silhouette score e della distribuzione rispetto le componenti principali.

Guardando al Silhouette Score (Tabella 5.17) si nota come per il K-means con Cosine similarity l'aggiunta del mean frame time e la standardizzazione delle features abbiano prodotto un significativo aumento della Silhouette, segno del fatto che i clusters individuati risultino separati e non sovrapposti. Al contrario, per gli esperimenti nei quali sono stati utilizzati il K-means con distanza euclidea oppure l'Agglomerative Clustering con distanza euclidea e linkage Ward, la Silhouette si è attestata su livelli lievemente negativi segno che alcuni elementi siano stati posizio-

nati in cluster sbagliati in quanto, per caratteristiche, risultano più simili ad oggetti presenti in altri clusters[7].

Algoritmo	**Silhouette score**
K-Means	-0.067
Cosine K-Means	0.666
Agglomerative Clustering	-0.030

Tabella 5.17: Silhouette score medio per ciascun algoritmo

A questo punto si è tentato di verificare se le variazioni del punteggio di Silhouette abbiano influito sulla visualizzazione della distribuzione dei clusters rispetto le prime due principal components ottenute tramite PCA.

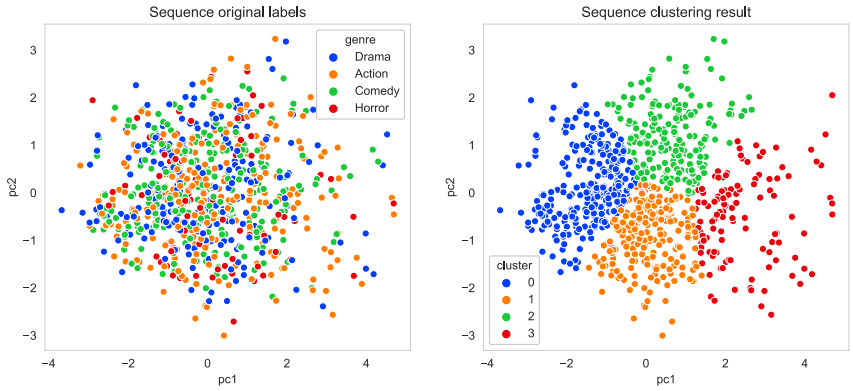

Figura 5.29: K-Means con Distanza Euclidea

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`

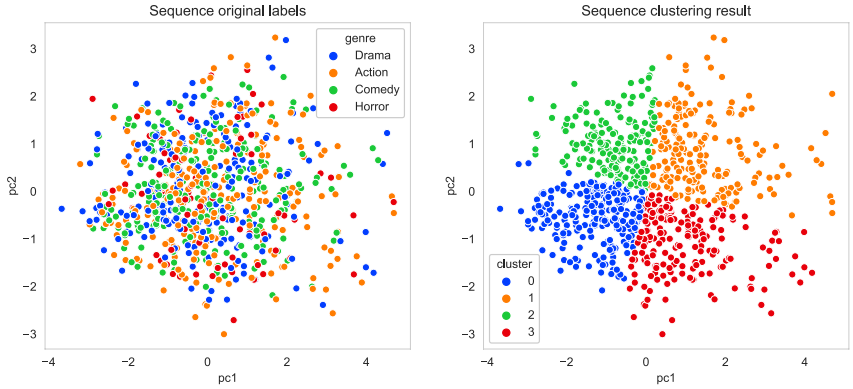

Figura 5.30: K-Means con Cosine Similarity

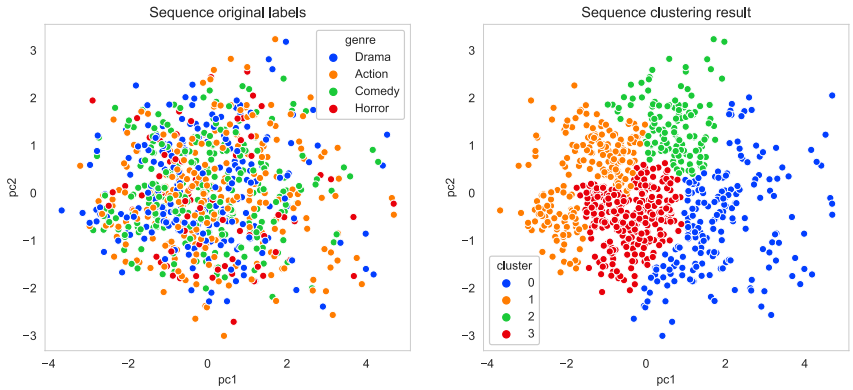

Figura 5.31: Agglomerative clustering con distanza euclidea e linkage Ward

I cluster individuati dal K-Means e dall'Agglomerative clustering in unione con la distanza euclidea sembrano sovrapporsi senza nette zone di demarcazione, in linea con le ipotesi fatte studiando i rispettivi valori di Silhouette. Al contrario, il valore molto alto del Silhouette Score ottenuto con il K-Means e la cosine similarity risulta avvalorato dal grafico di destra presente in Figura 5.30, dove si possono vedere i quattro cluster ben separati lungo due rette ideali incidenti nel punto (0,0).

Caratterizzazione dei clusters

L'ultimo step richiesto per completare l'analisi dei clusters formati sulla base delle sequenze di lunghezza dieci e del mean frame time è stata la caratterizzazione dei

clusters. In questo caso si era infatti interessati a valutare l'impatto del mean frame time sulle regole estratte dalla funzione di report circa le features maggiormente responsabili delle scelte degli algoritmi.

	class_name	instance_count	rule_list
0	0	296	[0.7189349112426036] (element9 > 3.5) and (element4 > 4.5) and (element7 > 2.5)
1	1	220	[0.5021645021645021] (element9 <= 3.5) [0.32653061224489793] (element9 > 3.5) and (element4 <= 4.5) and (element0 > 4.5) [0.6140350877192983] (element9 > 3.5) and (element4 > 4.5) and (element7 <= 2.5)
2	2	187	[0.8350515463917526] (element9 > 3.5) and (element4 <= 4.5) and (element0 <= 4.5)
3	3	118	NaN

Figura 5.32: K-Means con Distanza Euclidea

	class_name	instance_count	rule_list
0	0	260	[0.7254237288135593] (element0 > 3.5) and (element7 > 2.5) and (element2 > 3.5)
1	1	198	[0.7657657657657657] (element0 <= 3.5) and (element4 <= 3.5) [0.3953488372093023] (element0 > 3.5) and (element7 > 2.5) and (element2 <= 3.5)
3	2	178	[0.6238095238095238] (element0 <= 3.5) and (element4 > 3.5)
2	3	185	[0.7142857142857143] (element0 > 3.5) and (element7 <= 2.5)

Figura 5.33: K-Means con Cosine Similarity

	class_name	instance_count	rule_list
2	0	191	[0.5333333333333333] (element0 <= 3.5) and (element9 <= 3.5) [0.6319444444444444] (element0 > 3.5) and (element7 <= 3.5)
1	1	211	[0.44715447154471544] (element0 <= 3.5) and (element9 > 3.5)
3	2	125	NaN
0	3	294	[0.5702247191011236] (element0 > 3.5) and (element7 > 3.5)

Figura 5.34: Agglomerative clustering con distanza euclidea e linkage Ward

Sebbene l'introduzione del mean frame time e l'utilizzo in fase di preprocessing della standardizzazione delle features abbiano prodotto risultati inaspettati in termini di maggiore Silhouette score, come ad esempio nel caso del K-Means con cosine similarity, e di cluster meglio separati emersi dalla analisi delle distribuzioni rispetto alle prime due principal components, lo stesso non si può dire passando in rassegna le regole estratte. In tutti e tre i casi, infatti, le regole si concentrano su pochi

elementi sparsi tra i quali l'element0, l'element4, l'element7 ed, infine, l'element9. Ciò non ha permesso di scorgere eventuali pattern ricorrenti in termini di scelte di inquadrature nelle sequenze. Inoltre, nonostante i numerosi test effettuati scegliendo differenti combinazioni più o meno stringenti dei parametri *min_samples_leaf* e *pruning_level*, in nessun caso sono state riscontrate percentuali adeguate di elementi soddisfacenti le regole relative al corrispettivo cluster di appartenenza.

5.6 Cluster analysis con numero di clusters variabile

Gli esperimenti e i test mostrati fin'ora dimostrano come non sia possibile isolare il genere delle clip cinematografiche attraverso metodologie di clustering operate sulle sequenze in presenza o meno di misure temporali come, ad esempio, il mean frame time. Venuto meno l'obiettivo primario dello studio, ci si è chiesto se, mettendo da parte la nozione di genere, si potessero individure pattern ricorrenti all'interno delle sequenze rappresentativi di uno specifico stile narrativo o tecnico dettato proprio dall'utilizzo e dalla combinazione delle diverse tipologie di inquadrature in una stessa sequenza cinematografica.

In termini analitici, quello che si è cercato di fare è stato ripetere sia sulle sequenze di dieci inquadrature e sia su quelle di lunghezza venti la cluster analysis, questa volta però, non imponendo il numero di clusters a quattro, come nelle precedenti sperimentazioni, bensì andando alla ricerca di un numero ottimale in grado di restituire i migliori risultati in termini qualitativi.

Per questo tipo di analisi, si è andati, dunque, alla ricerca di un parametro k ottimale, rappresentante il numero di clusters, da specificare in input agli algoritmi di clustering. Gli algoritmi utilizzati per questo terzo tipo di analisi sono stati il K-Means, con entrambe le misure di distanza, e lo Spectral Clustering. Tale scelta non è stata casuale, anzi. Si è deciso di optare per queste due tipologie di algoritmi in quanto la stima del k ottimale nel caso del K-Means può essere effettuata tramite l'*Elbow method* o "metodo del gomito", che fornisce un'idea sul numero di clusters

da specificare in input all'algoritmo sulla base della riduzione marginale del SSE ottenuta aggiungende un cluster in più. Inoltre, dalle evidenze empiriche ottenute dai numerosi test effettuati con ciascun algoritmo, sono emerse buone performances dall'utilizzo dello Spectral Clustering con la cosine similarity. E' stata poi effettuata un'analisi dell'andamento della Silhouette al variare di k per monitorare i clusters in termini di coesione e separazione all'aumentare del parametro k.

Come detto, questo approccio di analisi è stato utilizzato sia sulle sequenze di dieci inquadrature e sia su quelle di lunghezza venti.

5.6.1 Sequenze di dieci inquadrature

Il primo passo nell'analisi delle sequenze è stato stimare il numero di clusters ottimale da utilizzare come input degli algoritmi sopra citati. Come accennato, si è scelto di procedere utilizzando il "Metodo del gomito" per il K-Means.

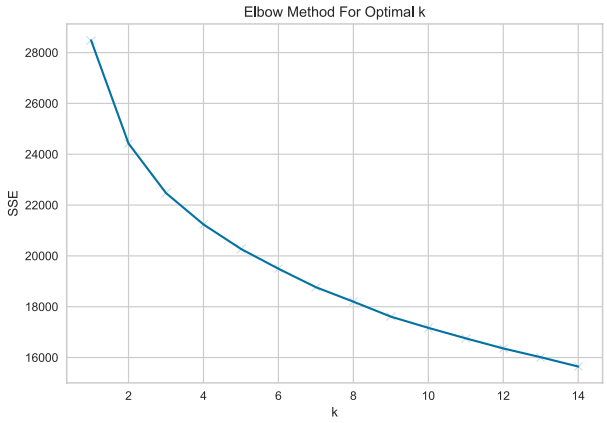

Figura 5.35: Elbow Method per la scelta del K ottimale

Il grafico in Fig.5.35 mostra l'andamento delll'SSE all'aumentare del numero di clusters k. In particolare, si tratta di un'euristica il cui obiettivo è mostrare la presenza di eventuali "gomiti" o "ginocchi" all'interno della funzione. Un "gomito" rappresenta un punto a ritorni marginali decrescenti, ossia tale per cui il costo di aggiungere un cluster in più non è "ricompensato" da pari benefici in termini di riduzione dell'SSE. Nel caso specifico della nostra analisi su sequenze di dieci

inquadrature, sembrano emergere dei possibili "gomiti" in corrispondenza di valori di k in un intorno di 3, ossia per valori pari a 2, 3 o 4. E' bene ricordare che il metodo del gomito rappresenta un'euristica e che, quindi, fornisce esclusivamente una prima indicazione per scegliere il numero di clusters.

In virtù di ciò, si è studiato in contemporanea l'andamento della Silhouette media al variare di k.

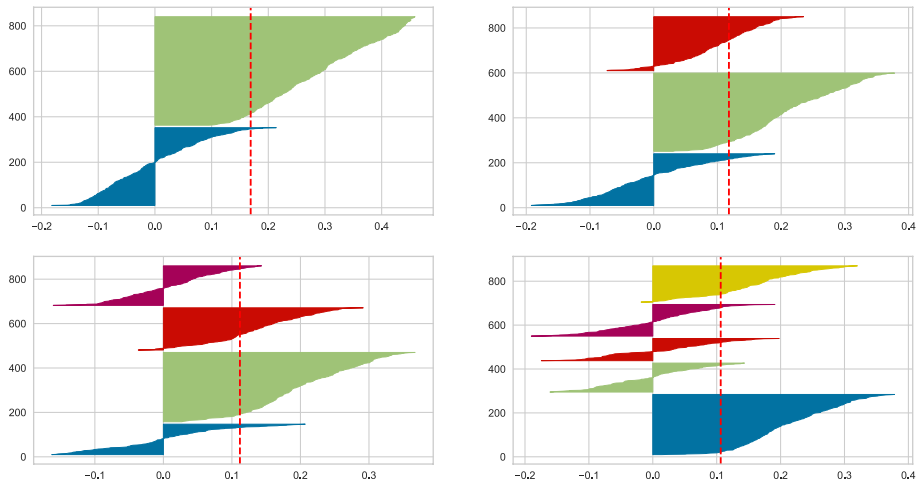

Figura 5.36: Andamento della Silhouette media per k in [2,5]

Il grafico in Fig. 5.36 mostra come la Silhouette media della partizione, indicata dalla linea tratteggiata in rosso, sia massima per valori di k bassi, mentre decresce all'aumentare delle fluttuazioni della misura dei clusters che si hanno per valori più alti di k. In tutti i casi però, la Silhouette media calcolata localmente per ciascun cluster supera i livelli di Silhouette media calcolati per l'intera partizione.

Dalle precedenti analisi è emerso, dunque, come valori di k bassi massimizzino la Silhouette e rappresentino punti a ritorni marginali decrescenti in termini di SSE. Per cercare di verificare le effettive permorfance ottenute per ciascun parametro si è proceduto anche in quetso caso alla caratterizzazione dei clusters mediante la funzione *cluster_report*.

In genereale, dall'analisi dei report si osserva che per k molto bassi, gli algoritmi si concentrino prevalentemente su pochi elementi per effettuare la partizione in clusters. Ciò, quindi, sfavorisce l'interpretazione qualitativa degli elementi all'interno dei cluster, impedendo di portare alla luce eventuali caratteristiche specifiche di una tecnica del montaggio o dello stile di racconto.

	class_name	instance_count	rule_list
0	0	524	[0.994] (element0 > 3.5)
1	1	297	[0.9158878504672897] (element0 <= 3.5)

Figura 5.37: Report K-Means con cosine similarity per k=2

	class_name	instance_count	rule_list
2	0	232	[0.6736111111111112] (element0 > 3.5) and (element7 <= 3.5)
1	1	243	[0.7476635514018691] (element0 <= 3.5)
0	2	346	[0.8117977528089888] (element0 > 3.5) and (element7 > 3.5)

Figura 5.38: Report K-Means con distanza euclidea per k=3

Nel caso di k=2, ad esempio, il K-means con cosine similarity effettua il partizionamento sulla base di un'unica features, l'element0 (Fig. 5.37). In modo del tutto analogo, lo stesso K-means ma questa volta con distanza euclidea e per k=3, estrae regole basate su due sole inquadrature, la prima e l'ottava, rappresentate dall'element0 e dall'element7 (Fig. 5.38).

AL contrario, operando la stessa analisi per numeri di clusters alti, ossia per k da 5 a 10, emergono clusters difficili da caratterizzare in modo proficuo mediante le regole. Tuttavia, ripetendo l'analisi anche per i valori di k appena riportati, sono stati ottenuri buoni risultati per k=6 con il K-Means in unione con la cosine similarity e con lo Spectral Clustering con medesima metrica di distanza.

	class_name	instance_count	rule_list
5	0	92	[0.86] (element0 > 2.5) and (element4 > 2.5) and (element9 > 2.5) and (element1 > 2.5) and (element8 <= 3.5)
0	1	216	[0.8981481481481481] (element0 > 2.5) and (element4 > 2.5) and (element9 > 2.5) and (element1 > 2.5) and (element8 > 3.5)
2	2	123	[0.8620689655172413] (element0 > 2.5) and (element4 > 2.5) and (element9 > 2.5) and (element1 <= 2.5)
1	3	185	[0.650375939849624] (element0 <= 2.5)
3	4	108	[0.6825396825396826] (element0 > 2.5) and (element4 <= 2.5)
4	5	97	[0.7904761904761904] (element0 > 2.5) and (element4 > 2.5) and (element9 <= 2.5)

Figura 5.39: Report K-Means con cosine similarity per k=6

	class_name	instance_count	rule_list
2	0	112	[0.42758620689655175] (element0 > 3.5) and (element1 <= 3.5)
0	1	249	[0.6762589928057554] (element0 > 3.5) and (element1 > 3.5) and (element2 > 3.5)
3	2	102	[0.3185840707964602] (element0 <= 3.5) and (element2 <= 3.5) [0.4545454545454453] (element0 > 3.5) and (element1 > 3.5) and (element2 <= 3.5)
5	3	91	NaN
1	4	175	[0.7067307692307693] (element0 <= 3.5) and (element2 > 3.5)
4	5	92	NaN

Figura 5.40: Report Spectral Clustering con cosine similarity per k=6

Guardando le liste delle regole riportate nei due report si evince che entrambi gli algoritmi riescano ad isolare con buone percentuali due diversi gruppi di sequenze.

In particolare, il report ottenuto per il K-means con cosine similarity (Fig. 5.39) mostra come nel cluster 1 l'algoritmo sia riuscito ad isolare la maggior parte (90% circa) delle clip che iniziano e terminano con dialoghi, mentre nel cluster 3 sono stati raggruppati il 67% circa delle sequenze che iniziano con inquadrature di tipo ambientale, che come si è visto nel capitolo sulla classificazione delle inquadrature cinematografiche, sono definite campi. La conferma a quanto detto la si è trovata andando, effettivamente, a scrutare gli elementi all'interno dei suddetti clusters.

Fantasy Island (2020) - Saving the Bully	[5, 5, 5, 1, 5, 5, 5, 2, 6, 5]
Fargo (1996) - Carl and the Parking Attendant	[6, 6, 5, 6, 5, 6, 6, 6, 6, 5]
Fatal Attraction - Not Going to Be Ignored (1987)	[4, 5, 4, 5, 5, 5, 5, 5, 5, 5]
First Knight (2017) - I Owe You a Kiss	[5, 5, 5, 7, 5, 7, 6, 7, 5, 7]
Flawless (1999) - Walt Tries to Sing	[5, 5, 5, 5, 4, 5, 6, 5, 6, 5]
Fletch Lives - Welcome Home, Fletch (1989)	[5, 2, 5, 2, 6, 1, 6, 4, 4, 5]
Flipped - You Hate Her (2010)	[6, 5, 6, 5, 6, 5, 6, 5, 6, 6]
Freaky (2020) - The Killer Returns	[5, 4, 2, 5, 5, 0, 5, 2, 5, 5]
Free Willy (1993) - Willy's Performance	[5, 5, 1, 5, 6, 2, 5, 6, 7, 5]
Freedom Writers - I Am Home (2007)	[6, 5, 3, 4, 3, 5, 6, 4, 4, 5]

Figura 5.41: Cluster 1 (K-Means con cosine similarity e K=6)

30 Minutes or Less (2011)	[1, 5, 5, 1, 5, 5, 5, 5, 6, 5]
7 From Etheria (2017)	[2, 6, 3, 5, 4, 2, 5, 6, 5, 5]
A Funny Thing Happened on the Way to the Forum (1966)	[2, 4, 3, 6, 6, 7, 6, 6, 6, 7]
A Good Day to Die (2013)	[0, 5, 0, 1, 5, 2, 6, 2, 6, 6]
A Million Ways to Die in the West (2014)	[0, 2, 6, 2, 2, 5, 3, 2, 5, 0]
A Royal Affair (2012)	[0, 6, 1, 6, 6, 6, 6, 6, 6, 1]
AVP Alien vs Predator (2004)	[0, 5, 5, 1, 5, 1, 5, 6, 2, 5]
Action Point (2018)	[0, 1, 5, 6, 2, 5, 2, 5, 6, 1]
American Graffiti (1973)	[1, 5, 4, 5, 5, 4, 4, 4, 4, 6]
American Pie 2 (2001)	[1, 5, 4, 4, 6, 6, 6, 5, 5, 5]

Figura 5.42: Cluster 3 (K-Means con cosine similarity e K=6)

In maniera analoga, il report in Fig. 5.40 relativo allo Spectral Clustering con cosine similarity, evidenzia come l'algoritmo riesca ad isolare nel cluster 1 il 70% delle clip che presentano dialoghi ad inizio sequenza, mentre nel cluster 4 vengono raccolte le sequenze con inquadrature di contesto, ossia campi, introduttive di dialoghi o di scene di azione. Come nel caso precedente, ciò è confermato dall'analisi empirica degli elementi contenuti nel dataset.

Patch Adams - Scene 5	[5, 5, 5, 5, 5, 5, 4, 5, 5, 5]
Patch Adams - Scene 9	[5, 6, 5, 6, 0, 6, 5, 0, 3, 3]
Police Academy 4 (1987)	[5, 4, 4, 4, 5, 5, 3, 5, 3, 2]
Pre-Game Brawl (1977)	[5, 5, 5, 1, 4, 3, 2, 1, 1, 1]
Project Almanac (2015)	[5, 6, 6, 6, 5, 6, 5, 1, 3, 2]
Promising Young Woman (2020) - S	[6, 6, 6, 6, 6, 6, 5, 6, 6, 6]
RIPD - That's a Deado (2013)	[6, 5, 6, 2, 0, 1, 5, 5, 5, 2]
Raising Cain (1992) - The Baby Thief	[6, 5, 6, 0, 0, 5, 5, 0, 2, 2]
Ray - Stealing From Ray (2004)	[6, 5, 5, 6, 5, 6, 6, 6, 5, 4]
Risen (2016) - Jesus Heals the Leper	[6, 6, 7, 6, 3, 5, 1, 6, 6, 2]
Robin Hood	[6, 6, 6, 6, 6, 5, 6, 5, 6, 6]

Figura 5.43: Cluster 1 (Spectral Clustering con cosine similarity e K=6)

Midnight Run (1988) - Scene 7	[0, 6, 5, 5, 4, 2, 5, 5, 5, 5]
Miss Bala (2019)	[0, 4, 6, 6, 6, 6, 2, 5, 6, 6]
Mississippi Burning (1988)	[1, 5, 6, 6, 6, 4, 3, 6, 4, 6]
Moka (2017)	[2, 5, 5, 5, 5, 5, 5, 5, 6, 5]
Moonstruck (1987)	[3, 5, 5, 5, 5, 5, 5, 6, 5, 5]
Mortal Kombat Annihilation (1997)	[2, 6, 5, 6, 6, 6, 6, 4, 5, 6]
My Best Friends Wedding (1997) - Scene	[0, 5, 6, 6, 5, 6, 5, 5, 6, 6]
My Girl (1991) - Scene 1	[2, 5, 4, 5, 5, 6, 5, 5, 6, 5]
My Girl (1991) - Scene 5	[1, 5, 5, 6, 5, 5, 3, 6, 4, 6]
My Spy (2020)	[2, 5, 5, 5, 3, 4, 5, 4, 5, 4]
Mystery Science Theater 3000 (1996) - S	[0, 6, 5, 6, 5, 5, 5, 4, 6, 5]

Figura 5.44: Cluster 4 (Spectral Clustering con cosine similarity e K=6)

Sebbene in tal caso siano stati ottenuti risultati interessanti, è bene sottolineare che nel caso di k=4, analizzato nel primo approccio di analisi descritto per le sequenze di dieci inquadrature, si erano ottenuti valori nettamenta più alti di elementi isolati rispetto a questi due macro gruppi di sequenze individuate sia dal K-means con cosine similarity e dallo spectral clustering con cosine similarity nel caso di k=6. Avevamo visto, infatti, che per k=4, il K-means con cosine similarity era in grado di isolare il 91% delle sequenze che iniziano con campi lunghi introduttivi di dia-

loghi o scene di azione nel cluster 2 e il 92% delle clip inizianti con dialoghi che si protraggono per tutta la prima metà della sequenza.

5.6.2 Sequenze di venti inquadrature

Analogalmente a quanto descritto nella sezione precedente, l'analisi è stata effettuata anche sulle sequenze di venti inquadrature. Anche in questo caso, il primo passo è consistito nel ricercare un valore di k ottimale per estrarre conoscenza utile dai clusters.

Dall'analisi della Silhouette (Fig. 5.45) emerge come anche per questo tipo di sequenze, la Silhouette media della partizione risulti massima per numeri di clusters molto bassi. In generale, poi, anche in corrispondenza del suo massimo la Silhouette non supera mai valori di 0.11 il che testimonia una apparente difficoltà da parte degli algoritmi nel formare clusters coesi e separati.

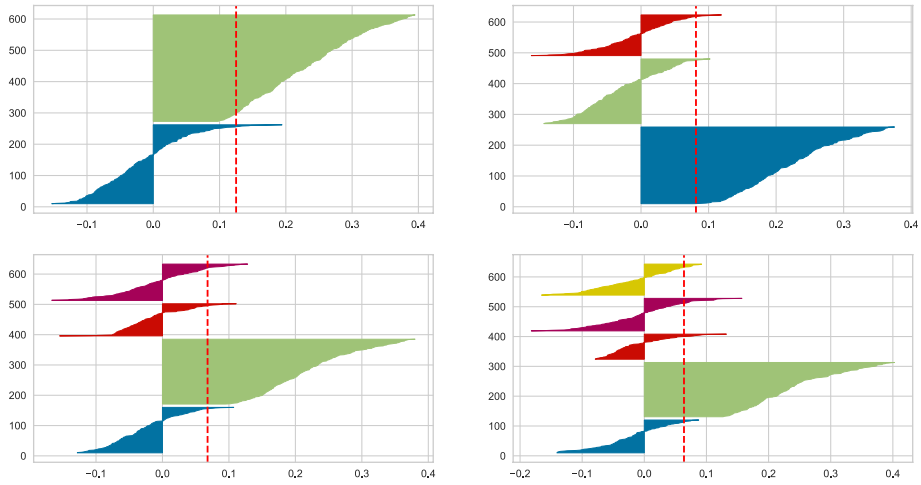

Figura 5.45: Andamento della Silhouette media per k in [2,5]

Tali difficoltà emergono anche e soprattutto in fase di caratterizzazione dei clusters. In generale, come per le sequenze di lunghezza dieci, valori di k bassi producono partizioni basate su regole che si incentrano pochi elementi sparsi delle sequenze. Al contrario, all'aumentare del numero di clusters crescono le difficoltà di indivi-

duare gruppi di sequenze con caratteristiche omogenee all'interno della partizione operata dagli algoritmi di clustering.

Nel caso delle sequenze di venti inquadrature, l'analisi delle regole estratte mediante la funzione di report testimonia come, nonostante le numerose combinazioni di algoritmi e parametri testate, risulti difficoltoso caratterizzare i clusters a causa dell'elevato numero di features prese in considerazione. Per non appesantire la trattazione di seguito vengono riportati solo alcuni dei report ottenuti in fase sperimentale per diversi valori di k.

	class_name	instance_count	rule_list
4	0	68	[0.5609756097560976] (element0 > 3.5) and (element4 > 3.5) and (element18 > 3.5) and (element1 <= 2.5)
1	1	141	[0.7604790419161677] (element0 <= 3.5) and (element6 > 2.5)
0	2	191	[0.4117647058823529] (element0 <= 3.5) and (element6 <= 2.5) [0.7814569536423841] (element0 > 3.5) and (element4 > 3.5) and (element18 > 3.5) and (element1 > 2.5)
2	3	109	[0.7307692307692307] (element0 > 3.5) and (element4 <= 3.5) and (element18 > 3.5)
3	4	85	[0.4] (element0 > 3.5) and (element4 <= 3.5) and (element18 <= 3.5) [0.5657894736842105] (element0 > 3.5) and (element4 > 3.5) and (element18 <= 3.5)

Figura 5.46: Report K-Means con cosine similarity per k=5

	class_name	instance_count	rule_list
2	0	112	[0.42758620689655175] (element0 > 3.5) and (element1 <= 3.5)
0	1	249	[0.6762589928057554] (element0 > 3.5) and (element1 > 3.5) and (element2 > 3.5)
3	2	102	[0.3185840707964602] (element0 <= 3.5) and (element2 <= 3.5) [0.45454545454545453] (element0 > 3.5) and (element1 > 3.5) and (element2 <= 3.5)
5	3	91	NaN
1	4	175	[0.7067307692307693] (element0 <= 3.5) and (element2 > 3.5)
4	5	92	NaN

Figura 5.47: Report K-Means con cosine similarity per k=6

	class_name	instance_count	rule_list
0	0	144	[0.6834532374100719] (element0 > 2.5) and (element8 > 3.5) and (element4 > 3.5) and (element10 > 2.5) and (element9 > 2.5)
2	1	82	[0.5185185185185185] (element0 <= 2.5) and (element4 <= 2.5) [0.4819277108433735] (element0 > 2.5) and (element8 > 3.5) and (element4 <= 3.5)
4	2	70	[0.46153846153846156] (element0 > 2.5) and (element8 <= 3.5)
3	3	73	[0.6585365853658537] (element0 > 2.5) and (element8 > 3.5) and (element4 > 3.5) and (element10 <= 2.5)
1	4	98	[0.656] (element0 <= 2.5) and (element4 > 2.5)
7	5	30	NaN
6	6	47	NaN
5	7	50	[0.5428571428571428] (element0 > 2.5) and (element8 > 3.5) and (element4 > 3.5) and (element10 > 2.5) and (element9 <= 2.5)

Figura 5.48: Report K-Means con cosine similarity per k=8

I report sopra citati evidenziano come l'elevato numero di features faccia si che, come avveniva per l'analisi dei cluster effettuata per k=4, le regole risultino essere incentrate su diverse features sparse e relative a gruppi di oggetti poco numerosi, non permettendo l'estrazione di conoscenza utile dai clusters di sequenze.

Capitolo 6

Conclusioni

L'obiettivo del seguente elaborato, come illustrato già nell'abstract, è stato quello di analizzare sequenze di inquadrature cinematografiche mediante tecniche e metodi tipici dell'Unsupervised Learning al fine di trovare possibili pattern significativi all'interno delle sequenze tali per cui si potesse operare una partizione delle stesse rispetto al genere attraverso l'utilizzo di differenti algoritmi di Clustering. Inoltre, si è cercato di caratterizzare i clusters risultanti da ogni analisi al fine di scorgere caratteristiche tipiche di una determinata tecnica di montaggio o di stile narravito determinate da specifiche combinazioni di particolari tipi di inquadrature. Le classi di inquadrature scelte sono le otto più diffuse all'interno dell'industria cinematografica. Esse sono: campo lungo, campo medio, figura intera, piano Americano, mezza figura, mezzo busto, primo piano e primissimo piano. I generi presi in considerazione per lo studio, invece, sono stati quattro: Drama, Action, Comedy ed Horror. Creato il dataset di sequenze cinematografiche, ciascuna di essa accompagnata dall'etichetta di classe del genere, sono stati implementati diversi approcci di cluster analysis. In particolare, avendo le sequenze lunghezza variabile, in primo luogo si è deciso di riportare il problema in uno spazio minore in cui tutte le sequenze analizzate avessero uguale lunghezza; ci si è quindi, ricondotti a sequenze di dieci e venti inquadrature sulle quali sono stati applicati i diversi algoritmi di clustering. Non riuscendo ad isolare il genere delle sequenze, si è scelto di effettuare l'analisi non più solo sulle sequenze ma introducendo tra le features analizzate anche la metrica

del Mean frame time. Constatato l'insuccesso anche di quest'ulteriore analisi si è proceduto mettendo da parte la nozione di genere e cercando di esplicitare eventuali relazioni significative tra le sequenze testando gli algoritmi di clustering su numeri di clusters variabili. Tale approccio ha portato buoni risultati sulle sequenze di dieci inquadrature, dove sia il K-means con cosine similarity e sia lo SPectral Clustering con medesima misura di distanza hanno isolato bne due macrogruppi di sequenze: quelle con presenza di dialoghi ad inizio sequenza e quelle aventi come prima inquadratura dei campi introduttivi di dialoghi o scene di azione. Tuttavia, i risultati migliori erano già stati ottenuti nel primo approccio di analisi con quattro clusters (idealmente uno per ogni genere), quando il K-means con cosine similarity era riuscito ad isolare il 91% del primo gruppo di sequenze ed il 92% circa del secondo.

Dai risultati dello studio si può constatare l'impossibilità di isolare il genere di sequenze cinematografiche utilizzando tecniche unsupervised sulle sole sole sequenze di inquadrature. Per far sì che un tale approccio possa produrre risultati soddisfancenti occorre introdurre all'interno dell'analisi nuove features maggiormente caratterizzanti il genere di una clip cinematografica. Tra tutte ad, esempio, si potrebbe pensare di utilizzare la color delle sequenze, in modo tale da disporre della mappa di colore predominante all'interno della sequenza e, quindi, considerare nell'analisi anche fattori diversi come le luci e le ombre nella scena che determinano una mappa di colore diversa a seconda dei casi. Tutto ciò, in associazione con la creazione di un dataset più numeroso e più equilibrato nella distribuzione dei diversi generi cinematografici, potrebbe condurre a risultati più interessanti.

Tuttavia, dallo studio emerge come l'analisi unsupervised delle sequenze possa essere un ottimo spunto per dare il via a ricerche incentrate sulle tecniche di montaggio e sugli stili di racconto mediante diverse combinazioni di inquadrature. Anche qui la creazione di un dataset maggiormente numeroso ed equilibrato, anche in relazione a scene di diverso tipo, potrebbe condurre a risultati favorevoli in futuri studi.

Bibliografia

[1] A. M. Turing, *Computing Machinery and Intelligence*, *Mind*, Volume LIX, Issue 236, Pages 433–460, October 1950

[2] Arthur L. Samuel, *Some studies in machine learning using the game of checkers*, *in IBM Journal of research and development*, 1959.

[3] J. Hurwitz, D. Kirsch, *Machine Learning for Dummies*, *John Wiley & Sons Inc.*, 2018

[4] Alex Smola, S.V.N. Vishwanathan, *Introduction to Machine Learning*, *Cambridge University Press*, 2008

[5] Stuart J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, *Prentice Hall*, 2010

[6] R. S. Sutton, A. G. Barto, *Reinforcement Learning, second edition: An Introduction*, *The MIT Press*

[7] R. Polikar, *Ensemble learning*, *Scholarpedia*, 4(1):2776, 2009

[8] Y. LeCun, Y. Bengio, G. Hinton, *Deep Learning*, *Nature*, Vol 521, 2015

[9] B. Vacchetti, T. Cerquitelli, R. Antonio, *Cinematographic Shot Classification through Deep Learning*

[10] Tan, Steinbach, Kumar, *Introduction to Data Mining*, *McGraw Hill*, 2006

[11] J. Han, M. Kamber, Jian Pei, *Data Mining: Concepts and Techniques*, 2012