Università degli Studi di Genova
Facoltà di Scienze Matematiche Fisiche e Naturali

Anno accademico 2005/06

Tesi di Laurea Specialistica in Informatica

Tecniche di Spectral Clustering: Analisi e Applicazioni

Candidato:
Nicola Rebagliati

Relatori:
Prof. A. Verri
Dott. L. Rosasco

Correlatore:
Prof. F. Masulli

Indice

alla mamma e al papà

Introduzione

Il Machine Learning[2] nacque negli anni sessanta quando F.Rosenblatt propose un modello che imitasse il funzionamento dei neuroni nel cervello, chiamandolo Percettrone Semplice([Ros58]). L'idea era già stata esplorata dai neurofisiologi, ma nessuno aveva ancora pensato a trasportarla sul computer.

In seguito furono sviluppati diversi algoritmi e parallelamente venne cominciato lo sviluppo delle fondamenta teoriche.

All'interno di questo sviluppo è racchiusa la teoria dell'Apprendimento Statistico di Vapnik e Chervonenkis [Vap99].

0.1 Come può una macchina imparare?

Osservando i caratteri di una scrittura ideografica, come ad esempio il cinese o il giapponese, si ha l'impressione iniziale che i caratteri siano molto simili tra loro e ci si chiede come sia possibile distinguerli l'uno dall'altro e utilizzarli per trasmettere informazioni. Di solito si trova un metodo personale per riconoscere e catalogare i vari simboli, fino a che ogni dettaglio viene utilizzato per fare distinzioni.

Immaginiamo di essere al posto del computer. Dal suo punto di vista la fotografia di un kanji (figura 1) non è altro che una sequenza di numeri (figura 2), ma allora come può imparare? Si può argomentare che anche per l'uomo i colori non sono altro che lunghezze d'onda differenti, ma non discuteremo il cognitivismo del cervello e il modo in cui esso apprende, piuttosto formalizzeremo matematicamente questi concetti in modo tale che da rispecchiare come la nostra intuizione chiama l''apprendere'.

Figura 1: Kanji giapponese

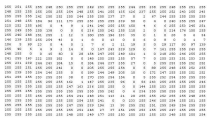

Figura 2: Ancora un kanji giapponese

[2]Apprendimento attraverso la macchina

Il problema del Learning

Dal momento che per il computer i dati non sono altri che sequenze di numeri, consideriamo lo spazio di input come un insieme $X \subseteq \mathbb{R}^n$, allo stesso modo consideriamo lo spazio degli output un insieme $Y \subseteq \mathbb{R}$. X e Y sono due insiemi di variabili aleatorie a cui possiamo associare una probabilità $p(x, y)$ fissata, ma sconosciuta.

L'insieme dei dati che abbiamo a disposizione per apprendere si chiama training set e viene denotato come un insieme di n coppie

$$(x_1, y_1), \ldots, (x_n, y_n)$$

estratte identicamente e indipendentemente (i.i.d.) a caso con la distribuzione di probabilità $p(x, y)$; di solito y viene chiamata *etichetta* di un dato.

Assumiamo che ci sia effettivamente un processo sottostante che correli gli x_i con y_i e che quindi $p(x_i, y_i)$ non sia un fenomeno aleatorio uniforme, in tale evenienza non ci sarebbe nulla da imparare, allo stesso modo in cui la probabilità dei numeri del Lotto è indipendente dalla ruota su cui vengono estratti.

A seconda della natura dell'output y possiamo dividere i problemi di Learning in tre categorie:

Classificazione y assume valori discreti in $\mathbb{Z}$ e qualifica l'appartenenza di un dato ad una classe. Si dice 'classe' un insieme da cui provengono i dati e si assume che questo insieme sia modellabile, quindi che vi sia un motivo effettivo per ritenere che un dato appartenga a questo insieme piuttosto che ad un altro. L'obiettivo dell'apprendimento è essere in grado di classificare correttamente nuovi dati.

Regressione y assume valori continui in $\mathbb{R}$, si vuole trovare la funzione che approssimi al meglio l'output sui dati.

Clustering Assumiamo che il training set sia composto da dati appartenenti a classi diverse, ma ignoriamo il valore dell'output y. Vogliamo stimare le classi di appartenenza del training set con considerazioni di carattere geometrico, guardando la distribuzione e la densità delle $x \in \mathbb{R}^n$.

Queste categorie sono aspetti differenti del problema di apprendere delle regole da un numero limitato di esempi, per questo condividono molti concetti legati alla statistica e possono beneficiare dei reciproci avanzamenti teorici.

Tecniche di Spectral Clustering

La strada per arrivare allo Spectral Clustering non è immediata, prima vedremo dei fondamenti di apprendimento da esempi, quindi approfondiremo il clustering ed infine entreremo nel vivo dell'argomento.

Il principio chiave delle tecniche di Spectral Clustering è considerare i dati come se fossero i vertici di un grafo e pesare le connessioni in base alla similarità tra due vertici. Questa interpretazione porta nel framework della *Teoria Spettrale dei Grafi*, una teoria che studia la caratterizzazione di un grafo attraverso gli autovalori di una matrice costruita a partire dal grafo: il Laplaciano; sorprendentemente questi autovalori possono rispondere ad un'ampia gamma di domande che potremmo porci sul grafo ed una di queste domande è proprio come partizionarlo nel miglior modo possibile.

L'ordine dei capitoli

Il primo capitolo spiega le basi dell'apprendimento da esempi; si parte dal caso in cui conosciamo completamente la distribuzione di probabilità dei dati ed in seguito si vede come modellare questa distribuzione, nel caso in cui sia sconosciuta, con dei modelli.

Successivamente entreremo nel vivo dell'Apprendimento Statistico; in esso vengono affrontate una serie di problematiche chiave del Machine Learning; i concetti che vengono formalizzati prevedono sostanzialmente la abilità di *controllare* la capacità di generalizzazione di una Learning Machine.

Questo controllo è di fondamentale importanza se si vuole che la Learning Machine sia in grado di commettere pochi errori su dati che non fanno parte del training set. Dopo la spiegazione di questi concetti vedremo come tornare alla pratica utilizzandoli per la creazione di algoritmi.

Nel secondo capitolo entreremo nel vivo del Clustering ed in particolare dello Spectral Clustering.

Dopo aver visto le basi teoriche degli approcci partizionali verrà spiegato l'algoritmo di K-Means, un punto di riferimento per questo tipo di approcci.

Dal K-Means salteremo allo Spectral Clustering. La sua esposizione incomincia con lo studio delle proprietà di un grafo attraverso il Laplaciano; vedremo come utilizzare questa matrice per partizionare il grafo e alcune varianti legate a questo utilizzo.

Nel terzo capitolo verranno esposte delle applicazioni dello Spectral Clustering su dati simulati e dati reali. I dati simulati hanno lo scopo di spostare la nostra attenzione su alcune proprietà del Laplaciano che emergono a partire da certe configurazioni dei dati.

Gli esperimenti reali si basano su un caso di telesorveglianza e sulla segmentazione di immagini. Nella telesorveglianza siamo interessati a raggruppare tra loro i movimenti che avvengono in un corridoio; verranno esposte tutte le fasi di questo esperimento, dalla codifica dei dati al controllo dei risultati.

Nel caso della segmentazione verrà proposto un approccio per affrontare un problema tecnico legato al Laplaciano: lo spazio di memoria allocato. Questo approccio consiste nel partizionare un sottocampionamento dell'immagine e quindi di utilizzare questi risultati per l'intera immagine.

Durante l'esposizione delle applicazioni dello Spectral Clustering, verrà rivolta particolare attenzione alle problematiche più importanti per utilizzare nella pratica questa tecnica e alle possibili risoluzioni.

Capitolo 1

L'Apprendimento da Esempi

Il Machine Learning è considerato una branca dell'Intelligenza Artificiale, una scienza che si propone di simulare l'intelligenza umana con l'uso del calcolatore. La qualità che lo contraddistingue da altre branche, come la programmazione logica, è il fatto di operare su dati che non sono simbolici, ma di solito provengono da un sottospazio di $\mathbb{R}^n$.

Le regole che si ricercano sono, in genere, delle funzioni di varia natura che suddividono questo spazio, in alcuni casi lo tassellano, stabilendo dei confini netti fra concetti diversi, come se ogni concetto fosse a sua volta un sottinsieme di $\mathbb{R}^n$.

La natura dei dati è quindi statistica e il training set è un rappresentante *aleatorio* del vero spazio di input.

1.1 Approccio statistico

Per capire la metodologia con cui si affrontano i problemi di apprendimento da esempi illustrerò alcuni approcci basilari della Classificazione e del Clustering. In entrambi i casi possediamo un training set, nel primo caso soffermiamo la nostra attenzione sulla distribuzione delle *etichette* nello spazio di Input, nel secondo caso invece guardiamo la distribuzione dei *dati* nello spazio di Input.

Unire i due approcci è un'evoluzione interessante nella ricerca del Machine Learning, sfruttare sia l'etichetta che la distribuzione geometrica dei dati. Alla base di questa unione c'é lo Spectral Clustering e conduce all'Apprendimento Semi-Supervisionato.

Ora vediamo in parallelo il caso in cui l'informazione che abbiamo a disposizione sia completa, nel caso della Classificazione e nel caso del Clustering.

1.1.1 Classificazione

Nei problemi di Classificazione le etichette dei dati y assumono valori discreti, per semplicità di esposizione trattiamo il caso in cui $y \in Y = \{-1, 1\}$.

Possiamo associare ad ogni esempio del training set una $p(x, y)$ che indica la probabilità di estrarre proprio quell'esempio dall'insieme $X \times Y$. Data soltanto

la x, la probabilità condizionata di y data x si indica con $p(y|x)$ e in seguito verrà chiamata probabilità a posteriori. La probabilità a posteriori indica quanto l'etichetta y è in accordo con x; più precisamente quantifica la probabilità che si possa assegnare la classe y conoscendo già x. Ci si potrebbe aspettare che questa probabilità valga uno per una classe e zero per ogni altra, cosicché ogni dato possa essere univocamente classificato[1]. Vi sono vari motivi per cui questo può non avvenire e la stessa x possa generare diversi y:

- Il processo sottostante è deterministico, ma è afflitto da un rumore sulla misura di y.
- Il processo sottostante è deterministico, ma le informazioni disponibili sono incomplete.
- Il processo sottostante non è deterministico.

Sotto queste condizioni, ci accorgiamo di non poter essere perfetti nell'assegnare un etichetta y ad un dato x ; questa è una conseguenza intrinseca del problema. Tutto quello che sappiamo è associare ad ogni x diverse classi i con probabilità $p(y_i|x)$. L'idea è che scegliendo la classe che massimizza questa probabilità operiamo la miglior scelta attuabile.

La Teoria di Bayes

L'idea della probabilità condizionata è di Thomas Bayes, un matematico inglese dell'Ottocento, che la introdusse per risolvere dei problemi di probabilità legati all'estrazione di palline colorate da urne: si voleva scoprire infatti, sapendo di aver estratto una pallina di un certo colore, la probabilità di averla estratta da una certa urna.

Se conosciamo a priori la probabilità di un evento $p(x)$ e la distribuzione di probabilità $p(x, y)$ allora, per definizione, la probabilità a posteriori vale:

$$p(y|x) = \frac{p(x, y)}{p(x)} \tag{1.1}$$

A questo punto dato un x la classificazione binaria sembra facile, basta vedere se $p(y = 1|x) > p(y = -1|x)$ o viceversa. Bisogna ricordarsi che così facendo non siamo perfetti, ma ottimizziamo quello che possiamo fare. La regola generale potrebbe essere:

$$f(x) := \begin{cases} 1 & \text{se } p(y = 1|x) > p(y = -1|x) \\ -1 & \text{se } p(y = -1|x) > p(y = 1|x) \end{cases} \tag{1.2}$$

L'equazione 1.2 si chiama regola di Bayes ed è intuitivamente la migliore soluzione possibile al problema di classificazione.

La regola di Bayes 1.2 può essere scritta sotto forma di un'unica funzione, se

$$g(x) = \log\left(\frac{p(y = 1|x)}{p(y = -1|x)}\right) \tag{1.3}$$

allora

[1] Ovviamente $\sum_{i=1}^{n} p(y_i|x) = 1$

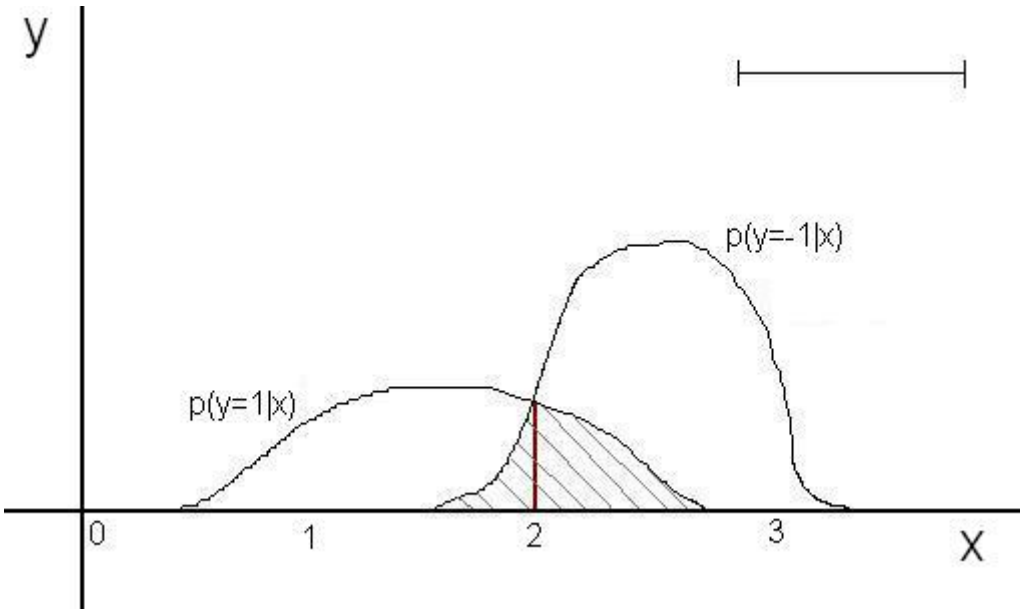

Figura 1.1: Un esempio di spazio di input con la distribuzione nota p(x,y)

$$f(x) = sign(g(x)) \tag{1.4}$$

I punti per cui $g(x) = 0$ costituiscono la superficie di decisione.

Questa superficie può essere visualizzata in figura 1.1, le due distribuzioni rappresentano rispettivamente $p(y = 1|x)$ e $p(y = -1|x)$ e per $x = 2$ sono equiprobabili.
L'area tratteggiata rappresenta la probabilità di errore, essa vale:

$$p(errore) = \int_{\{x|g(x)>0\}} p(x, y = -1)\, dx + \int_{\{x|g(x)<0\}} p(x, y = 1)\, dx \tag{1.5}$$

Ovviamente $p(errore)$ non può essere inferiore a 0.5, nel caso in cui valga esattamente 0.5 si può pensare che non ci sia nulla da imparare del processo sottostante alla probabilità.

1.1.2 Clustering

Approfondiremo la trattazione del Clustering nel capitolo 2, per il momento supponiamo di avere a disposizione un training set non etichettato, quindi semplicemente

$$(x_1), \ldots, (x_n) \tag{1.6}$$

supponiamo inoltre che ci siano k classi e che gli elementi siano estratti da queste classi in accordo alla probabilità di ogni classe $p(x_i|j)$ con $1 \leq j \leq k$.

Se $p(x)$ e $p(x, j)$ sono noti si può applicare anche nel Clustering la regola di Bayes vista per la Classificazione

$$f(x) := \begin{cases} 1 & \text{se } p(1|x) > p(2|x) \\ 2 & \text{se } p(2|x) > p(1|x) \end{cases} \tag{1.7}$$

si può riutilizzare la figura 1.1 sostituendo il significato dell'asse y, che non è più la distribuzione di probabilità dell'etichetta y, ma la densità del dato x.

I cluster trovati si intersecano e ovviamente la linea di interfaccia ci informa del risultato migliore che possiamo ottenere.

1.1.3 Due metodi di inferenza

Nei casi reali non si conosce esplicitamente la probabilità a posteriori, quindi non si può conoscere la regola di Bayes corretta, ma la si può stimare con dei metodi che inferiscano la distribuzione sconosciuta dei dati dal training set. Questo approccio è comune alle varie categorie di apprendimento: classificazione, regressione e clustering.

I metodi che vediamo in esempio sono semplici e nel contempo validi. Presentano però dei limiti che stimolano la ricerca di metodi più raffinati e precisi.

Metodi parametrici

L'approccio dei metodi parametrici consiste nell'inferire la distribuzione della probabilità a posteriori attraverso un modello parametrico,

Il modello più semplice che si possa utilizzare è una retta o, estendendo in più dimensioni, l'iperpiano; supponiamo che gli output y siano funzione degli input x nel seguente modo:

$$y = \sum_{i=1}^{n} a_i x_i + b \tag{1.8}$$

L'iperpiano separa tra loro classi differenti e i parametri da stimare sono $(a_1, \ldots, a_n, b)$. Questo è l'approccio seguito dal Percettrone Semplice e dei Minimi Quadrati. Ovviamente il modello parametrico non è legato a funzioni lineari, ma possono essere considerate, ad esempio, anche funzioni parametriche polinomiali o trigonometriche.

Quando si parla di approcci parametrici si arriva in modo naturale al concetto di *likelihood*. La probabilità a posteriori può essere riscritta, considerando l'insieme dei parametri ϑ, come $p(y|x, \vartheta)$. Dato il training set $S = (X_{ts}, Y_{ts}) = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ possiamo introdurre la *likelihood* in funzione di ϑ

$$L(\vartheta) := p(Y_{ts}|X_{ts}, \vartheta) = \prod_{i=1}^{n} p(y_i|x_i, \vartheta) \tag{1.9}$$

La *maximum likelihood* è la stima dei parametri ϑ che massimizzano la *likelihood* $L(\vartheta)$; intuitivamente possiamo pensare all'insieme di parametri che massimizzano l'accordo tra il modello e il training set.

Anche molti algoritmi di Clustering adottano l'approccio parametrico; in genere si suppone che la densità dei dati sia conforme ad una distribuzione gaussiana, quindi si assume che $L(\vartheta)$ sia una funzione differenziabile e si escogitano degli algoritmi che massimizzino $L(\vartheta)$, ad esempio che simulino una discesa a gradiente. L'algoritmo K-Means fa parte di questa categoria, ne approfondiremo i dettagli nella sezione 2.2.

Metodi non parametrici

I metodi non parametrici si adattano ai dati, senza imporre alcun modello sul processo che li ha generati. In sostanza non si interessano di stimare la densità della probabilità a posteriori.

I Primi Vicini sono uno dei metodi non parametrici più semplici. Dato un training set di n coppie

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \tag{1.10}$$

classifichiamo un nuovo punto allo stesso modo del punto del training set più vicino a lui, così facendo non calcoliamo esplicitamente una funzione classificatrice ma ci basiamo sulla *memoria* del training set.

Ovviamente con pochi dati questo metodo è instabile, per contro si comporta bene asintoticamente all'aumentare dei dati: se p^* è la probabilità di errore dei Primi Vicini e $p(errore)$ è come prima, abbiamo che

$$p(errore) \leq p^* \leq 2p(errore)(1 - p(errore)) \tag{1.11}$$

L'approccio dei Primi Vicini è suscettibile all'effetto dell'overfitting. Si dice che una funzione fa overfitting quando dà troppa importanza ai dati contenuti nel training set. Come visto prima i dati nel training set non sono perfetti, quindi è naturale cercare un metodo che li usi per generalizzare piuttosto che provi a non compiere errori su essi. Si può mostrare che il comportamento dei Primi Vicini migliora significativamente nel caso in cui si considerino k Primi Vicini. Il motivo è che attraverso k siamo in grado di regolare la capacità di generalizzazione evitando l'overfitting.

La Maledizione della Dimensionalità

Se apparentemente i k Primi Vicini sembrano un algoritmo ideale, non devono fare assunzioni a priori e possono regolare la loro capacità di generalizzazione, in realtà si scontrano duramente contro la Maledizione della Dimensionalità[2].

Questo termine è stato coniato da Richard Bellman [Bel61] e si riferisce al fatto che un'eccessiva dimensionalità dei dati può portare ad una considerevole diminuzione delle prestazioni di una learning machine.

La causa prima risiede nel numero dei dati che si devono avere per 'ricoprire' lo spazio degli esempi, vediamo un esempio esplicativo: supponiamo che questo spazio sia semplicemente un quadrato di lato dieci, e che l'unità di misura sia proprio uno, se vogliamo ricoprire questo spazio in modo da avere un punto per ogni quadratino di lato uno, abbiamo bisogno di cento dati, non certo un gran numero. Ma passiamo ad un quadrato in uno spazio a dieci dimensioni, ora il numero dei dati che ci servono è dieci milioni!

Dieci dimensioni non sono tante nel Machine Learning, basti pensare che se di un'immagine bitmap 100×100 considerassimo ogni pixel come una feature, otterremmo un vettore di diecimila elementi, ed uno spazio in cui per ricoprire allo stesso modo di prima un quadrato di lato dieci ci vorrebbero 10^{10000} elementi.

Una seconda fonte di problemi sono le distanze; all'aumentare della dimensionalità, le distanze tra i punti diventano sempre più ambigue, portando meno

[2]The Curse of Dimensionality

informazione; pensiamo ad un quadrato a dieci dimensioni e vediamo come i dati si distribuiscano sul 'guscio', se infatti consideriamo un quadrato di lato otto che ne costituisca il nucleo e calcoliamo il rapporto tra il numero di dati contenuti nel nucleo e il numero di dati totale otteniamo poco meno dell'undici percento, quindi il guscio raccoglie quasi il novanta percento dei dati!

Ovviamente anche in questo caso c'è un peggioramento all'aumentare delle dimensioni. La curse of dimensionality è un problema intrinsecamente non risolvibile, ma alcune tecniche come le SVM, Support Vector Machine [Vap99], sembrano non soffrirne, al contrario di molte altre tecniche come i k-nearest-neighbor [Das91].

Affrontare i Problemi

La semplicità di molti metodi, come i Minimi Quadrati, il Percettrone Semplice e i Primi Vicini, è anche la loro debolezza, essi non sono in grado di affrontare problematiche come il *controllo della capacità di generalizzazione* o la robustezza all'aumentare della dimensionalità dei dati.

La teoria dell'Apprendimento Statistico nasce per allargare la nostra visione sul problema del Learning e creare dei metodi che affrontino le difficoltà con successo.

I metodi che abbiamo schematizzato presentano dei grossi limiti, il motivo fondamentale è che apprendere è a tutti gli effetti un compito difficile. L'approccio che si segue è quindi quello di approfondire le fondamenta teoriche, estendendo con nuovi assiomi la precedente teoria.

1.2 Gli ingredienti per una Learning Machine

Prima di proseguire la strada verso la teoria dell'Apprendimento Statistico, soffermiamoci sulla ricetta di una Learning Machine. Così facendo non solo vediamo un po' di terminologia, ma contestualizziamo nella pratica alcuni concetti che in seguito sono utilizzati in maniera astratta.

Il Prodotto Informatico

Il prodotto informatico che si ottiene dopo una formalizzazione teorica dell'apprendimento si chiama Learning Machine[3], nella pratica è un algoritmo[4] che implementa il processo dell'apprendimento.

Il fine di questo processo è ottenere e implementare una funzione in grado di classificare nuovi esempi che si potrebbero proporre.

Il training set è formato da esempi che utilizziamo per addestrare la Learning Machine. Possibilmente questi esempi sono diversi da quelli che in futuro verranno classificati dalla Learning Machine.

Dato il training set, la Learning Machine deve cominciare a classificarlo usando delle funzioni, l'insieme di funzioni da cui si sceglie si chiama spazio delle ipotesi H.

[3] Macchina ad Apprendimento

[4] Quando parlo di algoritmi suppongo, ovviamente, che siano indipendenti dal linguaggio di programmazione, infatti i meccanismi necessari all'implementazione sono basilari e presenti nella maggior parte dei linguaggi di programmazione non banali

Per definire H si può usare una conoscenza a priori sul problema, oppure si possono usare degli H generici, come i Reproducing Kernel Hilbert Space ([DVEA]). Un dettaglio forse più implementativo, ma di fondamentale importanza, è il modo in cui viene esplorato H, che ovviamente dipende da H stesso.

Affinché sia in grado di apprendere nella maniera corretta si desidera che la Learning Machine sia consistente. La consistenza è la capacità di comportarsi sempre meglio nel caso in cui il numero di esempi a disposizione aumenti quantitativamente.

Nella realtà è difficile disporre di molti esempi[5], anzi per problemi pratici dobbiamo verificare che la Learning Machine si comporti bene nel caso disponiamo di pochi esempi.

Infine è indispensabile specificare come pesare gli errori commessi. Se ad esempio non vogliamo consentirli in maniera assoluta, possiamo valutarli infinito, oppure possiamo pesare allo stesso modo ogni errore, per quanto grande esso possa essere, attribuendo ad ognuno lo stesso valore costante.

In generale la teoria dell'Apprendimento Statistico è nata per risolvere questi problemi e sviluppare metodi per costruire le Learning Machine, essa include:

- Descrizione delle condizioni necessarie e sufficienti per la consistenza di una Learning Machine.
- Limiti che descrivano la capacità di generalizzazione di una Learning Machine.
- Addestramento con pochi esempi.
- Metodi di implementazione.

Gli ingredienti utilizzati per questi fini sono:

1. Il training set
2. Un insieme di funzioni H, chiamato spazio delle ipotesi
3. Un procedimento per ricercare la funzione da H
4. Una funzione di Loss che valuta la gravità di un errore

Il procedimento del punto 3 deve assicurare che la Learning Machine sia in grado di generalizzare. Se la capacità di generalizzazione è buona, la Learning Machine si comporterà bene su nuovi dati, non presenti nel training set.

Nei paragrafi seguenti vedremo un primo approccio in cui si dà la massima importanza nel classificare correttamente il training set, l'*empirical risk minimization*, in seguito vedremo un'evoluzione di questo approccio, in cui si minimizza l'errore compiuto sul training set e contemporaneamente si controlla la complessità della funzione classificatrice. Visto che queste due minimizzazioni sono in contrapposizione dovremo accettare un compromesso delle due.

[5]Si può dire che gli esempi sono 'molti' o 'pochi' a seconda del numero di dimensioni in cui essi vivono

1.3 La via dell'Apprendimento Statistico

L'Apprendimento Statistico si occupa di formalizzare molti concetti del Machine Learning per guidare la creazione di nuove Learning Machine. Per descrivere le linee principali di questa teoria prendiamo in considerazione l'apprendimento supervisionato([eSS03]), in cui i dati sono etichettati. I concetti generali che si possono estrapolare sono utilizzabili in altri contesti come lo Spectral Clustering.

1.3.1 La Minimizzazione del Rischio

Calandoci nel mondo reale ci aspettiamo che la regola di Bayes su dei dati sia applicata da un supervisore, che quindi è in grado di etichettare correttamente i dati.

Biologi, esperti di finanza o fenomeni atmosferici potrebbero essere dei supervisori possibili per il nostro problema, ma qualunque sia la natura del supervisore noi non siamo in grado di ricavare esplicitamente il processo che genera la risposta, possiamo solo approssimarlo con una Learning Machine.

La funzione di Loss ([RLA04]) è una funzione $L : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ che stima il peso dell'errore che una Learning Machine compie su un dato in input. Considerato l'intero spazio input siamo interessati a minimizzare il peso dell'errore totale, che chiamiamo rischio.

Formalmente il rischio integra il valore della Loss function lungo tutto il dominio dell'input, pesandolo a sua volta per la probabilità di quell'input, $I[f]$ si dice rischio atteso di una funzione f:

$$I[f] = \int_{XxY} L(y, f(y))\, dF(x, y) \tag{1.12}$$

Dove $f : \mathbb{R}^n \to \mathbb{R}$ è la funzione calcolata dalla Learning Machine, mentre $F(x, y)$ è la distribuzione di probabilità dei dati in input con le loro etichette.

Si può argomentare che è impossibile minimizzare direttamente il rischio atteso dal momento che non si conosce la soluzione, e questo

$$\operatorname*{argmin}_{f\in H} I[f] \tag{1.13}$$

è la soluzione ottimale, diventerà il nostro punto di riferimento. L'equazione 1.13 è in un certo senso l'evoluzione dell'equazione 1.5, in quanto rende esplicito il fatto che un errore possa avere pesi differenti e modella la regola di Bayes con una funzione sullo spazio degli input.

1.3.2 Il Principio della Minimizzazione del Rischio Empirico

Il rischio non può essere minimizzato esplicitamente dal momento che non conosciamo la funzione soluzione. Quindi siamo interessati a cercare delle alternative che lo approssimino, una prima idea che può venire in mente è il riutilizzo dei dati del training set.

Dopo che questi dati sono stati utilizzati per addestrare la Learning Machine si possono riutilizzare per vedere come si comporta su di essi la stessa Learning Machine. Questo comportamento potrebbe essere perfetto o meno, a seconda dei dati e dell'insieme di funzioni che la Learning Machine ha a disposizione.

Se, ad esempio, lo spazio delle ipotesi consiste nell'insieme delle rette sul piano, ma i dati del training set non sono linearmente separabili, dovremmo rassegnarci a compiere sempre degli errori. D'altro canto esistono funzioni che potrebbero comportarsi perfettamente su qualunque insieme di dati. Definiamo come errore empirico $I_{emp}[f]$ di una funzione f come il rischio atteso limitato al training set:

$$I_{emp}[f] = \frac{1}{l}\sum_{i=1}^{l} L(y_i, f(x_i)) \tag{1.14}$$

dato un training set

$$(x_1, y_1), \ldots, (x_l, y_l) \tag{1.15}$$

Data questa definizione possiamo impostare la Learning Machine affinché lo minimizzi:

$$\hat{f} = \underset{f \in H}{\operatorname{argmin}}\, I_{emp}[f] \tag{1.16}$$

Come ho già accennato prima si richiede che questo principio sia coerente e consistente. Se $I[f]$ è il rischio atteso di una funzione generica f, la coerenza è data da

$$\lim_{l \to \infty} I_{emp}[f] = I[f] \tag{1.17}$$

La coerenza ci assicura che all'aumentare dei dati nel training set l'errore empirico si avvicini effettivamente al rischio atteso. La quantità $I_{emp}[f]$ è ovviamente una variabile aleatoria dal momento che dipende dal training set, estratto a caso dall'input.

Il limite 1.17 può essere interpretato nei termini della convergenza della probabilità delle variabili aleatorie:

$$\forall \epsilon > 0, \lim_{l \to \infty} Prob(\|I_{emp}[f] - I[f]\| \geq \epsilon) = 0 \tag{1.18}$$

Ma l'equazione 1.18 è verificata semplicemente per la legge dei grandi numeri

$$\forall \epsilon > 0, \lim_{l \to \infty} Prob(\|\frac{1}{l}\sum_{i=1}^{l} Q_i - EQ\| \geq \epsilon) = 0$$

Dove Q è una variabile aleatoria generica.

La coerenza è soddisfatta e ci si può ora occupare della consistenza. Una Learning Machine è consistente se all'aumentare dei dati impara 'meglio'. Formalmente desideriamo che il rischio della funzione appresa dal training set tenda al rischio della f_H, la migliore funzione che potremmo scegliere dall'insieme H. Sia:

$$f_H = \underset{f \in H}{\operatorname{argmin}}\, I[f] \tag{1.19}$$

allora la consistenza può essere formulata come:

$$\lim_{l \to \infty} I[\hat{f}] - I[f_H] = 0 \tag{1.20}$$

Si vede che la formula 1.20 è sempre maggiore di zero, in quanto il rischio $I[f_H]$ è il minimo per definizione. Con poche disuguaglianze possiamo anche trovare un upper bound, limitando la formula 1.20 fra:

$$0 \leq I[\hat{f}] - I[f_H] \leq 2 \sup_{f \in H} |I_{emp}[f] - I[f]| \tag{1.21}$$

Se l'upper bound tende a zero all'aumentare dei dati, abbiamo assicurato la consistenza. Ma questo richiede che il rischio atteso tenda al rischio empirico uniformemente nello spazio delle ipotesi H.

Vedremo come questo sia possibile dopo aver definito la dimensione di Vapnik e Chervonenkis.

1.3.3 La VC-dimension e il controllo dello spazio delle ipotesi

La VC-dimension[6] è un concetto introdotto negli anni '60 da due matematici russi e costituisce uno dei punti di riferimento fondamentali nel Machine Learning.

Per semplicità ci limiteremo al caso di funzioni indicatore $f : \mathbb{R}^n \to \{0, 1\}$, quindi classificatori binari.

La VC dimension di un insieme di funzioni indicatore([Vap99])

La VC-dimension di un insieme H di funzioni indicatore è il massimo numero h di vettori $z_1, \ldots, z_h$ che possono essere separati in due classi in tutti i 2^h modi possibili usando funzioni dell'insieme[7].

Se un qualunque numero di vettori può essere separato si dice che la dimensione di VC è infinito.

La VC-dimension esprime la potenza classificatrice dello spazio H di funzioni scelto per la Learning Machine. Chiaramente questa potenza è legata alla complessità che una funzione può assumere.

Consideriamo ad esempio che lo spazio delle ipotesi sia costituito da rette e che i dati in input vivano sul piano. In questo caso la VC-dimension dello spazio delle rette è tre.

Per visualizzare questo risultato si consideri la figura 1.2, vi sono tre esempi e per loro vi sono otto possibili classificazioni[8] che possono essere ottenute da una retta[9]. Nella figura 1.3 ci sono quattro punti, ma in questo caso la divisione raffigurata non si può ottenere con una retta. La VC-dimension rimane quindi tre.

Nel caso in cui la VC-dimension sia infinita ci si aspetta che l'errore empirico nel caso migliore sia sempre zero, questo perché siamo in grado di classificare in qualunque modo questi punti. Per quanto questa prospettiva sia attraente essa non ci garantisce la consistenza, non ci assicura che al tendere dei dati all'infinito il rischio atteso della funzione calcolata sui dati tenda al rischio atteso della funzione migliore.

[6] La dimensione di Vapnik e Chervonenkis

[7] Gli in insiemi risultano due a seconda se la funzione indicatore prende valore zero o valore uno

[8] In realtà sono sufficienti quattro classificazioni, nel caso in cui siamo interessati semplicemente a dividerli tra loro.

[9] Non consideriamo il caso degenere di tre punti allineati, nel caso i punti vivano in $\mathbb{R}^2$ la probabilità di questo evento è zero.

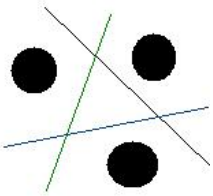

Figura 1.2: I tre esempi possono essere divisi in tutti gli otto modi possibili

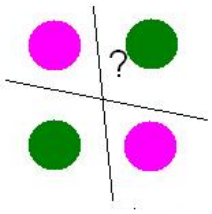

Figura 1.3: Un caso in cui non è ottenibile da un piano

Per visualizzare questo fatto vediamo come in figura 1.4 l'errore empirico della soluzione sia zero. Si potrebbe argomentare, osservando il risultato, che la funzione approssimata sia un po' 'forzata', ma i dubbi svaniscono quando si vede la vera distribuzione dei dati e la semplicità della funzione migliore tratteggiata in blu in figura 1.5.

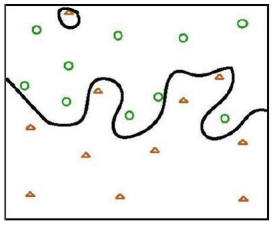

Figura 1.4: Una soluzione con errore empirico zero

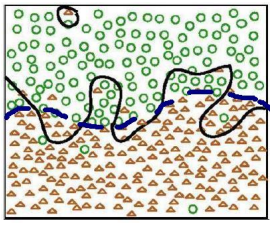

Figura 1.5: La stessa soluzione con un rischio atteso non ottimale

Vapnik e Chervonenkis sono stati in grado di trovare un limite superiore alla differenza tra il rischio empirico di una funzione e il suo rischio atteso, questo limite vale con probabilità $(1 - \eta)$:

$$\sup_{f \in H} |I_{emp}[f] - I[f]| \leq \sqrt{\frac{h \log \frac{2el}{h} - \log \frac{\eta}{4}}{l}} \tag{1.22}$$

Dove:

- h è la VC-dimension dello spazio delle ipotesi
- l è la cardinalità del training set
- η è la probabilità che la differenza tra il rischio empirico e il rischio atteso sia minore di ϵ, con ϵ piccolo.

Il secondo membro della disequazione 1.22 indica che all'aumentare del numero di dati l, o al diminuire della VC-dimension h la funzione trovata empiricamente tende a comportarsi come la funzione migliore che possiamo trovare in H.

Viceversa al diminuire dei dati o all'aumentare di h la garanzia di consistenza diminuisce.

La VC-dimension sposta la nostra attenzione sul *controllo* dello spazio delle ipotesi. Spesso è possibile annidare questo spazio in sottospazi la cui VC-dimension è minore dello spazio che lo contiene.

$$H_1 \subseteq H_2 \subseteq \cdots \subseteq H_m \subseteq \ldots \tag{1.23}$$

L'unione di questi sottospazi potrebbe avere VC-dimension infinita, ma per ognuno di essi potrebbe essere limitata, consentendo una ricerca che garantisca la consistenza.

Uno degli spazi delle ipotesi più utilizzato è il Reproducing Kernel Hilbert Space ([Vap99]):

consideriamo il training set $(x_1, y_1), \ldots, (x_l, y_l)$ e definiamo:

$$H_m = \{f | f(x_j) = \sum_{i=1}^{l} \alpha_i K(x_j, x_i),\ \phi[\alpha] \leq m\}$$

$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ è la funzione Kernel ed esprime la distanza tra due vettori. La funzione di complessità $\phi[\alpha]$ di solito viene chiamata $\|f\|_K^2$ ed è una norma in grado di stimare la VC-dimension di H_m.

1.3.4 La Minimizzazione del Rischio Strutturale

Abbiamo introdotto il principio di minimizzazione empirico e ne abbiamo verificato due qualità interessanti:

Coerenza La convergenza all'aumentare del numero dei dati dell'errore empirico con il rischio atteso della funzione trovata.

Consistenza L'avvicinarsi, all'aumentare dei dati o al diminuire della VC-dimension, della funzione trovata dalla Learning Machine con f_H, la miglior funzione che si potrebbe trovare nello spazio delle ipotesi, a parità di VC-dimension.

Ora rivediamo la figura 1.4, nonostante classifichi tutti i dati correttamente è chiaro che la sua struttura sia 'ricercata' o 'forzata', questa nozione intuitiva si traduce direttamente nella *complessità strutturale* della funzione. Negli anni '30 Popper propose il suo metodo per controllare la validità di una teoria, il metodo della non-falsificabilità, questo contemplava la possibilità di trovare degli esempi che, se accadessero, confuterebbero la teoria. In parole povere chi si fiderebbe di una teoria che può spiegare tutto, come ad esempio l'astrologia? È più sicura una teoria più semplice e che possa sbagliare, come ad esempio la meteorologia, ma che in genere sia affidabile.

Con questo razionale usiamo la VC-dimension non solo per assicurare la consistenza del rischio empirico, ma anche per estenderlo con una quantità che chiamiamo rischio strutturale.

Sia:

$$\epsilon(l, h, \eta) = \sqrt{\frac{h \log \frac{2el}{h} - \log \frac{\eta}{4}}{l}}$$

Abbiamo visto nell'equazione (1.22) che con probabilità $(1 - \eta)$ abbiamo un limite del tipo:

$$I[f] \leq I_{emp}[f] + \epsilon(l, h, \eta)$$

Dove il rischio strutturale è proprio $I_{emp}[f] + \epsilon(l, h, \eta)$. Questa disequazione può essere riscritta simbolicamente come:

$$\text{Rischio Atteso} \leq \text{Rischio Strutturale} \tag{1.24}$$

Per controllare il valore del Rischio Strutturale dobbiamo trasformare la VC-dimension in una variabile del problema. L'annidamento che abbiamo imposto allo spazio delle ipotesi può essere sfruttato per esplorare ordinatamente spazi con diverse potenzialità e risolvere la minimizzazione del Rischio Strutturale:

$$\min_{H_m}[I_{emp}[\hat{f}] + \epsilon(l, h, \eta)]$$

dove H_m è il sottospazio che minimizza il rischio strutturale.

Questo principio presenta delle difficoltà, innanzitutto bisogna essere in grado di annidare lo spazio delle ipotesi, possedendo una conoscenza a priori del sistema, inoltre una ricerca esaustiva di H potrebbe impegnare molte risorse di calcolo.

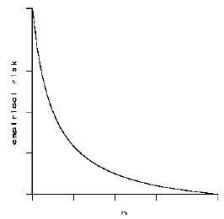

Figura 1.6: Comportamento del rischio empirico al diminuire della VC-dimension

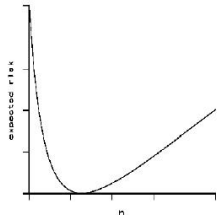

Figura 1.7: Comportamento del rischio strutturale all'aumentare della VC-dimension

In figura 1.6 vediamo il tipico comportamento del rischio empirico, all'aumentare della VC-dimension h esso diminuisce. Questo comportamento è intuitivo in quanto la Learning Machine ha più possibilità di inseguire i dati e di commettere meno errori. Osserviamo però il grafico 1.7, possiede un minimo quando c'è il miglior trade-off tra Rischio Strutturale e Rischio Empirico, ma questo è un minimo locale in quanto all'aumentare di h, nonostante il Rischio Empirico tenda a zero, il Rischio Strutturale aumenta sempre di più.

1.4 Ritorno alla pratica

Consideriamo gli ingredienti per costruire una Learning Machine e vediamo come trasformare i concetti esposti in questi ingredienti, risolvendo dei problemi legati alla complessità del calcolo e fissando degli elementi.

1.4.1 Metodi basati sulla complessità

La Minimizzazione del Rischio Strutturale è per vari motivi molto più affidabile della Minimizzazione del Rischio Empirico, in particolare ci assicura che la capacità di generalizzazione della Learning Machine sia controllata al suo interno. Per utilizzarla però dobbiamo imporre un annidamento dello spazio delle ipotesi H, al fine di controllare la complessità strutturale.

Spesso siamo in grado di definire una funzione $\phi[.]$ che stimi la complessità di una funzione. $\phi[.]$ incorpora la conoscenza a priori sulla soluzione attesa e può essere utilizzata per annidare H in moto tale che se $f \in H_m$ allora $\phi[f] \leq m$.

Data ϕ il problema del rischio strutturale diventa:

$$\begin{array}{c} \text{minimizzare: } I_{emp}[f] \\ \text{soggetto a: } \phi[f] \leq m \end{array} \tag{1.25}$$

e può essere risolto utilizzando la tecnica dei moltiplicatori di Lagrange

$$\begin{array}{c} \text{minimizzare: } I_{emp}[f] + \lambda\phi[f] \\ \text{soggetto a: } \phi[f] \leq m \end{array} \tag{1.26}$$

Si dice che il parametro λ sia un regolarizzatore della soluzione, in effetti pesa la complessità della funzione rispetto al Rischio Empirico, la sua scelta può essere fatta a mano o automatizzata, a seconda delle conoscenze a priori che si hanno sul problema.

La scelta degli ingredienti

Oltre allo spazio di funzioni e alla funzione di Loss, abbiamo aggiunto un nuovo ingrediente alla Learning Machine, la funzione di complessità.

Scegliamo come H un Reproducing Kernel Hilbert Space ([DVEA]). Sia $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ un kernel, e, dato un training set di l elementi, costruiamo una matrice K tale che $\mathrm{K}_{i,j} = K(x_i, x_j)$.

Le funzioni dello spazio H saranno della forma:

$$f(x_j) = \sum_{i=1}^{l} \alpha_i \mathrm{K}_{i,j} \tag{1.27}$$

Mentre la funzione di complessità sarà definita come:

$$\phi[f] = \|f\|_{\mathrm{K}}^2 = \sum_{i,j=1}^{l} \alpha_i \alpha_j \mathrm{K}_{i,j} \tag{1.28}$$

Fissato lo spazio H e la funzione di complessità possiamo scegliere differenti funzioni di Loss, di solito dipendenti dall'algoritmo che si vuole implementare all'interno della Learning Machine.

- Regularized Least Squares (RLS): $V = (y - f(x))^2$
- Support Vector Machine for Regression (SVMR): $V = |y - f(x)|_\epsilon$
- Support Vector Machine for Classification (SVMC): $V = |1 - yf(x)|_+$

Capitolo 2

Lo Spectral Clustering: Analisi

Il nucleo dell'apprendimento dei neonati è l'estrapolazione da esempi di regole, strutture comuni alle informazioni che raggiungono i sensi. L'obiettivo di questa ricerca di regole è fare economia di risorse e risparmiare tempo nell'adattarsi all'ambiente.

Ad esempio la maggior parte dei suoni e delle voci che i neonati ascoltano nascondono al loro interno delle strutture, come le sillabe pronunciate o la regolarità di un rumore meccanico. Fra queste ben poche vengono insegnate esplicitamente al bambino, la maggior parte invece viene diviso in gruppi all'interno del suo cervello in maniera automatica, e i gruppi sono utilizzati per classificare nuovi esempi.

Per capire la natura inconscia di questo processo pensiamo solo che di solito non ci si accorge del proprio accento, finché non ci si confronta con un accento di una differente regione italiana. I dettagli dell'accento, le sue grazie per così dire, fanno parte di quelle regole comuni ai suoni che riceviamo da bambini. Il fatto che difficilmente si riconosce il proprio è un successo nell'economia dei neuroni che ascoltano, infatti non devono compiere quello sforzo in più nell'interpretazione di una frase.

Questo tipo di apprendimento si chiama, nel Machine Learning, Apprendimento Non Supervisionato. Al suo interno sono racchiusi vari metodi, ma sicuramente il metodo che racchiude più tecniche è il Clustering.

2.1 Clustering

Il termine 'cluster' significa letteralmente 'grappolo', ma il suo utilizzo nella lingua inglese può variare da 'gruppo' a 'sciame'.

Nell'accezione usata nel Machine Learning possiamo attribuirgli il significato di 'gruppo compatto', dove per compatto si intende che, data una qualche misura di distanza, gli elementi del gruppo sono vicini tra loro e lontani dagli elementi degli altri gruppi.

Continuando ad utilizzare la terminologia del Machine Learning chiameremo i dati in input il *training set*.

2.1.1 Come si creano dei Cluster?

Non esiste una definizione esatta di Clustering e quindi non si può stabilire una procedura unica per trovare diversi cluster; guardiamo ad esempio la figura 2.1 e chiediamoci quanti gruppi vediamo, la risposta può variare tra un solo gruppo a tanti quanti sono gli elementi. Ma in questo intervallo sembra che la soluzione più corretta possa essere quattro, due gruppetti a destra e un gruppetto a sinistra racchiuso da un gruppo a corona. Dare una spiegazione di questa intuizione aiuta a stabilire un criterio per la ricerca dei cluster.

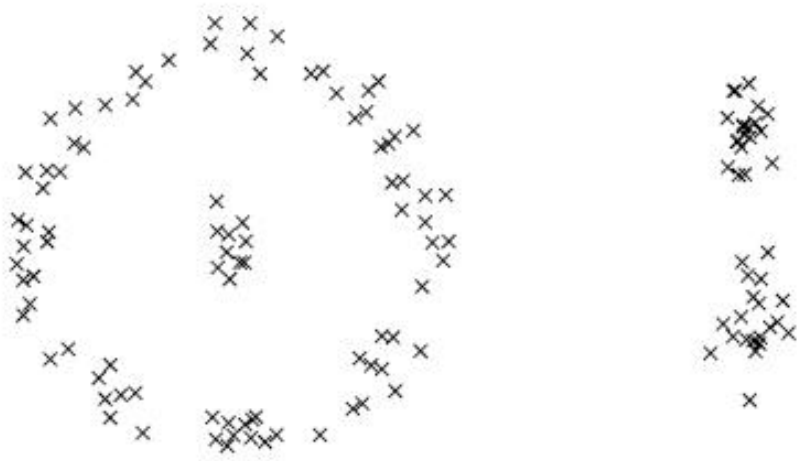

Figura 2.1: Quanti cluster ci sono?

L'esempio in figura 2.1 è costruito a mano, ma molti esperimenti pratici come la segmentazione di immagini presentano difficoltà simili, in cui ci sono molti possibili raggruppamenti dei pixel, ma qualcuno è più ragionevole di altri. Uno dei concetti trainanti è la distanza, o la similarità, tra i dati. Mentre osservando la figura 2.1 la distanza è ovviamente quella euclidea in genere la si può scegliere fra diverse possibilità, come vedremo nella sezione 2.1.3. Data una misura di distanza abbiamo diversi approcci per sfruttarla.

2.1.2 Vari Approcci

Gli approcci principali al clustering sono sostanzialmente due: gerarchici e partizionali.

Nei primi viene costruito un albero, chiamato dendrogramma, sui dati. La radice di questo albero corrisponde all'intero training set, mentre le foglie sono i singoli dati. Salendo dalle foglie verso la radice si trovano partizioni unendo, con un criterio stabilito, le partizioni al livello inferiore. Il punto chiave è scegliere il livello dell'albero con la migliore partizione.

Nei secondi si prova a inferire un modello che abbia generato i dati e lo si utilizza per cercare le partizioni, in questi approcci il partizionamento deve minimizzare una funzione energia definita a partire dalla misura di distanza scelta.

Il K-Means e il Fuzzy C-Means sono gli approcci partizionali più famosi e semplici ([F.98],[ePH73]), in essi il modello che ha generato i dati è la distribuzione gaussiana.

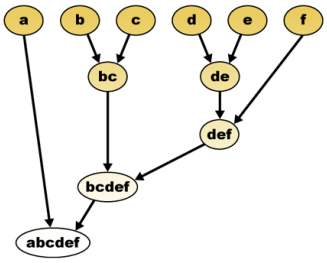

Figura 2.2: Un dendrogramma, dalle foglie alla radice

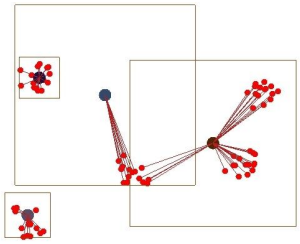

Figura 2.3: Un partizionamento di punti sul piano

Un terzo approccio meno diffuso consiste nel considerare i dati come nodi di un grafo, i Markov Random Fields ([eD84]). Lo Spectral Clustering rientra in questa filosofia, ma lo studio delle sue proprietà è radicalmente diverso, vedremo nella sezione 2.3 che i dati del training set possono essere considerati come l'approssimazione di un Manifold, uno spazio topologico le cui proprietà possono essere studiate attraverso una matrice chiamata Laplaciano.

Gli approcci hanno tutti in comune il fatto di dare la massima importanza alla misura di distanza scelta per i dati, spesso il successo di un algoritmo di clustering consiste nel scegliere la misura adatta.

2.1.3 Misurare le distanze

Per confrontare i dati bisogna introdurre una nozione di distanza tra essi, una funzione che, dati due esempi, ci dia una misura della loro vicinanza (nel clustering distanza diventa sinonimo di similarità).

Una distanza su un insieme X è una funzione $d : X \times X \to \mathbb{R}$ che deve soddisfare le seguenti proprietà:

- $d(x, y) \geq 0$
- $d(x, y) = 0 \Leftrightarrow x = y$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$ (disuguaglianza triangolare)

In questo caso la coppia (X, d) viene chiamata *Spazio Metrico*. Se $X = \mathbb{R}^n$ una misura che si utilizza spesso è quella euclidea:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

In questo caso $(\mathbb{R}^n, d)$ si chiama spazio euclideo e lo utilizzeremo nell'algoritmo di K-Means.

Nel caso dei grafi le connessioni tra i vertici possono essere pesate a seconda della distanza dei vertici stessi, il peso è una funzione $w : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ simile alla distanza, con l'unica differenza che $w(x, x) = 1$.

Nel caso dei grafi un peso che vale 0 significa che i due nodi non sono connessi, se vale 1 la connessione ha valore massimo.

La funzione che utilizziamo nello Spectral Clustering è la gaussiana:

$$w(x, y) = e^{-\frac{\|x-y\|_2^2}{2\sigma^2}} \tag{2.2}$$

Possiamo pensare che questa funzione ci restituisce, data la distanza tra due punti $\|x - y\|_2^2$, la probabilità che i due vettori x, y siano uguali.

Il parametro σ si chiama varianza della gaussiana ed è allo stesso tempo un pregio e un difetto di questa funzione.

È un pregio in quanto siamo in grado di regolare la funzione, appiattendola o stringendola, in modo tale da favorire di più la vicinanza, o appiattire le probabilità.

È un difetto perché spesso la conoscenza a priori del problema non è sufficiente per capire un suo valore corretto. In questo caso si possono eseguire delle analisi di pre-processing sui dati, oppure orientarsi con dei tentativi. Nel caso dell'algoritmo che proporrò più avanti la scelta di questo dato è automatica e dipendente dai dati.

2.2 K-Means

All'interno degli algoritmi partizionali K-Means([McQ67]) è il più semplice e rappresentativo.

Userò negli esperimenti un confronto tra lo Spectral Clustering e il K-Means, per sottolineare come il primo approccio superi delle difficoltà intrinseche di molti algoritmi partizionali che estendono il K-Means.

Il confronto è diretto nonostante i due approcci siano di natura diversa, il primo non è parametrico al contrario del secondo.

Per contro i due algoritmi sono spesso utilizzati insieme, infatti attraverso lo Spectral Clustering si è in grado di modificare e ridurre la dimensionalità dei dati e di utilizzare K-Means in seguito a questo passaggio.

Per capire perché K-Means funzioni bisogna riprendere la discussione sulla Maximum Likelihood della sezione 1.1.3.

2.2.1 Clustering Parametrico e Maximum Likelihood

Sia $X_{ts} = (x_1), \ldots, (x_n)$ il training set e $Y_{ts} = \{y_j | j = 1, \ldots, C\}$ un insieme di prototipi dei cluster ω_j. Assumiamo che sia noto il numero di classi, la probabilità a priori di una classe $p(\omega_j)$ e che i dati siano generati da una distribuzione

di probabilità dipendente dal cluster a cui appartengono $p(x|\omega_j, \theta_j)$. θ_j è un insieme di parametri che regola la distribuzione di probabilità j-esima, se ad esempio scegliessimo la funzione gaussiana ci servirebbe la coppia (μ_j, σ_j), la media e la varianza.

Sotto queste ipotesi la probabilità che un x generico sia estratto è data da:

$$p(x, \theta) = \sum_{j=1}^{c} p(x|\omega_j, \theta_j)p(\omega_j) \tag{2.3}$$

Uno dei metodi più conosciuti per stimare l'insieme dei parametri $\theta = \theta_{1,\dots,j}$ è basato sulla *maximum likelihood* [ePH73]. La *likelihood* di un training set X è data dalla probabilità totale:

$$L(\theta) = \prod_{k=1}^{n} p(x_k, \theta) \tag{2.4}$$

La *maximum likelihood* stima un insieme di parametri $\hat{\theta}$ che massimizza $L(\theta)$. Intuitivamente si può pensare che $\hat{\theta}$ indichi il modello che più probabilmente ha generato quei dati.

Sotto opportune ipotesi ([F.98]) si può mostrare che l'algoritmo di K-Means massimizza $L(\theta)$.

2.2.2 L'algoritmo di K-Means

L'algoritmo del K-Means in sé è semplice, preso il training set stabiliamo a priori il numero k di classi in cui vogliamo partizionarlo. Istanziamo quindi k prototipi, o centroidi, che rappresentano l'elemento medio di una classe. Nell'inizializzazione i prototipi vengono scelti casualmente nell'ipercubo contenente il training set; in seguito la loro posizione viene aggiornata in modo tale da minimizzare una funzione di energia:

$$V = \sum_{i=1}^{k} \sum_{j \in TS} d(x_j, \mu_i)^2 \tag{2.5}$$

dove μ_i è l'i-esimo prototipo e d di solito è la distanza euclidea.

I passi dell'algoritmo sono:

input: Il training set, il numero k di classi, la misura di distanza

1. Inizializzare casualmente k prototipi nell'ipercubo minimo che contiene il training set
2. Assegna ogni punto *esclusivamente* al cluster del prototipo più vicino
3. Ogni prototipo viene ricalcolato in modo che si trovi nel baricentro dei dati della propria classe
4. Si ritorna al punto 2 a meno che una condizione di stop non venga soddisfatta.

output: i cluster o i prototipi

Ci sono varie possibilità per la condizione del punto 4, una delle condizioni tipiche è richiedere che la posizione dei prototipi rimanga invariata rispetto ad un ciclo precedente.

La funzione di energia 2.5, che viene minimizzata dal K-Means, ha un minimo globale ma anche dei minimi locali. Uno dei problemi di questo algoritmo è il fatto di cadere spesso in questi minimi. Per risolvere in parte questo problema l'algoritmo viene eseguito più volte e vengono confrontati i diversi risultati.

Nel punto 2 è enfatizzata la parola *esclusivamente*, il motivo è che in altri approcci, come il Fuzzy C-Means, ad ogni punto viene assegnata una probabilità di appartenere ai diversi cluster, non necessariamente uguale ad uno.

La complessità computazionale del K-Means è in media $O(nk)$ dove n è la cardinalità del training set e k è il numero di prototipi stabilito a priori, questo perché di solito l'algoritmo converge rapidamente e quindi ci sono poche iterazioni al punto 4.

2.3 Lo Spectral Clustering

Negli ultimi anni la tecnica dello Spectral Clustering si è diffusa come valida concorrente alle precedenti tecniche di Clustering. I motivi sono vari, innanzitutto la semplicità dell'implementazione, per la quale è sufficiente una libreria di algebra lineare, e in secondo luogo risultati sbalorditivi che superano molte difficoltà che sembravano insormontabili.

La semplicità è solo apparente, la teoria sottostante è la Teoria dei Grafi, una materia vasta e profonda che coinvolge aree scientifiche completamente differenti. La visione che esporrò sui grafi è quindi solo una piccola finestra per capire il funzionamento di questa tecnica, dettagliando diverse varianti, spiegandone differenze e similarità.

L'esposizione avverrà a livelli successivi di dettaglio, da una visione intuitiva si arriverà alle dimostrazioni dei risultati. Per capire lo Spectral Clustering è sufficiente un po' di logica e la conoscenza dell'algebra lineare, oltre alla disponibilità a rimanere stupiti.

2.3.1 Una visione intuitiva

Ho motivato l'approccio del Clustering come un automatismo del nostro cervello. Ma spesso si utilizza questa tecnica nella vita quotidiana, ad esempio quando si confronta un evento con altri eventi simili oppure si prova a decifrare una cattiva calligrafia confrontando tra loro le lettere.

Nel caso dello Spectral Clustering vediamo in figura 2.4 un foglio di carta sagomato. Supponiamo di volerlo tagliare con un paio di forbici tagliando poca carta e dividendolo in due parti abbastanza voluminose, il taglio uno è un buon taglio ma forse il taglio due è migliore. Operando questa scelta eseguiamo qualcosa di molto simile allo Spectral Clustering, solo che invece di lavorare su un grafo lavoriamo su un foglio di carta.

Nella figura 2.5 osserviamo un esempio simile al foglio di carta, ma con una variabile in più, la densità del legno. Volendo spezzare la tavola raffigurata sembra chiaro che convenga farlo dove la densità è minore, ma bisogna ricordarsi

della necessità di ottenere due pezzi entrambi *voluminosi*, quindi si preferisce la seconda linea alla prima.

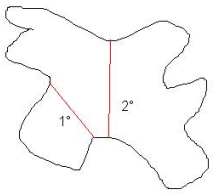

Figura 2.4: Il secondo taglio è più centrale e più equilibrato del primo

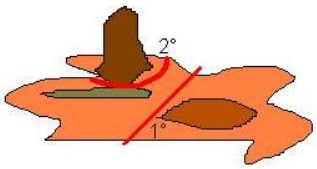

Figura 2.5: Vista la diversa densità il taglio centrale non è quello più equilibrato

Il foglio di carta o il pezzo di legno sono esempi di un oggetto matematico chiamato Manifold.

Un Manifold è uno spazio matematico che localmente si comporta come uno spazio euclideo; ad esempio la superficie terrestre è un Manifold. Nel problema del clustering supponiamo che i dati siano stati estratti da un Manifold di cui non conosciamo la struttura. Questa può essere approssimata considerando i dati come vertici di un grafo e definendo su questo grafo il Laplaciano, un operatore che si comporta sul grafo allo stesso modo in cui si comporterebbe sul Manifold.

2.3.2 Il Grafo

Informalmente un grafo è un insieme di nodi e connessioni tra essi. Internet, un circuito elettrico o le amicizie fra le persone sono esempi di grafi. Osserviamo dunque che il concetto di grafo si può applicare a realtà ben differenti e che quindi lo studio delle sue proprietà è di grande interesse sia nella matematica che nell'informatica.

Formalmente dati dei punti $v_1, \ldots, v_n$ si definisce come grafo una coppia $G = (V, E)$ dove $V = (v_1, \ldots, v_n)$ è l'insieme dei vertici, o nodi, del grafo ed $E = \{(v_i, v_j) | v_i, v_j \text{ sono connessi})\}$ è l'insieme degli archi che connettono dei vertici distinti. Non è detto che ogni vertice sia connesso con ogni altro, né che il grafo sia connesso, cioé esista un percorso che colleghi ogni possibile coppia di punti.

Di solito i vertici vengono chiamati con il loro indice, in modo tale che $(v_i, v_j) = (i, j)$. In questo caso studieremo grafi in cui la connessione è pesata da una misura di similarità $w(i, j) \geq 0$.

Se il grafo è pesato, l'insieme delle connessioni E può essere rappresentato in una notazione più comoda come una *matrice di adiacenza pesata* $W = (w_{i,j})_{i,j=1,\ldots,n}$. Se $w_{i,j} = 0$ significa che i due vertici (v_i, v_j) non sono collegati, al contrario se $w_{i,j} = 1$ allora il collegamento ha valore massimo. Supponiamo che il grafo sia indiretto e che quindi $w_{i,j} = w_{j,i}$; la matrice W risulta simmetrica.

Il grado d_i di un vertice v_i è uguale a

$$d_i = \sum_{j=1}^{n} w_{i,j}$$

Il grado di un vertice quindi non è altro che la somma dei pesi delle sue connessioni con altri vertici adiacenti. Definiamo la matrice D come

$$D_{i,j} := \begin{cases} d_i & \text{se } i = j \\ 0 & \text{altrimenti} \end{cases}$$

D quindi è una matrice diagonale contenente i gradi dei vertici. Dato un sottinsieme $A \subset V$ si denota il suo complementare $V/A = \bar{A}$. Il volume di A sarà $Vol(A) = \sum_{i \in A} d_i$.

La misura di similarità è strettamente legata alla nozione di distanza vista in 2.1.3, ma spesso vengono usate alternative completamente differenti.

2.3.3 Misure di similarità

Esistono diverse misure di similarità in letteratura [VL06] per costruire la matrice di adiacenza W, molte di queste sono utilizzate perché conducono ad una matrice sparsa, che ha dei grossi vantaggi computazionali.

ε-vicinanza: Sono considerate connesse le coppie di vertici la cui distanza euclidea è inferiore a ε, in tal caso la connessione vale 1, altrimenti 0.

k-primi vicini: si connette ogni vertice solo con i propri k primi vicini. Visto che la relazione dei primi vicini non è simmetrica, di solito si usano tutte le connessioni, cosicché un vertice potrebbe essere collegato a più di k vertici

connessione completa: si usa una similarità positiva, nel caso di questa tesi utilizziamo la funzione di similarità Gaussiana $w(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$. Il parametro σ controlla l'ampiezza della vicinanza, come abbiamo visto nella sezione 2.1.3

In questa tesi tratto simulazioni o casi in cui non ho problemi computazionali, quindi ho scelto di focalizzare l'attenzione sull'utilizzo della gaussiana e sulla scelta del parametro σ.

2.3.4 Il Taglio

Considerato il training set come se fosse un grafo pesato G, il segreto dello Spectral Clustering consiste nel 'tagliare' nel miglior modo possibile questo grafo G. Un taglio è un partizione del grafo in due insiemi disgiunti e ha un valore, chiamato cut[1]. Se A e B sono due insiemi disgiunti il valore del taglio è la somma dei pesi delle connessioni che vengono 'tagliate':

$$cut(A, B) = \sum_{i \in A, j \in B} w_{i,j} \tag{2.6}$$

Un primo approccio al problema del Clustering utilizzando un grafo è stato proprio quello di minimizzare questa quantità ([WZ93]). Questo purtroppo conduce a degli scarsi risultati in quanto spesso viene 'tagliato' via solo un vertice, come in figura 2.3.4.

[1] Taglio in inglese

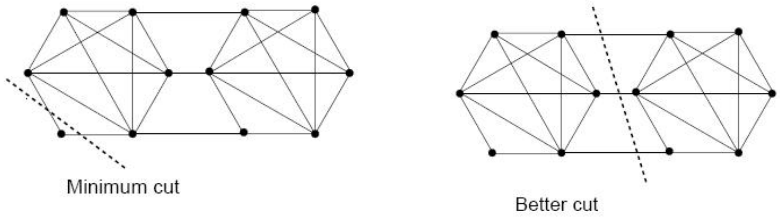

Figura 2.6: Piuttosto che tagliare via un vertice si preferisce un taglio più bilanciato

Quello che rende migliore il taglio in figura 2.3.4 è il volume delle parti tagliate, Shi e Malik [Shi00] hanno sfruttato questa intuizione estendendo il concetto di taglio e aggiungendo il vincolo che i due insiemi divisi siano abbastanza voluminosi, proprio come abbiamo visto nel paragrafo 2.3.1; si sperimenta che questo conduce a risultati soddisfacenti nel Clustering.

Si definisce il *taglio normalizzato* come il valore di un taglio di un grafo, normalizzato rispetto ai volumi degli insiemi che ne risultano:

$$Ncut(A,B) = cut(A,B)\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right) \tag{2.7}$$

Con questa formulazione minimizzare $Ncut$ 'tagliando' via un solo vertice non è più possibile perché i volumi di A e B in questo modo non sono massimizzati.

In questo senso Shi e Malik [Shi00] sono stati in grado di dimostrare che minimizzare il taglio normalizzato conduce contemporaneamente alla massimizzazione della connessione interna degli insiemi. Vediamo come sia possibile definendo $assoc(A,B) = \sum_{i\in A, j\in B} w_{i,j}$ come una misura della associatività di un cluster e definiamo l'associazione normalizzata

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \tag{2.8}$$

dove assoc(A,A) e assoc(B,B) misurano la coesione interna di un insieme. $Nassoc$ è una misura che, nel caso del Clustering, ci interessa massimizzare in quanto ci assicura che i cluster prodotti siano voluminosi.

$Ncut$ e $Nassoc$ sono in stretta relazione l'uno con l'altro, in effetti si può vedere che

$$Ncut(A,B) = 2 - Nassoc(A,B) \tag{2.9}$$

Questo significa che i due criteri che si vogliono soddisfare per partizionare il grafo sono in relazione e che minimizzare $Ncut$ equivale a massimizzare $Nassoc$.

Posto in questi termini il problema è chiaro, ma la sua risoluzione con un algoritmo presenta delle difficoltà. In effetti se si pensa di generare tutte le partizioni possibili di un grafo alla ricerca di quella che minimizza Ncut, si ha di fronte una complessità esponenziale $O(2^n)$. Forse si può contare sulla programmazione dinamica, ma prima di perdere ulteriore tempo in tentativi combinatoriali diciamo che si può dimostrare che il taglio normalizzato è NP-hard ([Shi00]), la dimostrazione mostra che PARTITION, un classico problema NP-completo, è riducibile a NCUT.

Dal momento che i problemi di Clustering hanno spesso un elevato numero di dati c'é bisogno di un algoritmo efficiente che approssimi $Ncut$, e questo è stato trovato per vie algebriche.

2.4 Il Laplaciano

La matrice di similarità W rappresenta le connessioni che caratterizzano un grafo, dalla matrice W si può costruire una matrice di fondamentale importanza nello spectral Clustering: il Laplaciano. Il Laplaciano ha diverse definizioni in letteratura, di solito si distingue tra Laplaciano e Laplaciano normalizzato, due oggetti correlati con diverse proprietà. Il Laplaciano si definisce come:

$$L = D - W \tag{2.10}$$

dove D è la matrice diagonale contenente i volumi dei singoli vertici. Il Laplaciano normalizzato è

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \tag{2.11}$$

dove I è la matrice identità. Dal momento che il Laplaciano normalizzato è una matrice simmetrica i suoi autovalori sono reali e non-negativi. Inoltre possiamo studiare le caratteristiche di questi autovalori attraverso il quoziente di Rayleigh ([Gol89]) di $\mathcal{L}$. Sia g un vettore colonna in $\mathbb{R}^n$, allora:

$$\begin{aligned} \frac{< g, \mathcal{L} g >}{< g, g >} &= \frac{< g, D^{-1/2} \mathcal{L} D^{1/2} g >}{< g, g >} \\ &= \frac{< f, Lf >}{< D^{1/2} f, D^{1/2} f >} \\ &= \frac{\sum_{i,j=1}^{n} (f(i) - f(j))^2 w(i,j)}{\sum_{i \in V} f^2(i) d_i} \end{aligned} \tag{2.12}$$

dove $g = D^{1/2} f$. Dall'equazione 2.12 si può dedurre che gli autovalori sono reali e non negativi, usualmente questi autovalori sono indicati con $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$.

L'insieme degli autovalori è chiamato *spettro* di $\mathcal{L}$ (o del grafo associato G). Se $\mathbb{1}$ denota il vettore che assume il valore costante 1 allora $D^{1/2}\mathbb{1}$ è un autovettore di $\mathcal{L}$ con autovalore 0. Inoltre

$$\lambda_1 = \inf_{f \perp D\mathbb{1}} \frac{\sum_{i,j=1}^{n} (f(i) - f(j))^2 w(i,j)}{\sum_{i \in V} f^2(i) d_i} \tag{2.13}$$

Questa formulazione corrisponde in maniera naturale all'operatore di Laplace-Beltrami su Manifold Riemanniani:

$$\lambda_M = \inf_{\int_M f = 0} \frac{\int_M |\nabla f|^2}{\int_M |f|^2} \tag{2.14}$$

2.4.1 Risolvere Ncut con il Laplaciano

Attraverso il Laplaciano possiamo risolvere il problema di Ncut, definiamo la forma della soluzione al problema come un vettore f_i che assume valori differenti a seconda se v_i appartiene o meno ad A:

$$f_i := \begin{cases} \sqrt{\frac{Vol(\overline{A})}{Vol(A)}} & \text{se } i \in A \\ -\sqrt{\frac{Vol(A)}{Vol(\overline{A})}} & \text{se } i \in \overline{A} \end{cases} \tag{2.15}$$

Si vede che:

$$\begin{aligned} f'Lf &= \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2 \\ &= \sum_{i\in A, j\in\overline{A}} w_{i,j}\left(\sqrt{\frac{Vol(\overline{A})}{Vol(A)}} + \sqrt{\frac{Vol(A)}{Vol(\overline{A})}}\right)^2 + \sum_{i\in\overline{A}, j\in A} w_{i,j}\left(-\sqrt{\frac{Vol(A)}{Vol(\overline{A})}} - \sqrt{\frac{Vol(\overline{A})}{Vol(A)}}\right)^2 \\ &= 2cut(A,\overline{A})\left(\frac{Vol(\overline{A})}{Vol(A)} + \frac{Vol(A)}{Vol(\overline{A})} + 2\right) \\ &= 2cut(A,\overline{A})\left(\frac{Vol(\overline{A}) + Vol(A)}{Vol(A)} + \frac{Vol(A) + Vol(\overline{A})}{Vol(\overline{A})}\right) \\ &= 2Vol(V)Ncut(A,\overline{A}). \end{aligned} \tag{2.16}$$

Inoltre si vede che

$$\begin{aligned} \sum_{i=1}^{n} d_i f_i &= \sum_{i\in A} d_i \sqrt{\frac{Vol(\overline{A})}{Vol(A)}} - \sum_{i\in\overline{A}} d_i \sqrt{\frac{Vol(A)}{Vol(\overline{A})}} \\ &= Vol(A)\sqrt{\frac{Vol(\overline{A})}{Vol(A)}} - Vol(\overline{A})\sum_{i\in\overline{A}} d_i \sqrt{\frac{Vol(A)}{Vol(\overline{A})}} \\ &= 0. \end{aligned} \tag{2.17}$$

In altre parole, il vettore f definito nell'equazione (2.15) è ortogonale al vettore $D\mathbb{1}$. Infine si nota come $f'Df = Vol(V)$. Il problema di Ncut potrebbe essere riscritto come

$$\min_A f'Lf, \text{ soggetto a } f \text{ come nella definizione 2.15, } Df \perp \mathbb{1}, f'Df = Vol(V) \tag{2.18}$$

Se l'approccio a questo problema rimane combinatoriale, allora Ncut rimane NP-hard; per contro ci accorgiamo che possiamo rilassare i valori di f e ottenere una formulazione con un vincolo in meno:

$$\min_A f'Lf, \text{ soggetto a } Df \perp \mathbb{1}, f'Df = Vol(V) \tag{2.19}$$

La forma della definizione 2.19 è di cruciale importanza perché è nella forma vista nella definizione 2.12:

$$\lambda_1 = \underset{Df\perp\mathbb{1}}{\operatorname{argmin}} \frac{< f, Lf >}{< f, Df >} \tag{2.20}$$

Se si sostituisce $g := D^{1/2}f$ si può riformulare l'equazione 2.20 in:

$$\lambda_1 = \underset{g \perp D^{1/2}\mathbb{1}}{\operatorname{argmin}} \frac{g' D^{-1/2} L D^{-1/2} g}{g'g} \tag{2.21}$$

Dal momento che $D^{-1/2}LD^{-1/2} = \mathcal{L}$ è il Laplaciano normalizzato, il problema 2.18 può essere risolto trovando il secondo autovalore ed il corrispettivo autovettore di $\mathcal{L}$. Se risostituiamo $f = D^{-1/2}g$ vediamo che f è il secondo autovettore del sistema generalizzato di $Lv = \lambda Dv$. Il vettore f e il vettore g trasportano le stesse informazioni, in seguito considereremo f come soluzione del problema del taglio normalizzato, come suggerito in [VL06], perché la moltiplicazione per $D^(1/2)$ potrebbe creare degli artefatti indesiderati.

Ci sono molti commenti che si potrebbero fare nel derivare lo Spectral Clustering con questo rilassamento. Il fatto più importante è che non c'è alcuna garanzia della qualità della soluzione confrontata con quella esatta. In particolare Guattery e Miller ([Gua98]) hanno mostrato che la differenza tra le due soluzioni potrebbe essere arbitrariamente grande. Altri approcci sono stati introdotti, come Bie e Cristianini ([Bie06]).

Comunque si può dire che, dal momento che $Ncut \leq 2\sqrt{\lambda_1}$ ([F.97]), se il secondo autovalore è piccolo, allora anche il taglio normalizzato avrà un valore piccolo.

2.4.2 Utilizzare il secondo autovettore per bipartizionare i dati

Grazie al rilassamento siamo in grado di trovare in tempo polinomiale il secondo autovettore del Laplaciano, ma bisogna ancora decidere come utilizzarlo per creare dei cluster. La definizione 2.15 suggerisce di dividere i vertici a seconda se il corrispondente valore nell'autovalore sia positivo o negativo. Nella pratica questo funziona bene; se ordiniamo il secondo autovettore spesso scopriamo che è della tipica forma a scalino in figura 2.7, troviamo una netta distinzione dei due cluster i cui vertici assumono valori simili positivi o negativi.

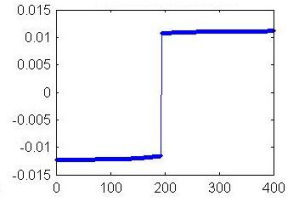

Figura 2.7: Non ci si può sbagliare

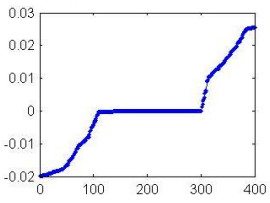

Figura 2.8: Ci si può sbagliare

Spesso capita che il secondo autovettore sia, per così dire, continuo, non ci siano nette variazioni (figura 2.8). In questo caso non si può essere sicuri che la positività sia un metodo che distingue chiaramente due cluster, per questo in Shi e Malik ([Shi00]) è stato proposto di provare diversi offset e calcolare per ognuno di essi il valore di Ncut, cercando l'offset che minimizza Ncut. Questa tecnica ha una complessità inferiore a quella della ricerca degli autovettori e viene eseguita in seguito, quindi non comporta un significativo aumento della complessità[2]. Dal momento contiene anche il caso in cui l'autovettore è ben diviso, ho scelto di utilizzarla come tecnica per cercare dei cluster.

Questa tecnica può essere estesa nel caso in cui vogliamo suddividere i dati in più classi, rimanendo in tempo polinomiale, comunque vediamo un'analisi più dettagliata del Clustering Multiclasse nella sezione 2.4.3.

[2]Entro le condizioni dettate, la complessità non aumenta per il teorema di Linear Speed-Up

2.4.3 Estensione multiclasse

Per quanto visto finora il taglio è in grado di bipartizionare un grafo. Nel Clustering tuttavia si desidera poter suddividere un training set in più classi, quindi di multipartizionare il grafo. Fra i vari approcci possibili presento i due principali, che costituiscono due idee completamente differenti.

Il primo approccio si basa sul fatto che lo Spectral Clustering riduce la dimensionalità del training set.

Dal momento che il Laplaciano è una matrice simmetrica i suoi autovettori sono perpendicolari tra loro e generano l'autospazio legato al Laplaciano, siano $t_0, \ldots, t_k$ i primi $k+1$ autovettori, se si sostituisce ogni $v_i \in \mathbb{R}^n$ con le componenti i-esime dei primi k[3] autovettori, in modo tale che $v_i^{new} = (t_{i,1}, \ldots, t_{i,k}) \in \mathbb{R}^k$ i nuovi punti vivranno in uno spazio a ridotta dimensionalità; la dimensionalità di cui si parla non è il numero di dimensioni n dello spazio $\mathbb{R}^n$ in cui vengono proiettati, bensì la quantità di informazione che ognuno di loro porta con sé.

Cercare cluster in questo spazio è molto più semplice e spesso viene utilizzato l'algoritmo di clustering K-Means proprio in questo spazio. Notiamo che l'autovettore t_0 non viene utilizzato in quanto il suo valore è sempre $D^{1/2}\mathbb{1}$.

In figura 2.9 vediamo un esempio classico di dati da partizionare, di solito K-Means raggiunge risultati non conformi alla divisione che ci si aspetta, che prevede la suddivisione delle due mezzelune. D'altra parte K-Means si comporta bene solo se i cluster hanno una forma ovoidale, o comunque in un problema linearmente separabile, come quello in figura 2.10.

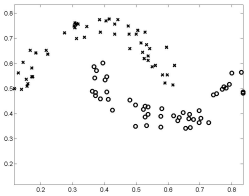

Figura 2.9: I dati a mezzaluna possono essere partizionati da K-Means ...

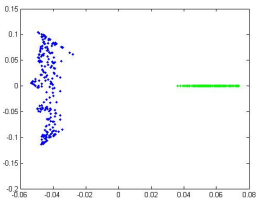

Figura 2.10: ... conoscendo la loro controparte spettrale

il problema di questo approccio è capire quando un autovettore porta un'informazione corretta per il clustering, oppure aumenta semplicemente la dimensionalità dei dati. L'approccio seguito da alcuni, come Marina Meila ([MMeX97]), è quello di considerare la differenza che c'é tra un autovettore e il successivo.

Questa misura, chiamata *gap*, dovrebbe sottolineare l'ultimo autovettore utile al clustering nella dimensione ridotta. Vedremo nel paragrafo 2.4.4 il motivo di questa scelta.

Un approccio completamente differente consiste nel utilizzare solo il secondo autovettore per bipartizionare il grafo, quindi ripetere il calcolo del Laplaciano[4] e degli autovettori, continuando ricorsivamente sulle partizioni ottenute.

Questa tecnica ha il vantaggio di utilizzare sempre l'autovettore che trasporta la maggior quantità di informazione, il secondo. Anche in questa caso si presenta il problema di capire quando fermarsi, infatti una bipartizione avviene sempre, e capire la sua qualità è uno dei problemi generali degli algoritmi di Clustering.

[3] Non è detto che $k \leq n$

[4] Nel caso in cui la similarità non sia influenzata dal minor numero di elementi non è necessario ricalcolare il Laplaciano.

2.4.4 La Matrice a Blocchi

In questa sezione approfondiamo il motivo che risiede nell'utilizzo del *gap* visto nella sezione precedente (2.4.3).

Nel caso in cui il grafo sia sconnesso il Laplaciano risulta essere una matrice a blocchi, dove ogni blocco rappresenta una componente connessa del grafo. In questo caso la molteplicità dell'autovalore zero è pari al numero di componenti connesse e i relativi autovettori sono dei vettori indicatore per ogni componente. Questo significa che se A_i è una componente connessa ma sconnessa dal grafo, un autovettore t_i, relativo ad un autovalore uguale a zero, è della forma:

$$t_{i,j} := \begin{cases} 0 & \text{se } j \in A_i \\ \alpha & \text{se } j \notin A_i \end{cases} \tag{2.22}$$

In questo caso è molto semplice trovare dei cluster, considerando semplicemente le componenti sconnesse del grafo.

Ci sono casi in cui il Laplaciano è una matrice a blocchi leggermente perturbata. Il grafo è connesso, ma in realtà sono facilmente distinguibili dei cluster che sono tra loro praticamente sconnessi, a meno di qualche connessione. In questo caso un solo autovalore sarà uguale a zero, ma altri saranno piccoli e i corrispondenti autovettori avranno una forma simile a alla figura 2.22. Se la perturbazione non è troppo grande, si è in grado di trovare facilmente dei cluster.

Questo risultato ci viene assicurato dal teorema di Davis-Kahan [eKW70] nel caso in cui il *gap* sia grande. Il *gap* è la massima differenza tra due autovalori consecutivi. Più grande è il *gap*, più gli autovettori sono simili alla forma nella figura 2.22 e di conseguenza più facile è trovare dei cluster.

L'affermazione del teorema si indebolisce con il diminuire del *gap* o l'aumentare della perturbazione, in tal caso potrebbe essere necessario considerare ulteriori autovettori.

Bisogna osservare che l'algoritmo di bipartizione ricorsiva che verrà usato in seguito è in grado di trovare gli stessi cluster nel caso in cui la matrice sia a blocchi, infatti nonostante consideri soltanto il secondo autovettore, il cui autovalore sarà zero come diretta conseguenza, lo spettro dei sottografi, nel caso fossero ancora sconnessi, continua a consentire una bipartizione ottimale in quanto il proprio secondo autovalore sarà ancora uguale a zero.

2.5 Un algoritmo generico

Dopo aver visto il funzionamento dello Spectral Clustering e alcuni dei suoi aspetti, procediamo a definire un algoritmo generico che bipartizioni un grafo.

Normalized Spectral clustering
input: matrice di similarità $S \in \mathbb{R}^{n,n}$, una condizione di stop

1. Costruire una matrice di similarità $W \in \mathbb{R}^{n,n}$ utilizzando una misura di similarità come quell descritte nella sezione 2.3.3
2. Calcolare il Laplaciano $\mathcal{L}$
3. Calcolare gli autovettori $t_1, \ldots, t_n$ di $\mathcal{L}$
4. Utilizzare il secondo autovettore per bipartizionare il grafo
5. Fermarsi se vale la condizione di stop, oppure continuare ricorsivamente sulle partizioni ottenute

output: Clusters $A_1, \ldots, A_k$ con $A_j = \{j | y_j \in C_i\}$

Nel caso operiamo Clustering Multiclasse con il secondo approccio della sezione 2.4.3 è sufficiente sostituire i punti 4 e 5 con

4 Sia $T \in \mathbb{R}^{n \times k}$ la matrice contenente i vettori $t_1, \ldots, t_k$ come colonne.

5 Per $i = 1, \ldots, n$ sia $y_i \in \mathbb{R}^k$ il vettore corrispondente alla i-esima riga di T.

6 Partizionare i punti $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ con l'algoritmo K-Means nei cluster $C_1, \ldots, C_k$

Visto che K-Means nel caso medio ha una complessità di $O(Cn)$, dove C è una costante, il costo maggiore nell'algoritmo si ha per la risoluzione degli autovettori. In genere un'operazione del genere richiede $O(n^3)$ operazioni per essere compiuta, dove n è il numero di nodi del grafo.

Dal momento che di solito il numero di autovettori richiesti è fissato, questo costo può scendere a $O(Cn^2)$ operazioni, dove C è il numero di autovettori richiesto. Un ulteriore miglioramento si può ottenere nel caso di grafi molto sparsi connessi solo localmente. In questo caso si può utilizzare il metodo di Lanczos([Gol89]) che ha una complessità attesa di $O(n^{\frac{3}{2}})$, supponendo che il numero massimo di connessioni per vertice sia fissato da una costante che non dipende dal numero di nodi del grafo.

Un ulteriore problema è costituito dallo spazio usato dall'algoritmo, infatti il Laplaciano occupa uno spazio proporzionale a $O(n^2)$, e nel momento in cui si prova a segmentare un'immagine, anche piccola, è facile che i normali Personal Computer non siano in grado di eseguire il calcolo.

A supporto di queste due problematiche si può utilizzare l'estensione out of sample[5], che consente di utilizzare solo una parte del training set e di classificare nuovi dati utilizzando i cluster acquisiti.

2.6 L'estensione Out-of-Sample

Come possiamo classificare dei nuovi esempi, diversi dal training set, che ci vengono proposti? L'azione più banale sarebbe ripetere l'algoritmo di nuovo sul training set, esteso con il nuovo punto. Nel caso volessimo utilizzare un metodo più veloce, possiamo utilizzare l'estensione *Out of Sample* dello Spectral Clustering (vedi [VL04]).

Dato il training set e la matrice dei pesi W definiamo una matrice dei pesi normalizzata:

$$\tilde{K}(x, y) = \frac{K(x, y)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} K(x, x_i)}\sqrt{\frac{1}{n}\sum_{i=1}^{n} K(y, x_i)}} \tag{2.23}$$

Utilizziamo questa matrice per classificare nuovi esempi, se t_j è il j-esimo autovettore:

$$f_j(x) = \frac{1}{1+\lambda}\sum_{i=1}^{n} t_{j,i}\tilde{K}(x, x_i) \tag{2.24}$$

Se un nuovo punto $x \in \mathbb{R}^n$ deve essere incorporato in uno spazio k-dimensionale, la sua immagine sarà $(f_2(x), \ldots, f_{k+1}(x))$. Utilizzando l'immagine possiamo osservare la sua vicinanza ad un cluster piuttosto che ad un altro. Nel caso bipartizionale è sufficiente guardare il segno di $f_2(x)$.

[5]Dall'inglese: *al di fuori degli esempi*

Capitolo 3

Lo Spectral Clustering: Applicazioni

Le tecniche di Clustering hanno un ampio ventaglio di applicazioni in campi molto diversi tra loro, dall'ambito biomedicale al marketing. Per poterle applicare nella pratica vogliamo verificare, con degli esperimenti, che i risultati siano conformi a quello che ci si aspetta.

Ricordiamo, ad esempio, che la soluzione del taglio normalizzato attraverso il Laplaciano normalizzato è un'approssimazione di quella vera, in pratica non si ha garanzia del suo corretto funzionamento. Per contro grazie a questa approssimazione si può costruire un algoritmo che risolva il problema in tempo polinomiale.

Vedremo il testing dello Spectral Clustering su una serie di esperimenti simulati; in questi esperimenti si verifica se, oltre al risultato del partizionamento, vi siano altri indicatori che rivelino la buona qualità dei risultati.

Gli esperimenti simulati sono costituiti da training set semplici che nel contempo presentano difficoltà per molte tecniche allo stato dell'arte; studiandoli incontreremo alcune problematiche incontrate dagli algoritmi partizionali, come il K-Means, e superate dallo Spectral Clustering.

In seguito mostrerò i risultati dello Spectral Clustering in casi pratici di lavoro: la segmentazione di immagini e la telesorveglianza.

3.1 L'algoritmo

Abbiamo visto nella sezione 2.5 un algoritmo generico per partizionare un insieme di dati attraverso lo Spectral Clustering, ora lo riesporrò più dettagliatamente per il suo utilizzo pratico negli esperimenti.

Innanzitutto utilizzo come misura di similarità la funzione gaussiana $w_{i,j} = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$. Questa misura può portare a grafi sparsi o sconnessi allo stesso modo di grafi completi, a seconda del training set e del valore del parametro σ.

Nel caso in cui il grafo sia abbastanza connesso il calcolo degli autovettori non può usufruire dell'accelerazione del metodo di Lanczos vista nella sezione 2.5[1]; comunque i casi simulati che tratterò sono semplici e il numero di dati piccolo, inoltre gli esperimenti non devono fornire risultati in tempo reale.

In conclusione nell'implementazione dell'algoritmo non ho dato importanza all'efficienza dell'esecuzione.

[1]Vale la pena notare che ottenere una matrice sparsa dai dati non comporta solo un'accelerazione dell'algoritmo, ma anche un notevole risparmio di memoria

3.1.1 Normalizzato o non normalizzato?

Una delle domande fondamentali nello Spectral Clustering è quale definizione di Laplaciano utilizzare per calcolare gli autovettori. Per rispondere a questa domanda bisogna capire la differenza pratica tra le definizioni date nella sezione 2.4. Ricordiamo che il taglio normalizzato assicura di

minimizzare il taglio fra due partizioni

massimizzare il *volume* di ogni partizione

Si può mostrare che anche gli autovettori del Laplaciano non normalizzato possono partizionare il grafo soddisfando dei requisiti simili:

minimizzare il taglio fra due partizioni

massimizzare la *cardinalità* di ogni partizione

Come mostrato in [VL06] i due approcci forniscono risultati molto simili nel caso in cui i gradi dei vertici del grafo siano simili tra loro. Nel caso in cui i gradi dei vertici assumano valori abbastanza diversi i due approcci sono distinti, in particolare il Laplaciano non normalizzato dà importanza a minimizzare il taglio e a massimizzare la *cardinalità* per partizione, piuttosto che il volume. I due concetti, cardinalità e volume, sono scorrelati tra loro, il primo si basa sul numero di vertici, il secondo sulle connessioni tra i vertici.

Dal punto di vista dei grafi il volume è una proprietà più interessante che indica la connettività interna di una partizione; questa è una prima motivazione per preferire la definizione di Laplaciano normalizzato.

Oltre a questo argomento in [VL04] viene mostrata un'altra differenza riguardante i due approcci: sotto alcune condizioni il Laplaciano normalizzato porta a risultati statisticamente consistenti, al contrario del Laplaciano non-Normalizzato. Questo significa che se la cardinalità del training set tende a infinito i risultati del primo convergono ad una partizione, ma questo non vale necessariamente nel secondo caso.

3.1.2 La scelta di sigma

Il parametro σ regola la varianza della funzione gaussiana. Per stimarlo si possono eseguire considerazioni sulla distribuzione dei dati come, ad esempio, calcolare la distanza media tra essi, oppure eseguire dei test. In effetti si vede sperimentalmente che spesso è sufficiente scegliere il suo corretto ordine di grandezza.

Eseguendo dei test ho osservato che il secondo autovalorecresce al crescere di σ, vedi figura (3.1)

Questa osservazione può essere sfruttata per controllare la diminuzione del valore del taglio normalizzato. Dal momento che, come visto nella sezione 2.4.1, $Ncut \leq \sqrt{2\lambda_1}$, diminuire il primo autovalore implica diminuire il valore del taglio normalizzato.

Si potrebbe osservare che se $\lambda_1 = 0$ avremmo un taglio perfetto, ma in questo caso variamo σ per controllare il valore di ncut e costringendolo a zero rischiamo di ottenere una disconnessione di un solo vertice, eventualmente perdendo il taglio ottimale.

Allo stesso modo si può ribattere che se $\lambda_1 \simeq 0$ potrebbe esserci effettivamente un taglio vantaggioso, ma di un singolo vertice praticamente sconnesso dal grafo. In questo caso ricordiamo che minimizzare il taglio normalizzato equivale a massimizzare il volume delle partizioni e che quindi il taglio di pochi vertici rimane comunque scoraggiato.

Queste considerazioni mi hanno portato a stimare automaticamente il parametro σ, in questa fase il parametro della gaussiana σ varia in modo tale che $10^{-3} \leq \lambda_1 \leq 2 * 10^{-3}$. L'algoritmo è semplice:

input: min e max

1. $\alpha = 1$

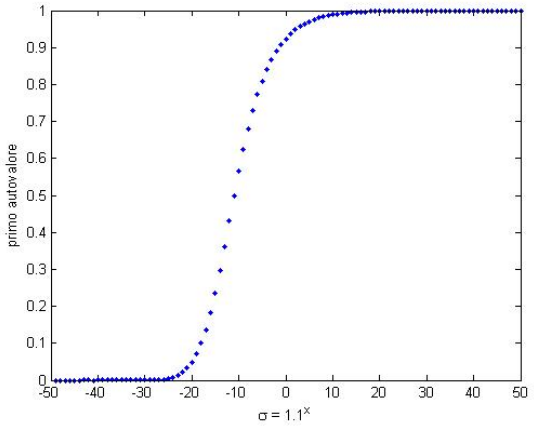

Figura 3.1: Al crescere di σ cresce λ_1

2. λ_1=primo_autovalore(α)
3. while ($\lambda_1 < \min$)
   - $\alpha := \alpha * 2$
   - λ_1=primo_autovalore(α)
4. $\alpha = \alpha/2$
5. $\beta = \alpha/2$
6. while (*true*)
   - λ_1=primo_autovalore($\alpha + \beta$)
   - if $\min \leq \lambda_1 \leq \max$
     - return $\alpha + \beta$
   - else if $\lambda_1 \geq \max$
     - $\beta := \beta/2$
   - else
     - $\alpha := \alpha + \beta$
     - $\beta := \beta/2$

output: σ

dove *primo_autovalore* calcola il primo autovalore del Laplaciano normalizzato prendendo σ in input, vedi sezione 3.1.3.

Il numero di passi eseguiti è logaritmico rispetto ad un ipotetico valore massimo di σ, fissato dato il training set. Questa ricerca quindi ha complessità $O(n^3 \log(n))$.

Stimare in questo modo la σ aiuta a eseguire un buon primo taglio, ma comunque resta il fatto che è difficile quantificare la bontà di questo taglio; nell'algoritmo che vedremo nel caso della telesorveglianza nella sezione 3.3.3, il grafo viene bipartizionato ricorsivamente; fissato σ, per accettare o meno la bipartizione, ho stabilito a priori una soglia. Scegliere automaticamente il valore di questa soglia è un suggerimento per un lavoro futuro.

3.1.3 Lo pseudocodice

Stabilito il criterio per la ricerca di σ vediamo lo pseudocodice dell'algoritmo.

input: il training set

1. Ricerca del parametro σ, come nella sezione 3.1.2
2. Costruire una matrice di similarità $W \in \mathbb{R}^{n,n}$.
3. Calcolare il Laplaciano normalizzato $\mathcal{L}$
4. Calcolare gli autovettori $t_1, \ldots, t_n$ di $\mathcal{L}$
5. Utilizzare il secondo autovettore per bipartizionare il grafo.

output: Clusters A_1, A_2 con $A_j = \{j | y_j \in C_i\}$

Nel caso degli esperimenti simulati i dati vengono semplicemente bipartizionati. Per quanto riguarda la segmentazione di immagini e l'esperimento sulla telesorveglianza la bipartizione avviene ricorsivamente; il criterio di stop è il numero di dati nella partizione o il valore del taglio normalizzato. La scelta di questi valori utilizza l'informazione a priori che si ha sul problema e viene verificata osservando i risultati.

3.1.4 Misure di qualità

Compiendo gli esperimenti desideriamo avere delle osservazioni che ci informino sulla qualità del risultato. La prima misura a cui si può pensare è il valore del taglio normalizzato. In letteratura non ci sono analisi di questo tipo e nonostante possa assumere valori in un intervallo limitato non vi sembrano essere motivi per cui, dati due grafi differenti, uno stesso valore possa portare a partizioni ugualmente valide.

Una seconda misura di qualità, esplorata in letteratura ([MMeX97]), è la forma degli autovettori. Abbiamo già avuto un'intuizione di questo nella sezione 2.4.2, se la forma dell'autovettore è costante a tratti allora la bipartizione sarà di buona qualità, al contrario nel caso in cui sia 'continuo' , la bipartizione è meno affidabile.

Una giustificazione intuitiva di questo fatto si ha osservando l'equazione 3.1 che indica la relazione, secondo il quoziente di Rayleigh, fra λ_1 e il primo autovettore.

$$\lambda_1 = \inf_{f \perp D\mathbb{1}} \frac{\sum_{i,j=1}^{n} (f(i) - f(j))^2 w(i,j)}{\sum_{i \in V} f^2(i) d_i} \tag{3.1}$$

nel caso in cui l'autovettore f sia a costante a tratti il numeratore, spesso chiamato somma di Dirichlet del grafo, assume valori piccoli. Infatti all'interno di ogni partizione i valori si annullano, perché costanti, e il valore che rimane dipende da quanto è piccolo il taglio, l'insieme di connessioni tra le due partizioni. Viceversa un autovettore 'continuo' non fornisce questa intuizione.

In molti casi semplici, il grafo è composto da più di due cluster abbastanza sconnessi tra loro, in alcuni di questi casi vedremo che l'autovettore assumerà tanti valori costanti differenti quanti sono queste parti voluminose. Purtroppo questo avviene in condizioni abbastanza particolari e non sembra che sia estendibile a casi non simulati; rimane il fatto che in alcuni casi si potrebbe multipartizionare il grafo usando soltanto il secondo autovettore.

La terza misura di qualità, di gran lunga la più informativa, è lo spettro stesso del grafo, in figura vediamo le quattro tipologie principali che si possono incontrare

Vediamo le peculiarità di ogni tipologia:

clique Si può dimostrare che il grafo in figura 3.2 è una clique in quanto ha un solo autovalore uguale a zero e l'autovalore uno ha molteplicità (n-1) ([F.97]). Questo può succedere se ad esempio si usa un valore di σ troppo elevato.

abbastanza connesso Sotto certi punti di vista questo è la migliore situazione per operare delle partizioni, il *gap* è grande e quindi ci sono diversi autovettori che indicano delle possibili partizioni, in questo caso due (figura 3.3).

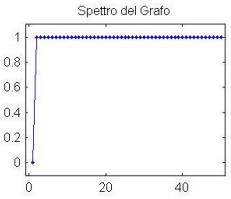

Figura 3.2: Una clique

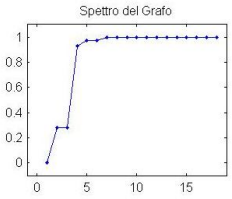

Figura 3.3: Un grafo con alcuni buoni tagli

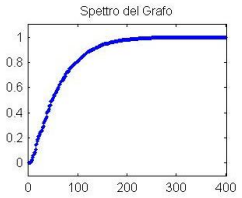

Figura 3.4: Un grafo dove solo il primo taglio è affidabile

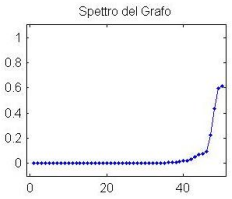

Figura 3.5: La molteplicità dell'autovalore zero è pari al numero di componenti sconnesse

poco connesso Gli autovalori hanno un aspetto 'continuo' ma il secondo autovalore è comunque diverso da zero. Il metodo del *gap* non assicura buoni risultati ma il fatto che il secondo autovalore sia piccolo ci assicura che il taglio normalizzato dovuto al secondo autovettore sia affidabile(figura 3.4).

sconnesso Se il grafo è sconnesso il Laplaciano è una matrice a blocchi e gli autovettori hanno una forma diversa da quelli per i grafi connessi: il primo autovettore non è costante e ogni autovettore relativo ad un autovalore nullo è un vettore indicatore per quel blocco (figura 3.5).

Si può pensare di partizionare semplicemente il grafo nelle sue componenti connesse.

La terza tipologia, un grafo poco connesso, è abbastanza comune negli esperimenti che presento, dal momento che, utilizzando la tecnica che ho proposto, di solito σ rimane piccolo; di conseguenza la varianza diminuisce e il grafo tende a perdere connessioni.

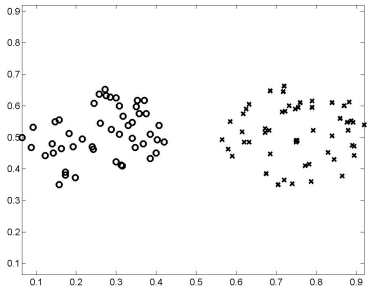

Figura 3.6: Un training set

Oltre a queste tre misure c'é una quarta possibilità che non è menzionata nella

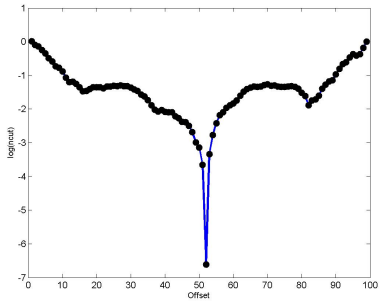

Figura 3.7: La variazione del taglio normalizzato(in scala logaritmica)

letteratura dello Spectral Clustering: la variazione del valore di Ncut rispetto all'offset scelto per suddividere il secondo autovettore.

Il procedimento per visualizzare questa variazione è semplice: gli n elementi del secondo autovettore vengono ordinati, quindi si scorre la lista degli n valori scegliendo come offset per la partizione del grafo il valore n-esimo e si calcola il corrispondente valore del taglio normalizzato. Data la lista dei valori si ottiene il grafico. Per avere un esempio vediamo in figura 3.6 un training set linearmente separabile ed in figura 3.7 la corrispondente variazione del valore del taglio normalizzato; si può pensare le n partizioni del grafo avvengano, in questo caso, in maniera sequenziale da sinistra verso destra, e che il minimo corrisponda alla separazione centrale.

Ho applicato questa tecnica in alcune simulazioni e si potranno osservare delle cose interessanti al riguardo, comunque il suo studio per migliorare le prestazioni del partizionamento fa parte di una ricerca futura.

3.2 Simulazioni

In questa sezione affronteremo il problema del clustering all'interno di un insieme di casi simulati ad hoc. Questi casi sono esempi semplici ma utili, perché sintetizzano problematiche che si possono incontrare nei casi pratici.

In queste simulazioni vedremo training set creati a partire da distribuzioni uniformi sul piano; l'algoritmo di Spectral Clustering non conosce queste distribuzioni e deve ricavarle proprio dal training set. Il problema sembra facile per l'uomo ma non lo è altrettanto per il calcolatore.

Ogni simulazione avrà due aspetti distinti: il primo è la partizione dei dati del training set nei cluster corretti, il secondo è generare una funzione classificatrice che associ al cluster corretto un dato futuro, questa funzione verrà in seguito indicata come funzione out of sample.

Oltre alla partizione siamo interessati ad analizzare i risultati con le misure di qualità descritte in precedenza (vedi sezione 3.1.4), per verificare se sono informative o meno.

3.2.1 Due partizioni sferiche

In questo esempio, figura 3.8, usiamo una distribuzione dei dati che in letteratura viene chiamata 'dumbbell'[2], due cerchi sul piano linearmente separabili. Estraiamo da questa distribuzione cento dati e proviamo a partizionarli usando l'algoritmo di

[2]In inglese: manubrio per sollevamento pesi

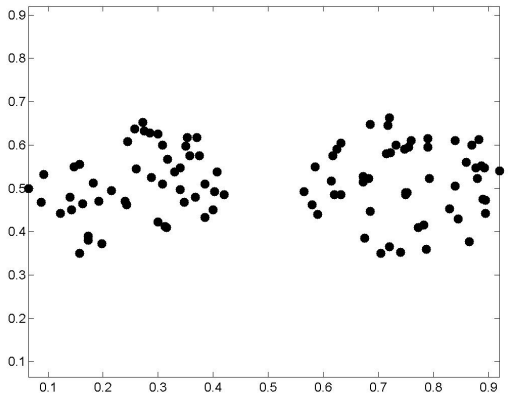

Figura 3.8: Insieme dei dati da partizionare

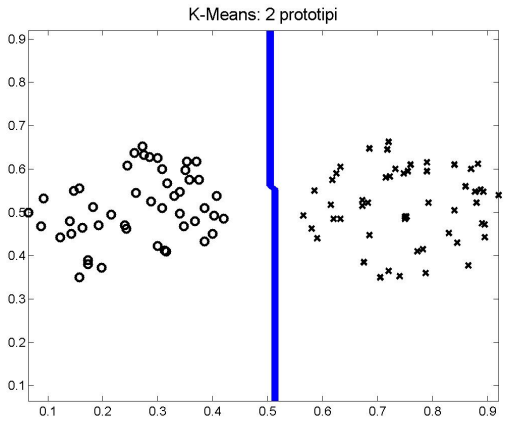

Figura 3.9: Partizionamento con K-Means

K-Means, fissando a due il numero di prototipi, e l'algoritmo di Spectral Clustering per bipartizionare i dati.

Confronto dei risultati Nelle figure 3.9 e 3.10 vediamo i partizionamenti ottenuti rispettivamente con l'algoritmo di K-Means e Spectral Clustering: entrambi hanno partizionato i dati allo stesso modo e correttamente.

La linea blu nelle figure rappresenta la funzione che classificherebbe eventuali dati futuri; sebbene le due funzioni sembrino simili è importante osservare che la funzione legata al K-Means è una retta, mentre quella legata allo Spectral Clustering non lo è. La differenza è che il K-Means è in grado di costruire solo funzioni di separazione lineare, mentre lo Spectral Clustering non ha questo vincolo e si adatta, generalizzando, ai dati.

Lo spettro e gli autovalori Approfondiamo il risultato dello Spectral Clustering; in tabella 3.1 vediamo i diversi valori assunti dalle variabili in gioco. L'osservazione più

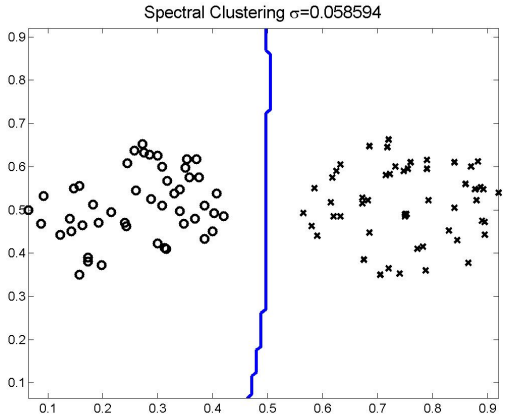

Figura 3.10: Partizionamento con Spectral Clustering

interessante è la vicinanza tra λ_1 e il valore del taglio normalizzato, questa vicinanza implica che il taglio ha fornito una buona partizione; per capire meglio questo fatto consideriamo che se i due valori fossero uguali avremmo ottenuto la reale minimizzazione del taglio normalizzato.

Tabella 3.1: Spectral Clustering

valori	
σ	0.05859
intervallo lambda	[0.001,0.002]
λ_1	0.00114
taglio normalizzato	0.00135
upper bound di λ_1	0.04783
gap	0.1035

Abbiamo altri due indizi chiave che indicano la buona qualità della partizione; come vediamo in figura 3.11 lo spettro del grafo ha un *gap* elevato e il secondo autovettore è a tratti costanti.

Dal momento che gli autovalori hanno un buon *gap*, si può pensare di utilizzarne più di uno, per operare un partizionamento multiclasse. In figura 3.12 vediamo come questo succeda: utilizzando il secondo e il terzo autovettore possiamo dividere i dati in quattro partizioni, e sono evidentemente partizioni corrette! Le linee blu indicano la funzione separatrice out of sample che in questo caso divide lo spazio di input in quattro zone.

Per capire meglio come vengano individuate queste quattro zone vediamo in figura 3.13 la riduzione della dimensionalità che si ottiene utilizzando il secondo, il terzo ed il quarto autovettore; i dati sono sparsi e si dividono ognuno in due partizioni.

3.2.2 Dati a mezzaluna

La distribuzione dei dati a mezzaluna che possiamo vedere nella figura 3.14 è uno dei casi di test più utilizzati nel clustering; in questo caso cento dati sono stati estratti da due distribuzioni uniformi a forma di a mezzaluna nello spazio di input, queste distribuzioni sono visibili in figura 3.15.

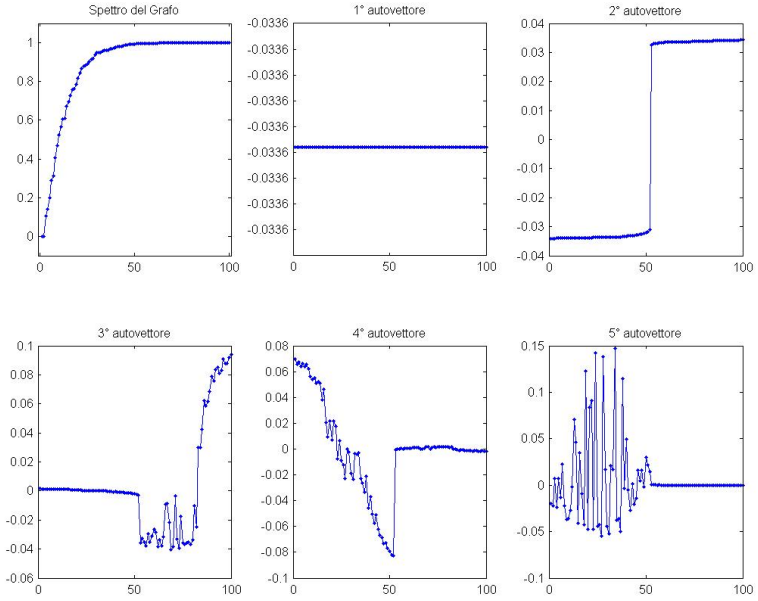

Figura 3.11: Spettro e Autovettori del grafo

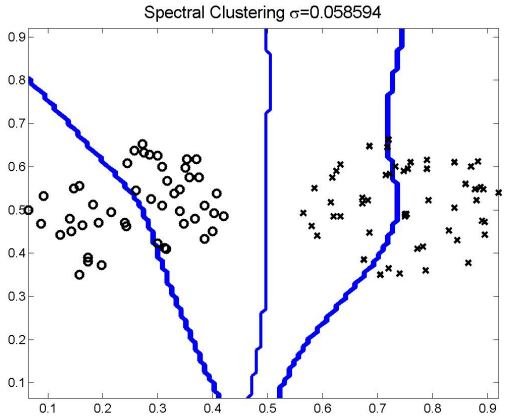

Figura 3.12: Partizione multiclasse

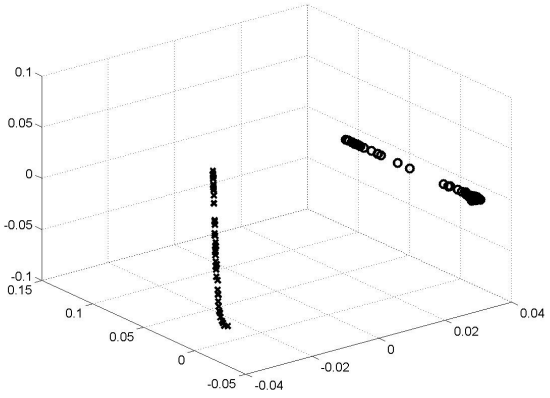

Figura 3.13: La riduzione della dimensionalità

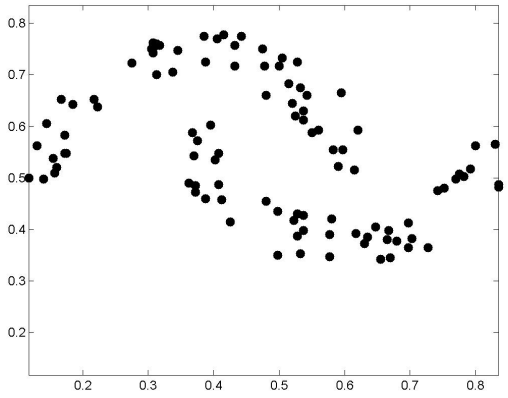

Figura 3.14: Insieme dei dati da partizionare

Figura 3.15: Distribuzione uniforme sottostante ai dati

Confronto dei risultati In figura 3.14 vediamo il training set. Si osserva che il numero di dati è insufficiente a coprire le distribuzioni, addirittura le mezzelune sembrano poter essere spezzate al loro interno; nonostante questo non solo lo Spectral Clustering è in grado di dividere correttamente i dati, ma la funzione out of sample è ben generalizzata.

Nelle figure 3.16 e 3.17 scopriamo i partizionamenti ottenuti rispettivamente con l'algoritmo di K-Means e Spectral Clustering: il primo ha commesso un errore del 21% mentre il secondo ha partizionato i dati correttamente.

La partizione lineare del K-Means non ha un errore grande, ma il suo risultato non può migliorare dal momento che il modello sottostante ai prototipi è una distribuzione gaussiana a campana, ben diversa dalla distribuzione a mezzaluna.

Per quanto riguarda lo Spectral Clustering guardiamo come lo spettro e gli autovalori ci indichino cosa stia succedendo.

Lo spettro e gli autovalori Lo spettro del grafo in figura 3.18 ha una forma continua, ma con un *gap* interessante (vedi tabella 3.2) intorno al dodicesimo autovalore, anche in questo caso quindi possiamo dedurre una buona qualità del taglio.

Tabella 3.2: Spectral Clustering

valori	
σ	0.03906
intervallo lambda	[0.001,0.002]
λ_1	0.00168
taglio normalizzato	0.00297
upper bound di λ_1	0.05791
gap	0.1555

La qualità del taglio è confermata dal fatto che il secondo autovettore sia costante

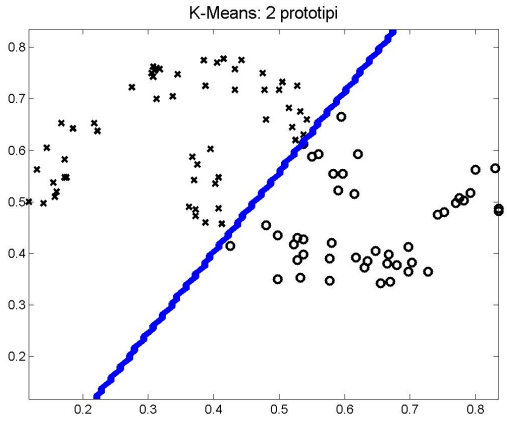

Figura 3.16: Partizionamento con K-Means

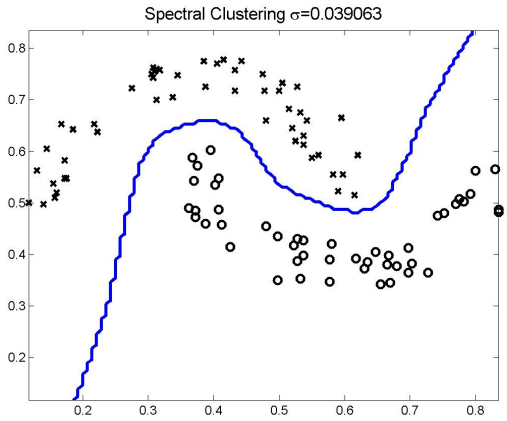

Figura 3.17: Partizionamento con Spectral Clustering

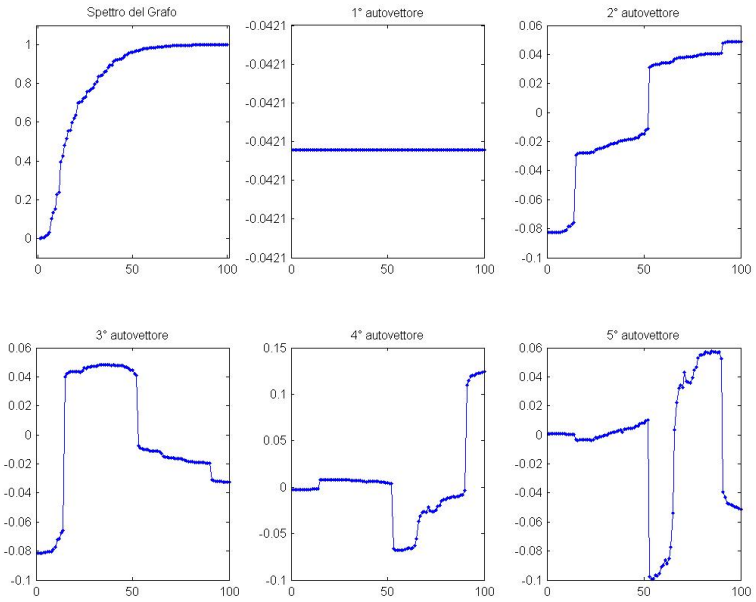

Figura 3.18: Spettro e Autovettori del grafo

a tratti, con una particolarità in più: i pezzi sono quattro. Mentre i valori positivi e negativi dell'autovettore corrispondono alle due mezzelune, all'interno di queste due mezzelune possiamo ritrovare un ulteriore bipartizione che spezza a loro volta le mezzelune, questa partizione in quattro parti è evidenziata dalla funzione out of sample ottenuta considerando il secondo, il terzo ed il quarto autovalore, la partizione in quattro parti è visibile nella figura 3.19.

Ritroviamo questa stessa partizione nello spazio a dimensionalità ridotta[3] cerchiata nella figura 3.20.

Un'ultima osservazione è che il secondo autovettore non è il solo ad essere costante a tratti, ma anche il terzo, il quarto ed in parte anche il quinto; questi tratti sono conformi alla divisione del secondo autovettore e rendono i cluster, nello spazio a dimensionalità ridotta, ben distinguibili.

3.2.3 Dati simmetrici

Negli esempi precedenti abbiamo cercato conferme della qualità del clustering da varie osservazioni sullo spettro del grafo e sulla forma degli autovettori; ora procediamo in maniera inversa generando un grafo simmetrico della forma di figura 3.21. Vogliamo controllare che effettivamente in questo caso semplice si verifichino le condizioni che ci aspettiamo.

Il risultato Data la semplicità del training set non ci interessa un confronto con K-Means, che tralasceremo. Nelle figure 3.22 e 3.23 vediamo i partizionamenti ottenuti

[3]In questo caso la dimensionalità aumenta perché passiamo da $\mathbb{R}^2$ a $\mathbb{R}^3$, ma dal momento che i dati sono in forma più 'semplice', si pensa comunque che la loro dimensionalità, intesa come quantità di informazione posseduta, sia in effetti ridotta.

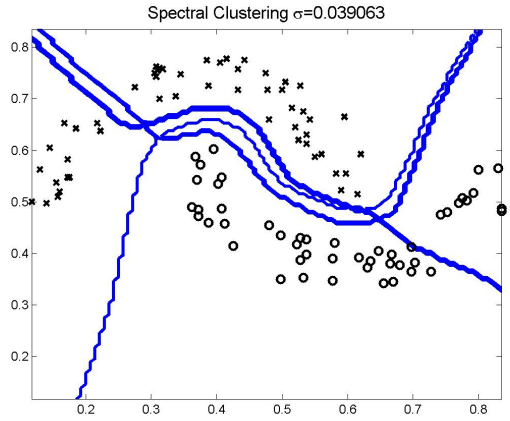

Figura 3.19: Partizione multiclasse

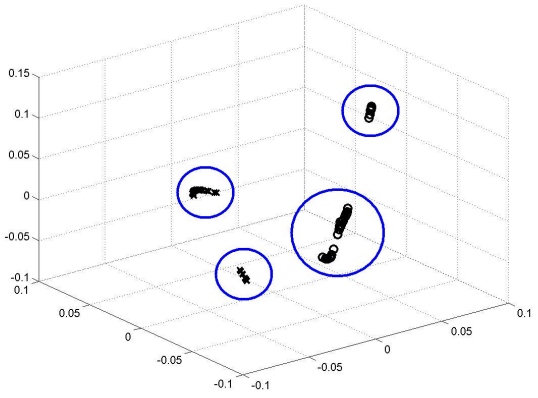

Figura 3.20: La riduzione della dimensionalità1

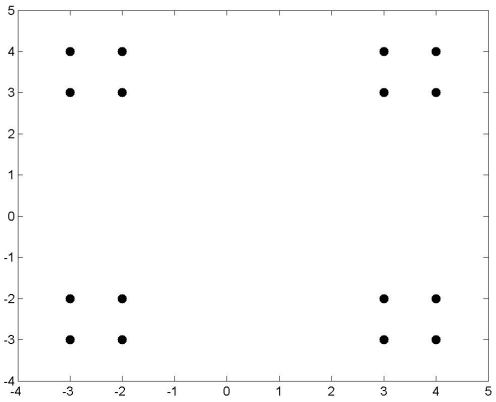

Figura 3.21: Insieme dei dati da partizionare

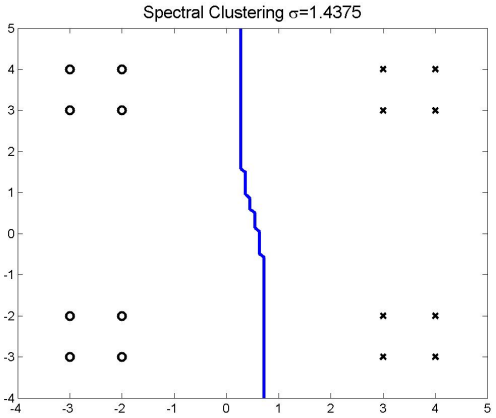

Figura 3.22: Partizionamento con Spectral Clustering

utilizzando solo un autovettore o i primi due; curiosamente la funzione out of sample ottenuta non è precisamente una retta, d'altra parte, nel calcolo degli autovalori più piccoli di una matrice simmetrica c'é un po' di instabilità, a cui si può imputare questa imprecisione.

Lo spettro e gli autovalori Lo spettro di questo grafo, figura 3.24, è ricco di informazioni; innanzitutto ci sono tre autovalori quasi uguali, questi corrispondono alle tre bipartizioni ottimali possibili dei quadrati. Per essere più precisi le partizioni ottimali sono due ed una è subottimale, ma dal momento che i pesi della connessioni sono bassi, la subottimale che accomuna quadrati opposti rispetto alla diagonale ha un valore di taglio normalizzato molto simile a quello ottimale. Gli autovettori relativi a questi autovalori non specificano solo le bipartizioni ottimali, ma anzi i primi due sono divisi in quattro tratti costanti, consentendo, ognuno di essi, una partizione multiclasse.

Nella tabella 3.3 si può constatare l'elevato valore del *gap*; invece acciglia il fatto che il taglio normalizzato non sia uguale al primo autovalore, come previsto dall'equazione 2.16. Per capire questa differenza bisogna considerare che tagliare in quattro parti

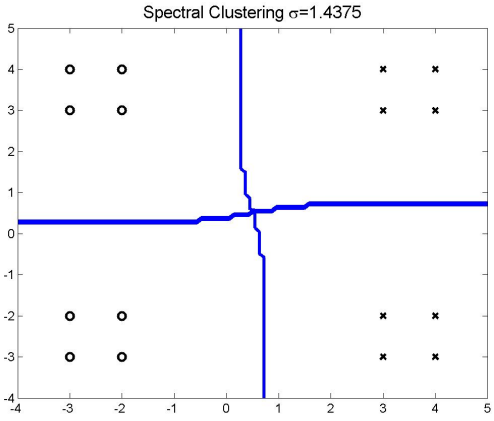

Figura 3.23: Partizionamento con Spectral Clustering multiclasse

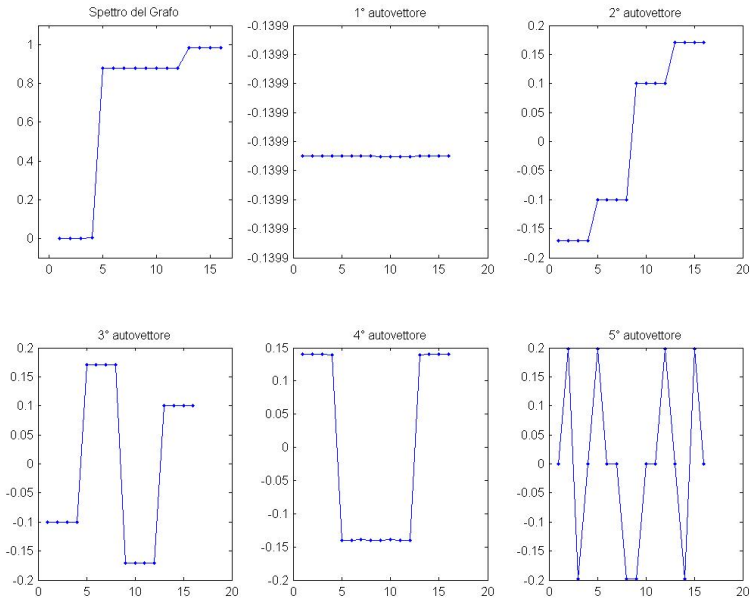

Figura 3.24: Spettro e Autovettori del grafo

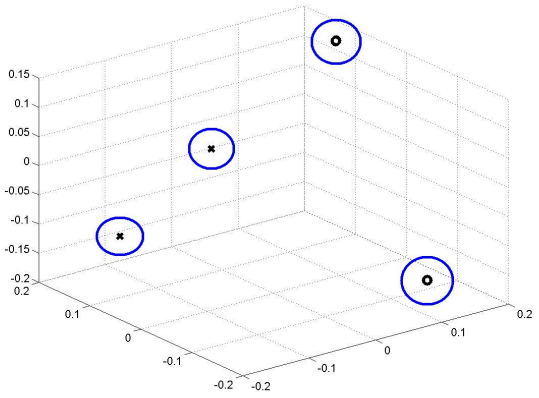

Figura 3.25: La riduzione della dimensionalità

questo grafo ha un taglio normalizzato con un valore inferiore al taglio dovuto alla bipartizione, in questa ottica il secondo autovettore ha una forma conforme ad una nuova definizione dell'equazione 2.16 che contempli il multipartizionamento del grafo.

Tabella 3.3: Spectral Clustering

valori	
σ	1.43750
intervallo lambda	[0.001,0.002]
λ_1	0.00151
taglio normalizzato	0.00220
upper bound di λ_1	0.05491
gap	0.87611

I primi tre autovettori sono costanti a tratti in maniera consistente, gli stessi dati assumono la stessa costante in ogni autovettore; questo conduce ad una riduzione di dimensionalità ideale, i dati dei quattro cluster si sovrappongono nello spazio a dimensionalità ridotta (vedi figura 3.25).

Un'ultima nota, la variazione del valore del taglio normalizzato ha tre minimi locali, in corrispondenza di tre possibili suddivisioni dei dati con il secondo autovettore; in pratica il minimo assoluto che divide in due il grafo non è molto distante dai minimi locali, che dividerebbero tre quadrati contro uno. La spiegazione è che avendo forzato λ_1 ad assumere un valore piccolo il Laplaciano normalizzato è molto simile ad una matrice a blocchi in cui le componenti sono connesse tra loro e poco connesse alle altre.

3.2.4 Dati a corona con outlier

Nel caso in cui alcuni dati siano outlier è difficile trovare il corretto valore di σ per operare delle buone partizioni; in figura 3.27 vediamo dei dati a corona con alcuni dati lontani, che sono gli outlier.

Il risultato Tralasciando il confronto con K-Means vediamo in figura 3.28 che il risultato del partizionamento è corretto, ma la funzione out of sample, relativa al

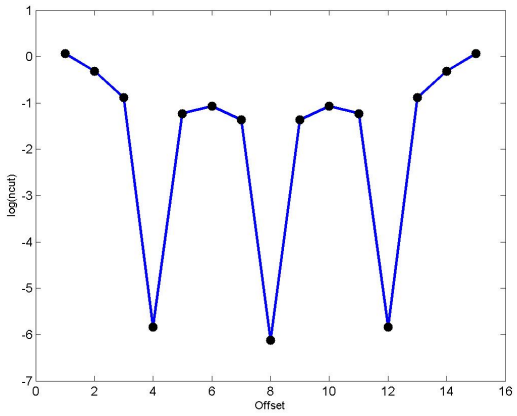

Figura 3.26: La variazione del taglio normalizzato

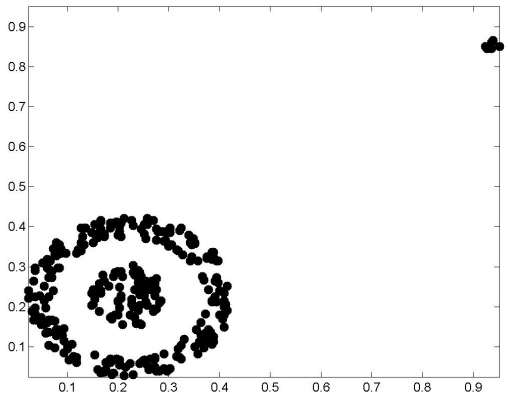

Figura 3.27: Insieme dei dati da partizionare

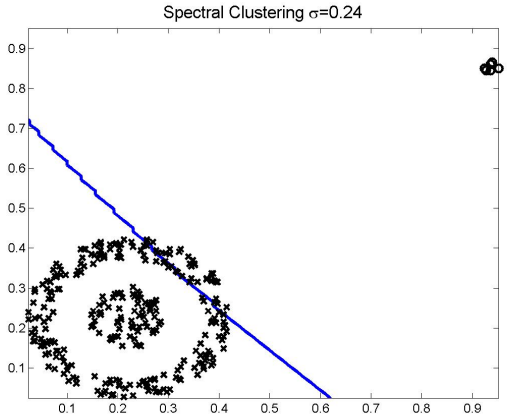

Figura 3.28: Partizionamento con Spectral Clustering

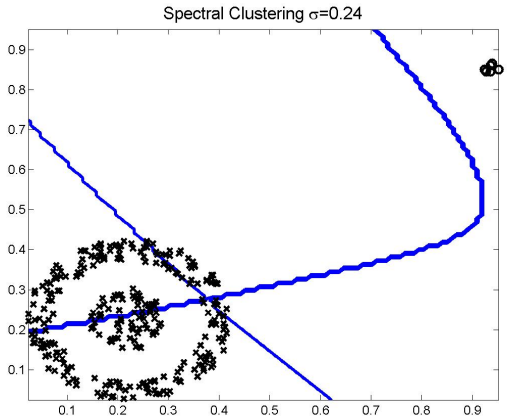

Figura 3.29: Partizionamento con Spectral Clustering multiclasse

secondo autovettore, non è conforme a questa partizione. Il problema è di carattere numerico, e lo si può notare dalla forma degli autovettori.

Lo spettro e gli autovalori Lo spettro del grafo rivela che esso è praticamente una clique, quasi tutti gli autovalori sono vicini uno; un unico autovalore piccolo corrisponde all'autovettore che partiziona via gli outlier.

Anche se il partizionamento è corretto ci sono diverse considerazioni in questo caso: è importante accorgersi durante l'esecuzione di un algoritmo di clustering della presenza degli outlier, questi condizionano negativamente la scelta di σ che potrebbe non essere ottimale, inoltre non si possono utilizzare gli autovettori dopo il secondo in quanto, come rivelano i corrispettivi autovalori, non operano dei buoni tagli.

Nella tabella 3.4 si può constatare l'elevato valore del *gap*; in effetti per accorgersi della presenza di outlier non è utile controllare il valore del *gap* o del taglio. Negli esperimenti relativi alla telesorveglianza ho stabilito una soglia percentuale al numero di dati partizionati, se questo numero è inferiore a questa soglia la partizione avviene solo fra gli outlier e il resto dei dati, senza utilizzare gli autovettori.

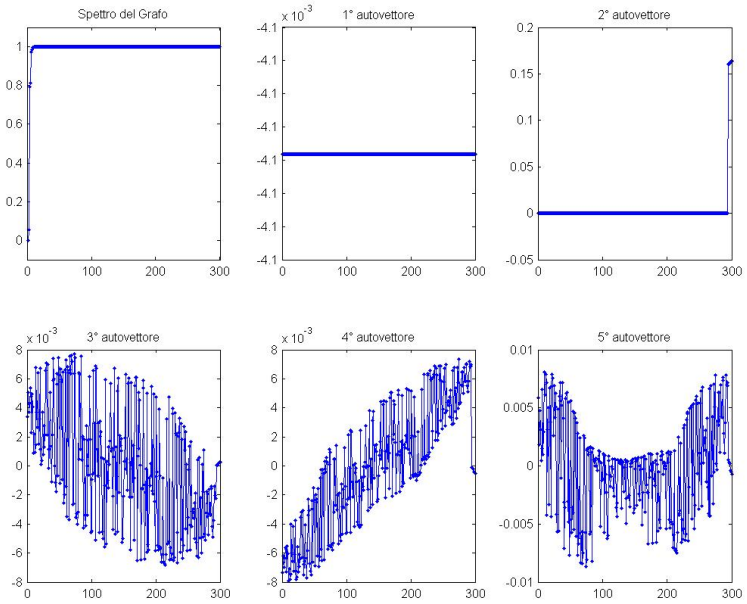

Figura 3.30: Spettro e Autovettori del grafo

Tabella 3.4: Spectral Clustering

valori	
σ	0.24000
intervallo lambda	[0.001,0.002]
λ_1	0.05568
taglio normalizzato	0.06617
upper bound di λ_1	0.33370
gap	0.73701

Purtroppo dopo aver rimosso gli outlier la σ scelta non è più adatta a partizionare i dati a corona; rieseguendo l'algoritmo della sezione 3.1.2 si ottiene un nuovo valore: $\sigma = 0.034924$, un valore molto differente dal precedente. In figura 3.31 vediamo il corretto partizionamento ottenuto con questo valore.

3.2.5 Una scritta in corsivo

Gli esempi precedenti sono costituiti da distribuzioni bidimensionali sul piano, ora consideriamo una distribuzione unidimensionale e connessa: una scritta in corsivo. Il risultato che vogliamo raggiungere è segmentare le lettere a partire da un loro campionamento.

L'utilizzo dello Spectral Clustering in questo caso è giustificato, infatti ci interessano tagli piccoli, fra le lettere, che massimizzino i volumi, la lettera stessa; questa è proprio l'operazione compiuta dal taglio normalizzato.

La scritta in questione è visualizzata nella figura 3.32, da essa estraiamo trecento dati che la rappresentino.

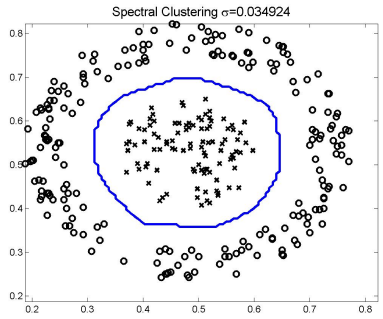

Figura 3.31: Dati a corona senza outlier

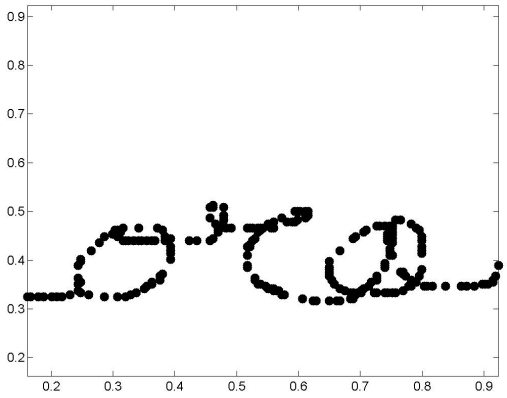

Figura 3.32: Insieme dei dati da partizionare

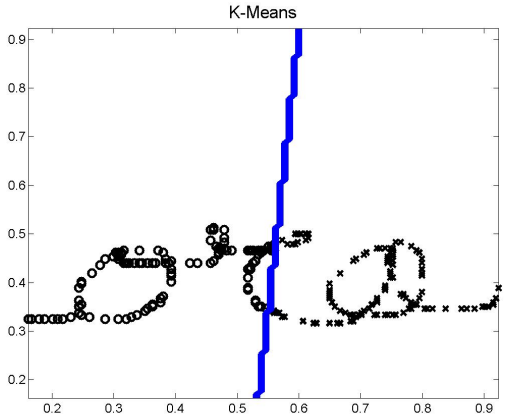

Figura 3.33: Partizionamento con K-Means

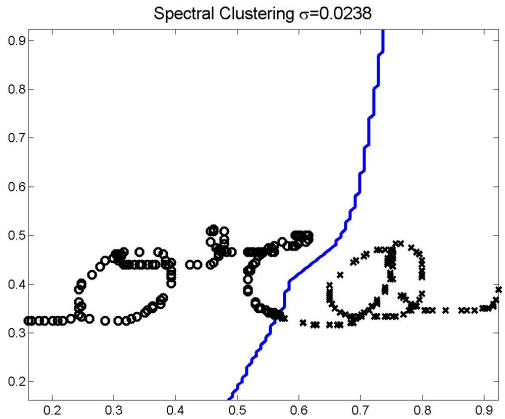

Figura 3.34: Partizionamento con Spectral Clustering

Confronto dei risultati Nelle figure 3.33 e 3.34 vediamo i partizionamenti ottenuti rispettivamente con l'algoritmo di K-Means e Spectral Clustering: entrambi hanno partizionato la scritta in 'or' e in 'a', con una fondamentale differenza, il K-Means non è stato in grado di seguire il tratto stilografico, partizionando in maniera errata una parte della lettera 'r', al contrario lo Spectral Clustering ha operato una partizione non lineare. Le funzioni out of sample ottenute sottolineano questa differenza e l'adattabilità ai dati dello Spectral Clustering.

Lo spettro e gli autovalori Dalla tabella 3.5 vediamo che non solo il *gap* ha un buon valore, ma il valore del taglio normalizzato è molto vicino a λ_1, sintomo che il partizionamento è di buona qualità. Questo però non implica che il clustering esegua quello che abbiamo in mente, dividere le lettere in corsivo, indica semplicemente che il valore di σ trovato è in grado di generare un grafo dove il taglio normalizzato è affidabile.

Se si vuole utilizzare lo Spectral Clustering in un'applicazione reale bisogna considerare se il clustering che abbiamo in mente può non essere ottenuto dal taglio normaliz-

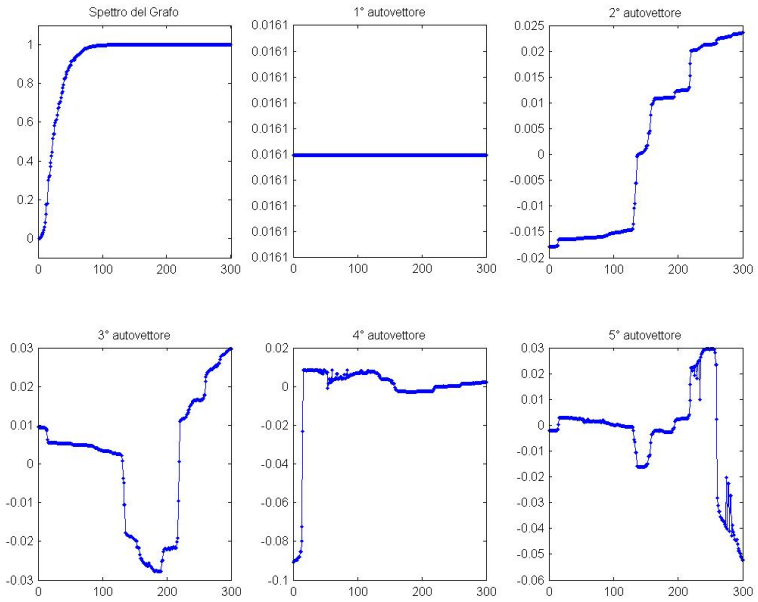

Figura 3.35: Spettro e Autovettori del grafo

zato; in questo caso il riscontro è stato positivo, la giustificazione vista nell'introduzione 3.2.5 è stata confermata dai risultati.

Tabella 3.5: Spectral Clustering

valori	
σ	0.02380
intervallo lambda	[0.001,0.002]
λ_1	0.00108
taglio normalizzato	0.02256
upper bound di λ_1	0.04653
gap	0.12195

Abbiamo altri due indizi chiave che indicano la buona qualità della partizione; come vediamo in tabella 3.5 lo spettro del grafo ha un *gap* elevato e il secondo autovettore è costante a tratti.

Il secondo autovettore nella figura 3.35 è costante a tratti, utilizzabile quindi per una partizione multipla, osserviamo però che i pezzi sono più delle tre lettere della scritta. Nella pratica se si volessero segmentare scritte in corsivo con lo Spectral Clustering bisognerebbe utilizzare altre assunzioni a priori, come ad esempio il volume minimo di un cluster; se limitassimo questo volume con un valore allora saremmo in gradi di decidere se accettare o meno una partizione.

Non ci sono molte osservazioni da fare sulla riduzione della dimensionalità in figura 3.37, se non che se applicassimo l'algoritmo di K-Means sui dati ridotti potremmo ottenere risultati simili al taglio normalizzato; in letteratura si trovano diversi algoritmi

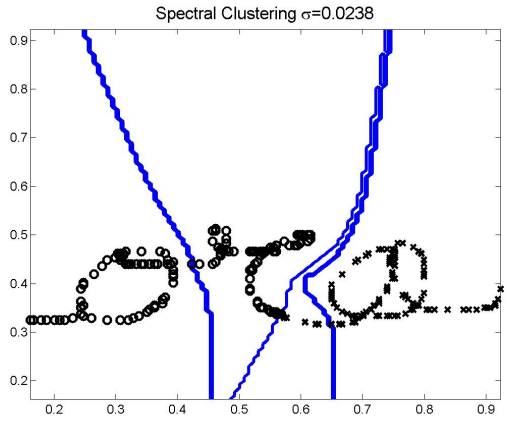

Figura 3.36: Partizione multiclasse

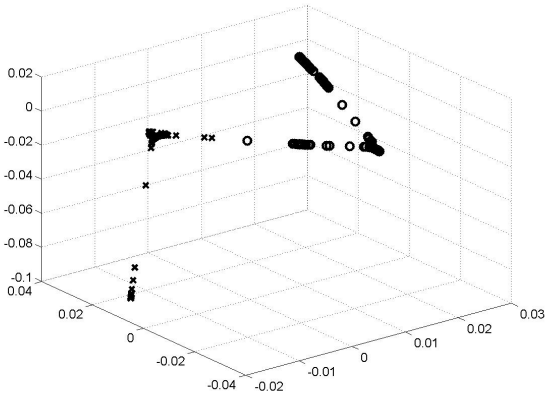

Figura 3.37: La riduzione della dimensionalità

che utilizzano l'informazione del secondo autovettore proprio con K-Means piuttosto che con la proposta di questa tesi.

Di grande interesse è la partizione che si ottiene utilizzando i primi due autovettori, in figura 3.36. Si è in grado di scomporre correttamente le tre lettere e un ulteriore linea di collegamento; questa linea potrebbe non passare un test di validazione che, ad esempio, conti il numero di vertici in una partizione, e potrebbe essere collegata ad un altro cluster. Per fare questo rimarchiamo la necessità di aggiungere assunzioni al problema.

3.3 Un'applicazione per la telesorveglianza

Nei sistemi di telesorveglianza si utilizzano delle telecamere per sorvegliare dei corridoi o delle stanze; di solito il fine è il controllo dell'accesso di persone autorizzate o estranei in questi luoghi.

Le telecamere sono usualmente fissate ad un appoggio, ad esempio una parete, e trasmettono le loro immagini ad un operatore; l'operatore valuta gli avvenimenti nella scena proiettata e decide se sia normale oppure un caso in cui intervenire.

Spesso è di interesse controllare delle zone in cui ci sono pochi avvenimenti, in questi casi si preferirebbe sostituire l'operatore umano con un calcolatore, ma può una macchina sostituire l'uomo?

Se si semplifica abbastanza il compito del calcolatore, questo può avvenire; adesso ne vedremo un semplice esempio.

3.3.1 Dalla realtà ai numeri

Utilizzare telecamere per controllare la realtà è un processo delicato che prevede l'utilizzo di tecniche di elaborazione di segnali e di visione artificiale.

Il primo fatto che cattura l'attenzione di chi lavora con le immagini digitali è la presenza del 'rumore', pixel provenienti dallo stesso punto della scena possono avere intensità leggermente differenti in diversi fotogrammi. Per questo si utilizzano dei filtri che 'ripuliscano' l'immagine. Dopo aver migliorato la qualità dell'immagine si procede ad applicare un algoritmo di visione artificiale per estrarre delle informazioni di interesse. La materia è vasta (vedi [ET98]), nel nostro caso utilizzeremo una semplice tecnica per capire se è presente del movimento nella scena e localizzare questo movimento.

Il corridoio In figura 3.38 una professoressa di visione artificiale si sta spostando attraverso un corridoio. Senza entrare nei dettagli tecnici vediamo un riquadro rosso che racchiude la figura. Questo riquadro prova a racchiudere le sagome degli oggetti in movimento nella scena.

Il riconoscimento dei pixel in movimento avviene comparando la scena vista dalla telecamera con uno sfondo memorizzato, rappresentante semplicemente il corridoio senza persone al suo interno.

3.3.2 Gli esempi

In questo esperimento gli esempi sono costituiti da registrazioni di persone, o oggetti, che si muovono nella visuale della telecamera; nella pratica quando viene individuato un movimento nella scena viene iniziata una registrazione, quando non ci sono più movimenti la registrazione viene interrotta. L'insieme di queste registrazioni costituisce il training set.

La maggior parte di queste registrazioni consiste in persone che camminano lungo il corridoio; questi passaggi sono diversi tra loro e siamo interessati a raggrupparli in cluster.

Figura 3.38: Una persona cammina nel corridoio

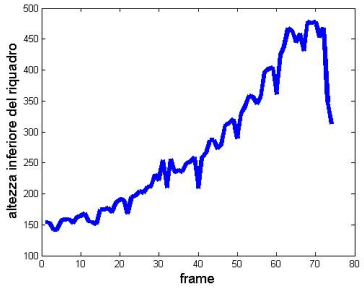

Figura 3.39: Una persona cammina nel corridoio: Grafico

Figura 3.40: Una persona cammina nel corridoio: Percorso

Il raggruppamento deve evidenziare il tipo di passaggio registrato, per farlo bisogna trasformare ogni registrazione in una struttura che evidenzi la variazione della posizione di una persona nel corridoio; la misura più corretta di questa variazione potrebbe consistere nel proiettare il baricentro del fisico della persona sul pavimento e quindi di considerare l'altezza di questa proiezione rispetto alla base dell'immagine. Purtroppo è difficile stimare con precisione la sagoma di un oggetto in movimento e di conseguenza il suo baricentro.

Una tecnica più semplice consiste nel considerare l'altezza dei piedi rispetto alla base dell'immagine; l'informazione è abbastanza simile alla proiezione del baricentro ed è facilmente ricavabile utilizzando la base del riquadro rosso attorno alla sagoma.

Anche in questo caso però non si riesce ad essere precisi e in effetti la stima della posizione del piede soffre di qualche oscillazione indesiderata; comunque la misura è sufficientemente buona per l'analisi ad alto livello che eseguiamo.

In figura 3.39 vediamo un esempio di un grafico di un passaggio di una persona nel corridoio, nell'ascissa del grafico ci sono i frame della registrazione, in ordinata c'é l'altezza della base del rettangolo rosso che racchiude l'oggetto. Si nota l'andamento ascendente del grafico e la presenza di oscillazioni.

Nell'implementazione l'altezza del piede viene considerata dall'alto dell'immagine, quindi l'andamento raffigurato nel grafico appartiene ad una persona che è apparsa in alto ed è scesa verso il basso, scomparendo dietro alla telecamera; il percorso corrispondente è raffigurato in figura 3.40.

Se due persone percorrono nello stesso modo il corridoio, ma con differenti velocità, i due grafici corrispondenti sono diversi, anche se la loro forma è uguale; per basare il confronto, tra le diverse passeggiate, sulla forma, i grafici sono stati normalizzati in modo tale da avere la stessa lunghezza.

Per il calcolatore un grafico non è altro che un vettore di numeri, in seguito ci riferiremo agli esempi chiamandoli vettori.

3.3.3 L'algoritmo di clustering

L'algoritmo segue la falsa riga di quello presentato nella sezione 2.5, vediamone i passi

input: training set di vettori normalizzati, una soglia per il valore del taglio normalizzato

1. stimo il parametro σ in modo che $\lambda_1 \in [10^{-3}, 10^{-3} + 10^{-6}]$ come visto nella sezione 3.1.2
2. calcolo il secondo autovettore
3. controllo la validità del secondo autovettore
   se è valido
   (a) se il valore del taglio è inferiore alla soglia, partiziono in due il grafo e ritorno ricorsivamente al punto 2 utilizzando in input ogni partizione e la stessa soglia
   (b) altrimenti non partiziono i dati e restituisco in output l'input

   altrimenti
   (a) vengono rimossi gli outliers dagli esempi e si ritorna ricorsivamente al punto 2 utilizzando in input i dati rimanenti in un caso e gli outliers nell'altro. La soglia rimane la stessa.

output: i cluster

Il processo di validazione di un autovettore è semplice; si controlla il valore del taglio normalizzato che si ha utilizzandolo, se questo valore è superiore all'upper bound del valore del taglio normalizzato (vedi sezione 3.1.2) significa che il taglio non è ottimale. Di solito si nota in tali casi la presenza di dati *outliers*: a questi dati l'autovettore attribuisce valori relativamente diversi dalla maggior parte degli altri dati, a cui invece viene assegnato un valore relativamente vicino a zero.

In figura 3.41 vediamo un esempio di autovettore non valido, la differenza relativamente elevata di alcuni dati viene utilizzata per considerarli *outliers*.

La soglia per il valore del taglio normalizzato è un parametro da stimare a priori; in questo caso ho eseguito degli esperimenti con una parte dei dati per valutare le differenti scelte, fissandolo a 0.7.

3.3.4 I risultati

Dato un training set di 278 elementi, i cluster trovati sono stati in tutto undici; sette di questi sono composti da meno di cinque dati considerati outliers. Nelle figure 3.42, 3.43, 3.44, 3.45 vediamo il vettore medio dei cluster principali. Il vettore medio è composto in modo tale che la sua cella i-esima è stata ottenuta mediando le celle i-esime di ogni vettore del cluster; rimarchiamo che i vettori sono stati normalizzati, e che quindi ogni frame sull'asse delle ascisse contiene un valore.

Le figure 3.42, 3.43, rappresentano le due partizioni principali, rispettivamente le camminate che vanno dalla telecamera verso il corridoio e dal corridoio verso la telecamera. Le figure 3.44, 3.45 non sono camminate, ma ad esempio è capitato che due o tre persone si fermassero nel corridoio a conversare, registrando diversi esempi in cui in effetti il movimento è breve e non consiste in una camminata.

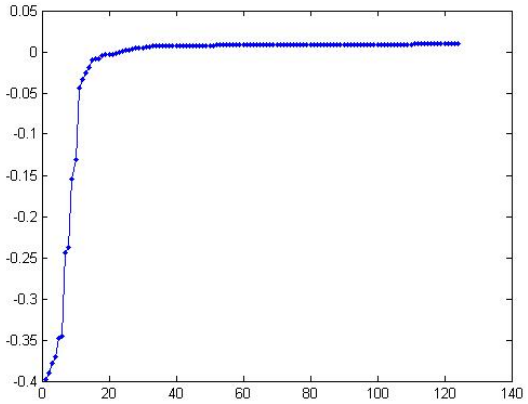

Figura 3.41: Un autovettore non valido con degli *outliers*

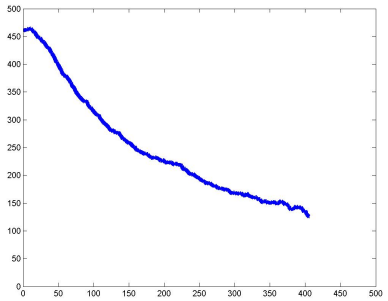

Figura 3.42: Elemento medio del primo cluster, 114 elementi

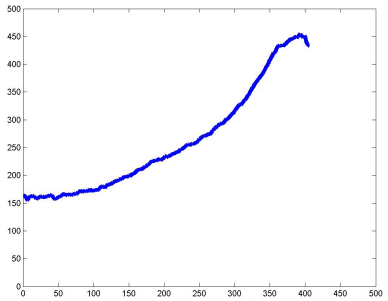

Figura 3.43: Elemento medio del secondo cluster, 112 elementi

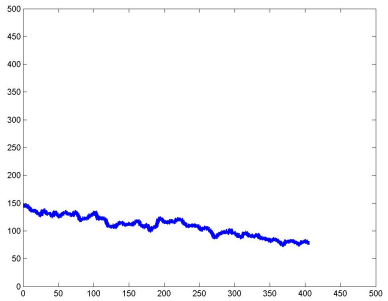

Figura 3.44: Elemento medio del terzo cluster, 23 elementi

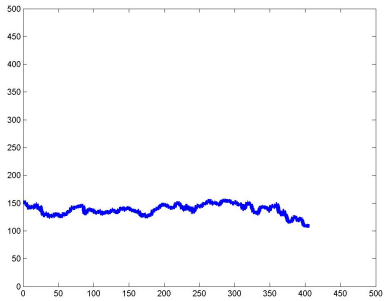

Figura 3.45: Elemento medio del quarto cluster, 12 elementi

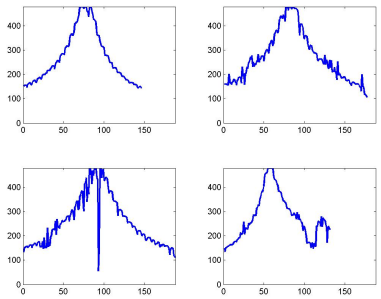

Figura 3.46: Cluster di camminate doppie

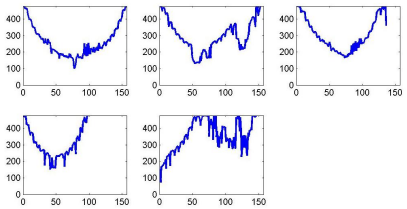

Figura 3.47: Un altro cluster di camminate doppie

Gli outliers Nell'esperimento sette cluster sono formati da outliers, alcuni di questi contengono un singolo dato molto diverso da tutti gli altri; al contrario altri raggruppano camminate dalla forma particolare, in figura 3.46 e 3.47 ho riportato due cluster significativi, essi rappresentano più passaggi nello stesso corridoio.

Per capire questo risultato basti pensare ad una persona che attraversa il corridoio in un senso e ad un'altra che lo attraversa subito dopo nell'altro senso; in questo caso la telecamera registra nello stesso esempio due camminate distinte, come si vede, ad esempio, dalla figura 3.46.

Conclusioni Questo esperimento dimostra che molti problemi di visione artificiale possono essere ricondotti al Clustering e risolti con delle tecniche di Spectral Clustering.

In questo caso si è dovuto risolvere un problema iniziale di trasformazione dei dati provenienti dalla telecamera. Durante questa fase è importante avere presente il tipo di apprendimento che si utilizzerà in seguito, per poter semplificare il problema nella direzione giusta. Il pericolo è smarrirsi nelle varie tecniche che si hanno a disposizione per raccogliere le informazioni dalla telecamera.

Uno sviluppo futuro di questa applicazione potrebbe prevedere la fusione di altre caratteristiche degli attraversamenti nel corridoio, come ad esempio l'ora del giorno o la dimensione della sagoma per capire se più persone lo attraversano contemporaneamente.

3.4 Segmentazione di immagini

Segmentare un'immagine significa riconoscere delle aree al suo interno che raccolgano pixel simili tra loro. La nozione di similarità tra pixel è ancora più vasta, nella let-

teratura della visione artificiale, della nozione di distanza esposta nella sezione 2.1.3, questo perché si possono assumere a priori molti fatti sulla provenienza dei pixel.

La caratteristica più semplice che possiamo considerare è l'intensità, in scala di grigi, di ogni pixel, unita alla sua posizione nel piano dell'immagine. In questi esperimenti utilizzeremo, proprio per la loro semplicità, queste informazioni.

3.4.1 Utilizzo dell'estensione out of sample

Utilizzare le tecniche di Spectral Clustering sulle immagini presenta alcune difficoltà di carattere tecnico. Prendiamo ad esempio un'immagine *bitmap* di dimensioni 100×100 pixel, al suo interno vi sono 10000 pixel; se costruiamo il Laplaciano su un grafo con 10000 vertici, dovremo allocare 10^8 celle di memoria, un numero piuttosto grande per un immagine piuttosto piccola.

Per ovviare a questo problema la principale tecnica proposta in letteratura è quello di utilizzare delle matrici sparse; non c'è una precisa definizione di matrice sparsa, in genere ci si riferisce al fatto che nella matrice un numero abbastanza elevato di valori è pari a zero. Un Laplaciano sparso denota quindi un grafo con un numero di connessioni relativamente basso.

In questi esperimenti vediamo un metodo alternativo per gestire il problema della dimensione del Laplaciano. Innanzitutto l'immagine viene sottocampionata in modo tale che il Laplaciano possa essere allocato in memoria e non si sia costretti ad utilizzare la tecnica delle matrici sparse; quindi ne vengono calcolati gli autovettori e gli autovalori.

Ora che abbiamo segmentato una versione rimpicciolita dell'immagine possiamo utilizzare i suoi autovalori ed autovettori per classificare ogni pixel dell'immagine originale, con la funzione out of sample relativa (vedi sezione 2.6).

3.4.2 Immagini segmentate

La forma dei dati Data un'immagine si costruisce un dato per ogni pixel considerando tre valori: l'intensità, nella scala dei grigi, del pixel, la sua posizione rispetto all'asse x e la sua posizione rispetto all'asse y. Il passo di sottocampionamento delle immagini è stato fissato a quattro, questo significa che l'immagine rimpicciolita contiene 1/16 dei dati iniziali.

Come detto nella sezione 3.1.2 scegliere il parametro σ è un problema difficile; fortunatamente è sufficiente trovare l'ordine di grandezza giusto e quindi lo si può fissare con alcuni tentativi su una parte dei dati. In questi esempi il valore di σ è stato così fissato a 1.45;

In questo caso non usiamo il bipartizionamento ricorsivo visto nell'esperimento della telesorveglianza in sezione 3.3.3,; come conseguenza bisogna stabilire il numero di autovettori da utilizzare per segmentare l'immagine. Dal momento che i segmenti che ci interessano sono abbastanza grandi, verranno utilizzati solo i primi nove autovettori.

Immagine sintetica In figura 3.48 vediamo un'immagine sintetica, 100×100 pixel, generata come un esperimento di prova.

Quest'immagine non presenta grosse difficoltà per l'algoritmo di partizionamento, in figura 3.49 vediamo il risultato dello Spectral Clustering sull'immagine sottocampionata; per semplicità di esposizione, nel caso dell'immagine sintetica, ho riportato il risultato del clustering in una sola immagine, attribuendo un'intensità di grigio ad ogni cluster. Quindi quello che vediamo in figura 3.50, apparentemente identico all'immagine originale, è un partizionamento corretto.

Fiore In figura 3.52 vediamo l'immagine ben contrastata di un fiore, 205×154 pixel; per questa piccola immagine il Laplaciano allocherebbe 10^9 celle di memoria, ognuna

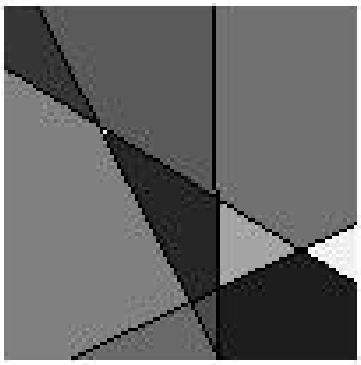

Figura 3.48: Un'immagine sintetica

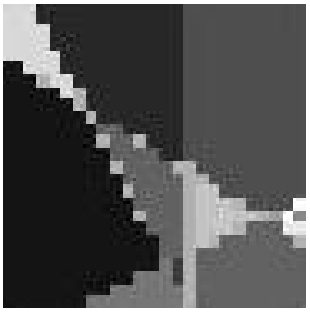

Figura 3.49: Il risultato sull'immagine sintetica sottocampionata

Figura 3.50: Utilizzo della funzione out of sample per l'intera immagine

Figura 3.51: Un fiore a colori

Figura 3.52: Un fiore in scala di grigi

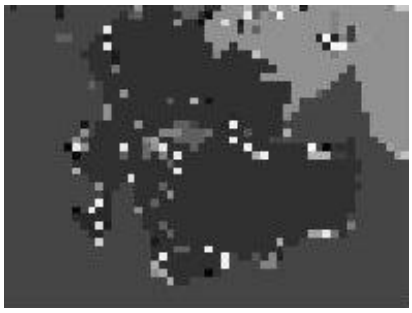

Figura 3.53: Il risultato sull'immagine sottocampionata

contente un numero in notazione floating point. Si capisce che il problema non è nel possedere un computer più potente, ma di studiare la gestione di casi di arbitraria grandezza.

In figura 3.53 vediamo il risultato dello Spectral Clustering sull'immagine sottocampionata; da questa immagine si può ricavare una segmentazione per l'immagine principale.

A differenza di prima mostrerò i principali segmenti ritrovati uno alla volta, a partire dai più importanti; anche se il problema è semplice, stupisce la precisione di questi segmenti, rispetto al risultato dell'immagine sottocampionata. I quattro frammenti più importanti sono riportati nelle figure 3.54, 3.55, 3.56, 3.57

Figura 3.54: I petali

Figura 3.55: Lo sfondo

Figura 3.56: Le foglie

Figura 3.57: Alcuni dettagli

Figura 3.58: Federico da Montefeltro

Figura 3.59: Federico da Montefeltro in scala di grigi

Federico da Montefeltro In figura 3.58 vediamo un dipinto di Piero della Francesca, eseguito intorno all'anno 1465; il soggetto è Federico da Montefeltro, un mecenate del Quattrocento; l'immagine è grande 131×190 pixel.

In figura 3.60 vediamo il risultato dello Spectral Clustering sull'immagine sottocampionata; da questa immagine si può ricavare una segmentazione per l'immagine principale.

Outlier Osservando gli autovettori rileviamo la presenza degli outlier; Rimuovendone una parte otteniamo l'autovettore in figura 3.61 e riconosciamo la sua struttura a tratti costanti; a differenza della simulazione in sezione 3.2.4 in questo caso possi-

Figura 3.60: Il risultato sull'immagine sottocampionata

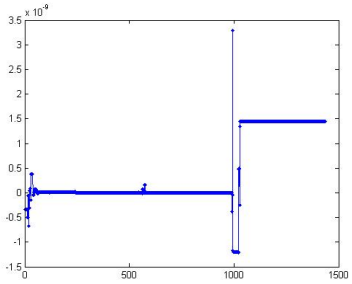

Figura 3.61: Autovettore costante a tratti

Figura 3.62: Risultato della segmentazione

Figura 3.63: Il corpo

amo fidarci dei risultati ottenuti utilizzando questi autovettori per l'estensione out of sample, fintanto che portano l'informazione che ci interessa.

Risultati Il risultato generale della segmentazione è riassunto nella figura 3.62, quindi sono stati trovate le partizione principali, ad eccezione dello sfondo. I dettagli dello sfondo sono più piccoli, così il sottocampionamento non riesce a rappresentarlo bene e lo partiziona con difficoltà. Questo è uno spunto per uno sviluppo futuro dell'algoritmo. I sette frammenti più importanti sono riportati nelle figure 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.69

Considerazioni tecniche Gli algoritmi per questa tesi sono stati sviluppati in Matlab^{TM}, questo linguaggio è costruito per operare efficientemente sulle matrici, quindi si è rivelato adatto per i nostri scopi; purtroppo l'utilizzo intensivo di questo strumento ha mostrato molti suoi limiti, in particolare per la gestione di matrici di grosse dimensioni.

Ogni algoritmo è stato sviluppato utilizzando le operazioni vettoriali più efficienti all'interno del linguaggio, così facendo l'ordine di tempo impiegato dall'esecuzione degli algoritmi è stato inferiore all'ora.

Gli sviluppi futuri prevedono il passaggio a linguaggi più efficienti che lavorano ad un livello di granularità più basso, come il C++.

Purtroppo il problema di trovare gli autovalori più piccoli di una matrice è numericamente malposto e quindi bisogna incorporare negli algoritmi futuri delle tecniche per gestire questi casi.

Figura 3.64: Il cappello

Figura 3.65: Il cielo

Figura 3.66: Il lato del viso

Figura 3.67: I capelli

Figura 3.68: Il viso

Figura 3.69: Un'altra porzione di cielo

Capitolo 4

Conclusioni

L'approfondimento dello studio delle tecniche di Spectral Clustering ci allontana dalla percezione che il calcolatore possa *imitare* il comportamento dell'uomo; per contro ci offre un'interessante applicazione della teoria spettrale dei grafi in un'ampia gamma di applicazioni di Machine Learning.

Nonostante sia stata fatta molta ricerca sullo Spectral Clustering, molte domande sono rimaste senza risposta e sono oggetto della ricerca attuale.

4.1 Lavoro futuro

Abbiamo visto degli esempi in telesorveglianza e in segmentazioni di immagini, i risultati sono ottimi, ma la strada da seguire non deve indirizzarsi a ottenere risultati migliori, ma a solidificare le fondamenta della teoria. Il lavoro futuro dovrà indagare alcuni problemi di fondamentale importanza:

Comprensione del ruolo di σ nell'utilizzo della funzione di similarità gaussiana Quando si applicano le tecniche di Spectral Clustering ci si rende immediatamente conto dell'utilità della funzione di similarità gaussiana e del ruolo di σ. Purtroppo sceglierlo è una questione difficile, che spesso richiede l'intervento dell'uomo.

In questa tesi è stata proposta una tecnica per stimarlo; questa tecnica assicura la qualità di un taglio normalizzato, ma questo non è sempre quello che desideriamo. Ad esempio sarebbe utile capire sin dall'inizio se 'conviene' bipartizionare il grafo o fermarsi senza operare divisioni.

Matrici sparse vs. Estensione out of sample Per motivi tecnici si utilizza spesso una matrice sparsa per rappresentare il Laplaciano normalizzato. In questa tesi ho proposto una tecnica alternativa utilizzando l'estensione out of sample.

Un confronto tra queste due tecniche è difficile, ma utile per capire come si possono affrontare i limiti attuali dei calcolatori.

Interazione fra diversi algoritmi di clustering Il Clustering non ha una definizione e non esiste una ricetta universale per risolverlo; inoltre spesso non è detto che le nuove ricette scavalchino quelle vecchie, piuttosto si può pensare che le affianchino.

Lo Spectral Clustering può essere utilizzato insieme ad altre tecniche partizionali, come il K-Means, grazie alla sua proprietà di riduzione della dimensionalità dei dati.

Capire come unire varie tecniche allo Spectral Clustering potrebbe permettere di sorpassare alcuni limiti di questa tecnica.

Gestione automatica del numero di cluster da ricercare Nello Spectral Clustering sono state fatte diverse proposte per capire il numero corretto di cluster (vedi [MMeX97]), queste spesso funzionano in una nicchia dei possibili casi che si possono affrontare. L'idea utilizzata nell'esempio della telesorveglianza è di imporre una soglia al valore del taglio normalizzato, ma la metodologia per scegliere questa soglia va studiata accuratamente.

In questo senso bisogna esplorare nuove soluzioni da confrontare con quelle esistenti.

Bibliografia

[Bel61] R.E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[Bie06] N Bie, T.D. e Cristianini. Fast sdp relaxations of graph cut clustering, transduction, and other combinatorial problems. *JMLR*, 7:1409, 2006.

[Das91] B.V. Dasarathy. Nearest neighbor norms: Nn pattern classification techniques. *CA: IEEE Computer Society Press*, 20, 1991.

[DVEA] Caponnetto A. Piana M. De Vito E., Rosasco L. and Verri A. Representer theorem for convex loss function. *DISI-TR-03-13*.

[eD84] S. Geman e D.Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. 1984. PAMI, 6:721-741, November 1984.

[eKW70] Davis C. e Kahan W. The rotation of eigenvectors by a perturbation. *III. SIAM J. Numer. Anal.*, 7:1, 1970.

[ePH73] R.O. Duda e P.E. Hart. *Pattern Classification and Scene analysis*. Wiley, New York, 1973.

[eSS03] Tomaso Poggio e Steve Smale. The mathematics of learning: dealing with data. *Notices Amer. Math. Soc*, 50:537, 2003.

[ET98] A. Verri E. Trucco. *Introductory techniques for 3D Computer Vision (Edizione 1)*. Prentice-Hall, 1998.

[F.97] Chung F. *Spectral Graph theory*, volume 19. 1997.

[F.98] Masulli F. Hard and fuzzy clustering paradigms. 19:701, 1998.

[Gol89] G.H. e Van Loan C.F. Golub. *Matrix computations*. John Hopkins Press, 1989.

[Gua98] G.L. Guattery, S. e Miller. On the quality of spectral separators. *SIAM journal of Matrix Anal. Appl.*, 19:701, 1998.

[McQ67] J. McQueen. 5th berkeley symposium on mathematics, statistics and probability. *Cornell Aeronautical Laboratory, Psychological Review*, 65:281, 1967.

[MMeX97] L. Meila M. e Xu. Multiway cuts and spectral clustering. 19:701, 1997.

[RLA04] De Vito E. Piana M. Rosasco L., Caponnetto A. and Verri A. Are loss function all the same? *Neural Computation*, 16, 2004.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory, Psychological Review*, 65:386, 1958.

[Shi00] J. Shi, J. e Malik. Normalized cuts and image segmentation. 2000. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Vap99] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory - second edition* -. Springer, 1999.

[VL04] U. e Belkin M. e Bousquet O. Von Luxburg. Consistency of spectral clustering. *Technical Report No. TR-134*, 2004.

[VL06] U. Von Luxburg. A tutorial on spectral clustering. *Technical Report No. TR-149*, 2006.

[WZ93] Leahy R. Wu Z. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *PAMI*, 11:1101, 1993.