DOTTORATO DI RICERCA

in

SCIENZE COMPUTAZIONALI E INFORMATICHE
Ciclo XXI

Consorzio tra Università di Catania, Università di Napoli Federico II, Seconda Università di Napoli, Università di Palermo, Università di Salerno

SEDE AMMINISTRATIVA: UNIVERSITÀ DI NAPOLI FEDERICO II

CARMINE DEL MONDO

PDM:*PREDATOR DATA MINING*
UN NUOVO APPROCCIO AL *DATA MINING UNSUPERVISED*

TESI DI DOTTORATO DI RICERCA

Responsabile scientifico:
Chiar.mo Prof. Lizzi Fedele

tutore:
Chiar.mo Prof. Trautteur Giuseppe

A mio padre . . .

Sommario

Si presenta in questa tesi un nuovo algoritmo adatto alla soluzione di un ampia categoria di problemi di data mining unsupervised: il PDM (Predator Data Mining).

Il PDM determina dei buoni centri di aggregazione dell'insieme dei dati classificandoli rispetto ad essi. I centri di aggregazione possono essere i classici centroidi, o strutture più o meno complesse. Queste possono immagazzinare, e quindi permettere di estrarre, maggiori, o comunque diverse informazioni rispetto alla classica operazione di data clustering, che è comunque un caso particolare del PDM

L'algoritmo sfrutta un approccio originale al problema che consiste nell'estrarre informazioni dall'insieme di dati, che vengono considerate come risorse per un insieme parallelo di "predatori". Gli insiemi di predatori sono generati da un particolare algoritmo evolutivo chiamato Competitive Evoluntion on Data (CED). L'insieme dei dati viene visto dal CED come un insieme di risorse: un ambiente, all'interno del quale popolazioni di predatori competono. Il PDM, tramite tecniche di data clustering, estrae, da alcune delle popolazioni generate dalla dinamica evolutiva del CED, i predatori che ritiene più significativi, la metrica utilizzata tra i predatori è definita intrinsicamente nell'algoritmo ed è indipendente dal particolare predatore adottato.

In questa tesi si è rivolta l'attenzione ad un particolare tipo di dati (punti in $\mathbb{R}^n$, si sono fatti vedere alcuni esempi di utilizzo con classe di predatori differente, usandolo come classico algoritmo di Data Clustering lo si è confrontato su alcuni data set che simulano l'espressione genetica dei geni (microarray) con gli algoritmi più frequentemente usati in quest'ambito, ed infine si è applicato il PDM su un data base reale confrontando i risultati con quelli ottenuti e pubblicati sullo stesso con altri procedimenti.

Inoltre, nell'acquisizione degli strumenti necessari per l'elaborazione dell'algoritmo, si sono fatte alcune considerazioni significative sui coefficienti utilizzati per confrontare classificazioni. In pratica si associa ad un risultato di un operazione di clustering un classificatore booleano, chiamato di accoppiamento, che unifica i coefficienti utilizzati nel confronto tra classificazioni *unsupervised* con quelli utilizzati tra classificazioni *supervised*. Questa unificazione permette tra l'altro di definire nuovi coefficienti che hanno interessanti caratteristiche.

Indice

Capitolo 1

Introduzione

1.1 Introduzione al Data Mining

Negli ultimi anni vi è stato un crescente interesse per le tecniche di Data Mining grazie anche all'aumento dell'importanza delle banche dati. Queste tecniche si dividono in due categorie: *Supervised* - In cui un supervisore esterno definisce, attraverso esempi, le classi. *Unsupervised* - Si tenta di estrarre alcune proprietà analizzando il data set, senza supervisore o classi predefinite. Generalmente i dati vengono raggruppati attraverso un criterio di somiglianza, spesso definito da un criterio di similarità o dissimilarità come una distanza (*Data Clustering*). In questa tesi si presenta un nuovo algoritmo di Data Mining Unsupervised che può essere usato come algoritmo di Data Clustering ma non è limitato a questo.

L'attivita di clustering, o più formalmente la *cluster analysis*, può essere definita come quel processo di organizzazione di oggetti (*pattern*) in gruppi (*cluster*) i cui membri sono "simili". Quindi, i pattern di un cluster presentano una forte similarità fra loro e, al contempo, un'inferiore similarità con i pattern degli altri cluster. La definizione fornita, sebbene sia intuitiva e apparentemente semplice, apre un ventaglio di problematiche operative legate alla definizione e alla misurazione della similarità, nonché riguardo ai metodi e alle implementazioni con cui effettuare il raggruppamento dei pattern in base a detta similarità.

Ci sono molte metodologie di Data Mining Unsupervised applicate ai più

svariati campi ([66] [67]), in questa tesi ci concentriamo su raggruppare dati multidimensionali ([68]) e la sua applicazione a catalogare geni tramite microarray ([74], [64], [70]). Le problematiche sulla similarità o dissimilarità sono, generalmente, specifiche alla applicazione e per questo verranno trascurate in questo capitolo.

Nel capitolo §2 si spiegherà il PDM, l'approccio, il funzionamento e, con l'aiuto di esempi pratici , gli obbiettivi. Nel capitolo §3 si utilizzerà il PDM come algoritmo di Data Clustering per raggruppare insiemi di dati che simulano le espressione geniche sui microarray, i risultati verranno confrontati con quelli ottenuti dagli algoritmi di data clustering principalmente utilizzati in questo ambito. Nel capitolo §4 si applicherà il PDM su alcuni dati reali: le espressioni geniche dei geni del lievito. Questo insieme di dati è stato molto studiato ed analizzato ([74]) e sono disponibili strumenti ([75]) che permettono di ricavare la significatività biologica dei cluster ottenuti, si sono utilizzati questi strumenti sia per valutare la bontà dei risultati ottenuti sia per confrontare rispetto ai risultati ottenuti con tecniche alternative sugli stessi dati ([4]). Infine nel capitolo §6 verranno tratte le conclusioni e si ipotizzano gli eventuali sviluppi di questa tesi.

In questo capitolo verranno introdotte le basi per meglio comprendere il PDM. In prima fase, essendo sostanzialmente il PDM un algoritmo atto a raggruppare dati, in §1.2 si daranno dei cenni su quelli che sono i principali algoritmi di raggruppamento. In §1.3 si parlerà di strategie evolutive dato che una componente del PDM utilizza questo tipo di strategia. Infine in §1.4 si parlerà dei coefficienti utilizzati per confrontare classificazioni, questi risulteranno utili sia per le tecniche di validazione e confronto dei risultati sia perché sono usati dall'algoritmo PDM per ricavare i risultati. La sezione §1.4 sarà abbastanza approfondita, in effetti in aggiunta al PDM questa tesi vuole dare un altro contributo: un nuovo approccio per ricavare coefficienti per confrontare classificazioni. Questo contributo è indipendente dal PDM per questo verrà trattato nel capitolo 5.

1.2 Data Clustering

Prima di parlare delle metodologie di Data Clustering è opportuno dare un breve sguardo ai possibili tipi di dati trattabili.

In generale possiamo classificare i tipi di dati in tre famiglie:

dati quantitativi: ossia semplici dati numerici; ogni pattern è un vettore che ne rappresenta quantitativamente le caratteristiche;

dati categorici: cioé dati sui quali non ha senso eseguire operazioni matematiche (es. calcolo della distanza) o predicati se non l'uguaglianza;

dati metrici: dati per i quali è definita una funzione distanza (cioé una funzione che rispetta i quattro assiomi della metrica) che associa a due pattern un numero reale rappresentate la distanza fra essi; ovviamente i dati quantitativi rientrano in questa categoria che però risulta più estesa.

La classificazione in famiglie degli algoritmi di Data Clustering non è molto semplice. Tradizionalmente questi vengono divisi in due grandi famiglie, la prima consiste in algoritmi di tipo gerarchico, la seconda, invece, raggruppa gli algoritmi partizionanti. Il continuo fervore e la diversificazione degli studi in quest'ambito porta ad aumentare costantemente il numero di algoritmi e con esso il numero di famiglie, entrando più nello specifico e tralasciando i metodi che trattano dati categorici, gli algoritmi principali possono essere meglio suddivisi nelle seguenti classi:

- Algoritmi gerarchici
- Algoritmi partizionanti
- Algoritmi basati sulla suddivisione dello spazio
- Algoritmi basati sulla densità.

1.2.1 Algoritmi gerarchici

Questa classe di algoritmi opera una decomposizione gerarchica del data set creando una struttura ad albero (detta dendogramma) che suddivide ricorsivamente il database; tale struttura può essere costruita con un approccio *bottom up* o *top down*. Nel primo caso si parla di algoritmi agglomerativi, questi iniziano con un cluster contenente un singolo pattern e ricorsivamente fondono due o più cluster "vicini". Viceversa l'approccio *top down* viene utilizzato dagli algoritmi chiamati separativi, quest'approccio consiste nel partire da un singolo cluster costituente l'intero data set che viene ricorsivamente diviso in due o più cluster (*subcluster*). Due algoritmi che esemplificano queste diverse metodologie sono rispettivamente AGNES e DIANA ([5]), fra i primi a implementare un clustering gerarchico. In entrambi i casi il processo di ricorsione termina quando si giunge ad un criterio di terminazione (spesso ma non sempre la richiesta di un certo numero k di cluster).

Il vantaggio degli algoritmi gerarchici sono:

- Flessibilità nel determinare il livello di granularità della classificazione.
- Facilità a maneggiare differenti forme di similarità o distanze.
- Applicabilità a qualsiasi tipo di dati.

Tra gli svantaggi, invece, c'è un oggettiva difficoltà nello scegliere il criterio di terminazione e, in almeno la maggior parte degli algoritmi, l'impossibilità a rivisitare cluster intermedi una volta che sono stati divisi o raggruppati.

Indipendentemente da se abbiamo a che fare con algoritmo di tipo agglomerativo o divisivo, un punto cruciale è scegliere i clusters da mettere assieme o da separare. Questa decisione dipende dalla similarità (o specularmente dissimilarità) tra gli elementi del cluster o dei clusters. Per fondere o dividere sottoinsiemi di pattern è necessario generalizzare ad i sottoinsiemi la distanza (o similarità) tra i singoli pattern. Questa misura viene chiamata metrica concatenata (*linkage metric*) il tipo di concatenazione usata ha un significativo impatto sull'algoritmo gerarchico, perché influisce su concetti di chiusura e connettività. Tra le metriche

concatenate più utilizzate abbiamo il singolo collegamento il collegamento medio o il completo (*single* o *average* o *complete link*). Chiamata d la metrica o il criterio di similitudine adottato tra pattern si ha che la metrica o criterio concatenato tra sottoinsiemi di pattern può essere formalizzata con la seguente equazione:

$$d(C_1, C_2) = Op\{d(x,y), x \in C_1, y \in C_2\} \tag{1.1}$$

Op è una funzione che può essere, come nel caso dell'algoritmo SLINK [9] (Op=Min), il minimo, o, come in [11], la media (Op = Avr) o infine, come con l'algoritmo CLINK [12], il massimo (Op = max).

Un altra maniera per definire una distanza concatenata è rappresentare i cluster con un loro punto centrale, ogni volta che si raggruppano due sottoinsiemi la distanza del sottoinsieme risultante dagli altri sottoinsiemi viene determinata dalla formula di Lance e Williams [10].

$$d(C_i \bigcup C_j, C_k) = a(i)d(C_i, C_k) + a(j)d(C_j, C_k) + b \cdot d(C_i, C_j) + c|d(C_i, C_k) - d(C_j, C_k)| \tag{1.2}$$

Scegliendo opportunamente i coefficienti a b e c ci si può ricondurre ai casi di *single*, *average* e *complete linkage*.

La distanza concatenata viene costruita dagli elementi della matrice delle distanze (dissimilarità) o similarità chiamata matrice delle connessioni. Le dimensioni di questa matrice sono $N \times N$ dove N sono le dimensioni del data base, questo crea dei problemi anche su Data Base non eccessivamente grandi. Per grandi data set mantenere in memoria tale matrice è impossibile per questo vengono usate alcune tecniche per "radere"(*sparsify*) la matrice, questo può essere fatto introducendo dei zero al posto di alcuni valori inferiori ad una certa soglia oppure utilizzando solo un sottoinsieme di pattern rappresentanti l'insieme di dati oppure conservando per ogni punto solo un numero limitato di "vicini" ([13]).

Fra gli algoritmi gerarchici è d'obbligo annoverare BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) ([14]), non tanto per i risultati prodotti dalla sua prima implementazione, quanto per la tecnica di compressione dei dati che introduce e la conseguente scalabilità che raggiunge, rappresentando

così una migliore soluzione per il trattamento dei data set di grandi dimensioni rispetto a tecniche di campionamento.

Attraverso algoritmi gerarchici è anche possibile ottenere cluster non sferici, in questo caso sono da menzionare il CURE (Clustering Using Representatives; [15]) e il CHAMELEON ([16]).

1.2.2 Algoritmi partizionanti

Gli algoritmi partizionanti funzionano iterando partizioni del data set ottimizzando la classificazione rispetto ad un criterio, a seconda del criterio e del metodo adottato ci sono diverse famiglie di algoritmi partizionanti le più note sono le seguenti:

- K-means
- K-medoids
- Modelli probabilistici

Questi algoritmi hanno un ottima qualità dei cluster, costi computazionali relativamente bassi e possono avere anche dei risultati facilmente interpretabili. Il loro limite principale sta nella necessità di stabilire a priori il parametro k, cioé il numero di cluster in cui raggruppare i pattern. Valori errati di k conducano a risultati decisamente scadenti anche per problemi di clustering apparentemente semplici. Questo tipo di esigenza si concilia male con un'attività non supervisionata.

Una possibilità è ricercare iterativamente il miglior valore di k, questa ricerca introduce però, oltre che una maggiore complessità, alcuni problemi nel confrontare la bontà tra classificazioni con differenti numero di cluster. Alcuni algoritmi percorrono comunque questa strada ([58]) anche solo per stimare il valore di k [53].

Un altro elemento dequalificante degli algoritmi partizionanti è costituito dalla forte dipendenza del risultato dalla scelta delle condizioni iniziali. Infatti, nonostante le successive iterazioni tendano a migliorare il risultato, può capitare, dipen-

dentemente dalla scelta iniziale, di terminare in un minimo locale che risulta ben lontano dall'ottima suddivisione dei pattern.

K-means

Il K-means ([60]) provvede alla suddivisione di n pattern di input, rappresentati in uno spazio d-dimensionale, in k cluster, in modo tale da minimizzare un certo criterio di convergenza ad esempio la deviazione totale di ogni pattern dal centro del suo cluster. Tipicamente si minimizza l'errore quadratico, ossia la somma della distanza di ciascun pattern dal centro del proprio cluster (centroide). Tali algoritmi operano iterativamente secondo il seguente schema :

1. scelta arbitraria di k centroidi come soluzione iniziale,
2. calcolo del cluster di appartenenza di ciascun pattern in accordo con l'attuale scelta dei k centroidi,
3. calcolo dei centri dei cluster che diventano i nuovi centroidi ,
4. iterazione, tornando al punto 2, se non e soddisfatto il criterio di terminazione.

Il raggiungimento dell'ottimo globale, ossia dei k centri migliori, e un problema NP completo e quindi il criterio di convergenza prevede di arrestarsi dopo un certo numero di passi, o quando il miglioramento della soluzione non è più apprezzabile.

K-medoids

La differenza tra il k-means e il k-medoids sta nel come sono costruiti i centroidi. Nel k-means un cluster è rappresentato da un suo centroide che è calcolato con una media di punti del cluster (generalmente una media pesata). Nel k-medoids([5], [17]) invece i centroidi sono punti dell'insieme, cioé sono i punti che hanno la minima distanza media dagli altri punti appartenenti al loro stesso cluster.

Questa differenza ha due vantaggi: il primo è che non ci sono limitazioni sui tipo di attributi dei dati, il secondo è che la scelta dei centroidi (medoids) è

dettata dalle zone all'interno del cluster a maggiore densità ed è poco sensibile alla presenza di dati rumorosi.

Modelli probabilistici

L'approccio di questi algoritmi è identificare i cluster come una certa distribuzione probabilistica e ricercare quindi quei parametri del modello che meglio si adattano all'insieme analizzato. Più specificatamente, i modelli probabilistici assumono che l'insieme di dati è un'unione di diverse popolazioni con una certa distribuzione (esempio gaussiane).

Un esempio per questo tipo di algoritmi è data dal metodo EM (*Expectation-Maximization* [18]). Questo è caratterizzato da due passi:

***Expectation*:** nel quale si effettua un assegnamento "soft" delle istanze ai cluster (si calcola la probabilità che un'istanza appartenga ad un cluster)

***Maximization*:** nel quale si utilizzano gli assegnamenti del passo precedente per stimare, in maniera pesata in base alla probabilità di appartenenza di un'istanza al cluster, i parametri dei modelli.

Si iterano questi due passi fino alla convergenza dei parametri che si stanno stimando

Uno dei principali vantaggi di questi metodi è l'interpretabilità dei cluster determinati.

1.2.3 Algoritmi che si basano sulla densità

Uno spazio topologico può essere diviso nelle parti che sono tra loro legate. Quest'idea è alla base di questa famiglia di algoritmi e necessità di definire concetti di connettività, confine e sopratutto densità. Esistono due approcci al problema, nel primo di questi, seguito da ad esempio DBSCAN ([59]) e OPTICS ([19]), si definisce un insieme dei pattern "vicini" ad un pattern p come:

$$N_\varepsilon = \{q \in D | dist(q,p) \leq \varepsilon\} \tag{1.3}$$

dove con D si è indicato l'intero database costituito da n pattern e *dist* è la metrica o criterio di simiglianza utilizzata. Si potrebbe così stabilire se un pattern fa parte di un cluster o meno controllando se N_ε (o *Eps-neighborhood* numero di "vicini") contiene un numero minimo di pattern che si indicherà con MinPts. Tuttavia questo approccio sarebbe semplicistico, poiché occorre discernere fra i pattern che sono all'interno del cluster, definiti *core point* e quelli che sono ai margini, definiti *border point*. Infatti tipicamente la *Eps-neighborhood* di un border point sara significativamente inferiore rispetto a quello di un core point.

Anche se alcune definizioni richiamano implicitamente dati di tipo spaziali questi algoritmi possono essere applicati in generale ad i dati metrici, a differenza dell'altro approccio seguito ad esempio in DENCLUE (DENsity CLUstEring; [20]).

Quest'approccio parte da una solida base matematica, quì si cercherà di sottolineare gli aspetti più intuitivi. In questo caso si presuppone che i pattern sono identificabili da vettori di uno spazio n-dimensionale che si indica con F_n. L'idea di base consiste nell'immaginare che i punti si "influenzino" a vicenda e, ovviamente, tale influenza è maggiore quanto minore è la distanza che li separa. Più specificatamente, si può fornire una "funzione di influenza" che indichi con precisione l'influenza di un punto su di un altro. Risulta adesso intuitivo che, laddove vi sia un cluster, la somma dell'"influenza" dei punti sara maggiore rispetto alle altre regioni dello spazio, questo permette di definire una "funzione di densità" attraverso la quale l'algoritmo determina i cluster.

Questi algoritmi possiedono l'intrinseca capacità di rilevare cluster di forma arbitraria e di filtrare il rumore identificando gli *outlier*. Hanno invece l'inconveniente che alcune volte i risultati sono poco significativi (ad esempio si mettono assieme cluster a diversa densità). Da un punto di vista teorico sarebbe conveniente poter disporre a priori dei corretti valori di ε e MinPts per ogni cluster, nella pratica non è ragionevole pensare di possedere questo tipo di informazione; per questo ad esempio DBSCAN opera con dei valori globali di ε e MinPts (ossia gli stessi per ogni cluster). Ciò comporta, in talune situazioni, dei limiti oggettivi che, se in parte vengono superati da OPTICS costituiscono comunque un problema

ancora aperto.

Sia DBSCAN che OPTICS hanno inoltre problemi quando si hanno pattern con alta dimensionalità ed in questo caso l'approccio del DENCLUE è sicuramente preferibile. In assoluto il problema reale è comunque la difficile interpretazione dei cluster.

1.2.4 Algoritmi basati sulla suddivisione dello spazio.

Questa categoria di algoritmi (in inglese algoritmi grid-based) utilizza un approccio fondamentalmente diverso dai precedenti: sostanzialmente anziché ragionare sui dati si ragiona sullo spazio. Lo spazio viene quantizzato in un numero finito di celle sulle quali sono effettuate le operazioni di clustering. Questa metodologia consente una veloce computazione indipendente dal numero di pattern da classificare ma unicamente dipendente dal numero di celle in cui lo spazio viene quantizzato. Le celle riassumono il contenuto dei pattern in esse, ciò consente tipicamente di mantenere l'intero data set in memoria centrale; inoltre operando sulle singole celle questi algoritmi sono intrinsecamente parallelizzabili. Diverse sono le tecniche adottate: STING ([7]) adopera un approccio statistico; WaveCluster ([8]) si basa sulla trasformata wavelet; CLIQUE ([6]), appositamente studiato per lavorare in alta dimensionalità, basa la ricerca di cluster in sottospazi. Tutti i precedenti algoritmi presentano comunque una complessità lineare nel numero di pattern costituenti il data set, trattando con "disinvoltura" database di centinaia di migliaia di pattern. Inoltre anche la scalabilità nei confronti della dimensionalità è decisamente migliore dei precedenti algoritmi consentendo di lavorare sufficientemente bene fino a 20 dimensioni nel caso di WaveCluster e decisamente oltre nel caso di CLIQUE. Vi è inoltre una buona capacità di riconoscimento di cluster di forma arbitraria ed una modesta sensibilità del risultato rispetto ai parametri di ingresso.

Il limite di questi metodi risiede nella qualità dei cluster forniti che dipende in modo significativo da quanto fine è la quantizzazione effettuata con la quale è quindi necessario stabilire un *trade-off*.

1.2.5 Applicazione di Reti Neurali Non Supervisionate per Data Mining

Tra le metodologie non supervisionate adottate, possiamo citare: le Self Organizing Map ([21]), le quali costituiscono un'architettura di rete neurale per l'apprendimento non supervisionato. I neuroni che formano una rete SOM sono disposti su di una griglia bi-dimensionale ed ognuno di essi ha un peso associato della dimensione dei dati di input e denominato *reference vector*. I neuroni competono tra loro: quando un dato di input x è presentato alla rete, il neurone ad esso più vicino (in senso topologico) è attivato secondo un'opportuna metrica (generalmente euclidea). In questo modo il peso associato al neurone vincitore (denominato anche *Best Matching Unit*) viene aggiornato in modo da essere più simile al vettore di input per il quale ha vinto la competizione. Inoltre, anche i pesi dei neuroni vicini (in senso topologico) al neurone vincitore sono aggiornati.

In questo modo, alla fine della fase di apprendimento la griglia di neuroni si auto-organizza in aree, ognuna delle quali corrisponde a gruppi di dati di input simili. Questa caratteristica viene sfruttata sia per scopi di clustering che per visualizzare i dati su spazi generalmente bi-dimensionali ([23]).

Da quando sono state proposte ([22]), esse hanno avuto un considerevole successo in un'ampia gamma di applicazioni e sono state oggetto di numerosi lavori di ricerca. Tuttavia, esse mancano di basi teoriche ben fondate e sono generalmente motivate da argomenti euristici. Le SOM consentono solo un'approssimazione della funzione densità di probabilità dei dati di input. Inoltre le proprietà di convergenza non sono state provate. Queste sono le motivazioni alla base dello sviluppo dello Generative Topographic Mapping [24], un modello generativo (ovvero che costruisce la funzione densità di probabilità nello spazio dei dati) a variabili latenti (tale modello di densità di probabilità è espresso in termini di un insieme di variabili di dimensione inferiore rispetto alla dimensione originale dello spazio dei dati). La corrispondenza tra lo spazio latente e lo spazio dei dati è realizzata mediante la costruzione di un opportuno mapping non lineare (generalmente mediante una rete neurale con funzioni radiali di base). Poichè lo spazio latente ha dimensione 2 o 3 è possibile visualizzare la distribuzione dei dati di

input in termini della distribuzione di probabilità nello spazio latente.

Il GTM offre diversi vantaggi rispetto alle SOM: 1) modello parametrico superiore 2) numero minore di parametri da impostare 3) la convergenza è garantita. Le SOM hanno però un interesse teorico intrinseco, sono state una delle prime applicazioni delle reti neurali non supervisionate ed i processi adattivi che avvengono all'interno della rete sembrano avere forti analogie con i processi adattivi che avvengono nel cervello dell'uomo.

1.3 Utilizzo di strategie evolutive nel Data Mining Unsupervised

1.3.1 Cenni sugli algoritmi genetici

La teoria dell'evoluzionismo, proposta inizialmente da Charles Darwin nel secolo scorso, ipotizza che il progredire degli esseri viventi, da semplici forme cellulari a complessi organismi, sia avvenuto attraverso variazioni casuali di caratteristiche ereditabili associate alla selezione naturale: solo con la più recente scoperta del DNA, che risulta essere una vera e propria codifica genetica di tali caratteristiche, si avranno ulteriori conferme di questa dinamica. Più precisamente, poiché in un dato ambiente le risorse necessarie per la vita degli organismi biologici sono presenti in quantità comunque limitate, nasce tra i singoli individui, di specie uguale o diversa, la competizione per assicurarsele. È questa la lotta per l'esistenza dalla quale si arriva alla selezione naturale degli individui migliori. I sopravvissuti a tale competizione mostrano di conseguenza maggiori capacità di adattamento all'ambiente ostile, ed hanno quindi anche maggiori probabilità di perpetuare la propria stirpe. Le nuove caratteristiche biologiche dell'organismo, introdotte in modo casuale nel processo riproduttivo, possono apportare o meno delle conseguenze rispetto alla lotta per la sopravvivenza nell'ambiente considerato, ma solo nel caso siano utili tenderanno a diffondersi e a consolidarsi stabilmente nella popolazione.

L'idea dei paradigmi evolutivi è simulare questa strategia per risolvere problemi complessi. Nelle linee generali lo schema è il seguente:

- Si definisce come individuo la generica soluzione del problema. Questa viene codificata in una struttura che permette di definire alcuni operatori. Gli operatori e la codifica, richiamando l'associazione biologica, vengono generalmente chiamati rispettivamente operatori genetici e genotipo.
- L'adattamento all'ambiente dell 'individuo è associato alla bontà della soluzione di risolvere il problema. Questa capacità viene generalmente quantizzata attraverso una funzione chiamata di *fitness*.

L'idea alla base è che con queste associazioni simulando le strategie evolutive si ottengono ottime soluzioni ad un problema cosi come la natura ha prodotto ottimi individui all'ambiente.

Introdotti da Holland [26], tra gli algoritmi che hanno utilizzato strategie evolutive i più famosi sono gli algoritmi genetici [25] e la programmazione genetica [27]. La differenza sostanziale tra questi è nel modo di codificare la soluzione di un problema, il tipo di genotipo utilizzato. Nel primo si utilizzano stringhe binarie nel secondo programmi scritti in LISP. Nelle linee generali lo schema comune è il seguente:

- Si crea una popolazione iniziale di individui
- Finché non si raggiunge un criterio di terminazioni
  - Si selezionano gli individui migliori (che hanno la migliore *fitness*).
  - Gli individui selezionati creano attraverso gli operatori genetici una nuova popolazione.
  - La nuova popolazione si sostituisce alla vecchia.

1.3.2 Algoritmi evolutivi e data mining

Nell'ambito del Data Mining gli algoritmi evolutivi vengono utilizzati per problemi *supervised* ([28]) mentre vi sono pochissimi esempi ([30]) di algoritmi

che si basano su strategie evolutive per risolvere problematiche di *Data Mining Unsupervised* .

Il motivo è facilmente spiegabile i paradigmi evolutivi sono adatti a risolvere problemi di ottimizzazione, necessitano sopratutto di una funzione di *fitness*. Nel supervised la funzione di *fitness* la si definisce naturalmente sfruttando l'insieme preclassificato (*training set*) e gli individui sono dei generici classificatori, nell'ambito dell'unsupervised si potrebbero usare i criteri di ottimizzazione utilizzati con gli algoritmi partizionanti (ed è quello che sostanzialmente fa il [30]) ma hanno notevoli costi computazionali.

1.4 Confronto fra classificazioni

La problematica del confronto tra diverse classificazioni è centrale nel data mining, in quanto permette di valutare l'efficacia e la bontà delle diverse tecniche di classificazione dei dati.

Il confronto tra le classificazioni permette principalmente di:

- Misurare l'errore del classificatore su un insieme di dati precedentemente classificato correttamente.
- Misurare discrepanze tra classificatori o tra differenti esecuzioni dello stesso classificatore[1].

I coefficienti per confrontare classificazioni vengono generalmente ricavati dalla *matrice di confusione* §1.4.1

Dato che uno dei principali scopi di questi coefficienti è valutare la bontà o differenze dei risultati di classificatori, questi possono risultare più o meno adatti a secondo del tipo di tecnica adottata per classificare. In generale i coefficienti adatti a confrontare classificazioni ottenute attraverso classificazione supervisionate non sono adatti per classificazioni ottenute in maniera non supervisionate.

[1]Diverse esecuzioni di uno stesso classificatore sono possibili nel caso in cui questo non sia deterministico o quando semplicemente la classificazione dei dati dipende da parametri variabili.

Numerosi autori hanno ricavato partendo dalla matrice di confusione molti coefficienti, sia per la problematica di confrontare classificazioni ricavate in modo supervisionato (§1.4.1), che da ora in poi chiameremo classificazioni supervisionate, sia in modo non supervisionato (§1.4.2), che chiameremo classificazioni non supervisionate. Come vedremo questi coefficienti sono molto diversi tra loro.

In §1.4.1 si darà una breve panoramica sui coefficienti usati nel caso di classificazioni booleane e nel caso di *classificazioni a molte classi*, in §1.4.2 si tratteranno i coefficienti invece usati nel caso unsupervised.

1.4.1 Confronto tra classificazioni a classi note

Definiamo una *classificazione A* di un insieme *D* come una partizione dello stesso: $A \equiv \{A_i\}_{i=1}^{C}$

dove gli A_i sono sottoinsiemi di D, C è il numero di sottoinsiemi in cui viene partizionato D. Gli A_i sono chiamati *classi* o *cluster* e devono godere delle seguenti proprietà:

- $A_i \bigcap A_j \equiv \emptyset \quad \forall i, j : i \neq j$
- $(\bigcup_{i=1..C} A_i) \equiv D$

La matrice di confusione

La Matrice di Confusione (in seguito MdC) viene introdotta come ausilio per valutare l'accuratezza di una classificazione.

Siano due classificazioni diverse di uno stesso insieme la MdC ad esse associata è tale che:

ogni generico elemento n_{ij} della matrice corrisponde al numero di dati classificati contemporaneamente nella *i*-esima classe della prima classificazione e *j*-esima classe della seconda classificazione.

La Mdc nel caso si confrontano classificazioni supervised è quadrata ed i coefficienti presenti nella diagonale rappresentano il numero di dati correttamente classificati.

Confronto di classificazioni booleane

Il classificatore più semplice è quello di tipo booleano. Il confronto tra le risposte di un classificatore booleano A rispetto ad una classificazione B ,considerata di riferimento, può venire riassunto nella seguente tabella:

$A \setminus B$	F	T
F	a	b
T	c	d

con i seguenti coefficienti:

a: il numero di dati classificati correttamente come negativi.

b: il numero di casi classificati come negativi ma che invece risultano positivi nella classificazione di riferimento.

c: il numero di casi classificati come positivi ma che risultano negativi nella classificazione di riferimento.

d: il numero di casi classificati correttamente come positivi.

La matrice di confusione si presenta così:

$$MdC(A,B) \equiv \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{1.4}$$

a partire da questa vengono (come sintetizzato in [31]) ricavati diversi coefficienti, in particolare abbiamo:

False Negative (FN): La proporzione di casi positivi incorrettamente classificati come negativi:

$$FN = \frac{c}{c+d} \tag{1.5}$$

False Positive (FP): La proporzione di casi negativi incorrettamente classificati come positivi:

$$FP = \frac{b}{a+b} \tag{1.6}$$

True Negative (TN): La proporzione dei casi negativi correttamente classificati:

$$TN = \frac{a}{a+b} \tag{1.7}$$

True Positive (TP): La proporzione dei casi positivi correttamente classificati:

$$TP = \frac{d}{c+d} \tag{1.8}$$

Precision (Pr): La proporzione dei casi predetti positivi e classificati correttamente:

$$Pr = \frac{d}{b+d} \tag{1.9}$$

Accuracy (AC): La proporzione dei casi totali che sono stati classificati correttamente.

$$AC = \frac{a+d}{a+b+c+d} \tag{1.10}$$

Improvement Accuracy (IAC): Particolarmente adatto quando non esiste una classificazione di riferimento. In questo caso i dati classificati come positivi da entrambe le classificazioni vengono confrontati con i dati positivi in almeno una delle due classificazioni:

$$IAC = \frac{d}{b+c+d} \tag{1.11}$$

Come illustrazione delle funzioni di questi coefficienti, consideriamo un data base in cui i casi negativi rappresentano il 90% dei casi totali; un classificatore che semplicemente classifica tutti i casi come negativi avrebbe un'ottima accuracy (0.9).

La necessità di avere misure che pesano equamente i casi positivi e negativi, indipendentemente dalla distribuzione dei dati nelle classi, ha portato a introdurre nuovi coefficienti ([32], [33]):

$$AAC = \tfrac{TP+TN}{2}[AverageAccuracy] \tag{1.12}$$

$$G-mean_1 = \sqrt{TP*P} \tag{1.13}$$

$$G-mean_2 = \sqrt{TP*TN} \tag{1.14}$$

$$F = \tfrac{(\lambda^2+1)*Pr*TP}{\lambda^2*Pr+TP} \tag{1.15}$$

Il coefficiente λ è usato per dare un diverso peso tra la capacità di classificare correttamente i casi positivi e quelli negativi, anche se sempre indipendentemente dalla distribuzione dei dati nelle classi. Ciò è utile nelle applicazioni dove si è interessati principalmente ad una specifica classe.

Un'altra strada per esaminare l'efficacia di una classificazione è tramite i grafi ROC [36]. Un *ROC graph* è un grafico con i veri positivi (Eq. 1.8) riportati sull'asse delle ascisse e i falsi positivi (Eq. 1.6) su quello delle ordinate.

In questa rappresentazione, il punto (1,0) rappresenta il perfetto classificatore: tutti i casi positivi e negativi sono classificati correttamente. Il punto (0,0) corrisponde a un classificatore che prevede tutti i casi come negativi. Il punto (1,1) corrisponde a un classificatore che cataloga ogni caso come positivo. Il punto (0,1) infine, corrisponde al classificatore che ripartisce tutti i dati incorrettamente.

Una maniera per ottenete una misura di accuracy da un ROC graph è quello di usare la distanza euclidiana rispetto al punto del perfetto classificatore:

$$d_{roc} = \sqrt{(1-TP)^2 + FP^2} \tag{1.16}$$

Le seguenti equazioni sono alternative per ricavare una distanza partendo da una qualunque delle misure di accuracy:

- $d_{AC} = n * (1 - AC)$
- $d_{IAC} = n * (1 - IAC)$
- $d_{AAC} = n * (1 - AAC)$

dove $n = (a+b+c+d)$ è il numero di dati dell'insieme classificato.

Confronto di classificazioni multi classe

Le misure tra classificatori non booleani sono concettualmente molto simili al caso booleano e sono state studiate in maniera più articolata nell'ambito del *remote sensing* ([34], [35]). In quest'ambito la classificazione è vista come la risposta di un processo che elabora immagini. Esistono due tipi di classificazioni

per immagini. Nella prima, ogni pixel dell'immagine è posto in una classe diversa. Nella seconda, i pixel possono essere ripartiti in una o più classi.

La prima di queste classificazioni ha maggiori punti in comune con il nostro ambito di studi, dato che la gran parte degli algoritmi di data mining usati come classificatori assegnano ogni dato ad una sola classe.

La MdC tra due classificazioni a molte classi è una matrice quadrata $[C \times C]$, dove C è il numero delle classi:

$$\begin{pmatrix} n_{11} & \dots & n_{1C} \\ \dots & \dots & \dots \\ n_{C1} & \dots & n_{CC} \end{pmatrix} \tag{1.17}$$

Gli elementi della diagonale maggiore sono il numero di dati in accordo con la classificazione di riferimento. Un primo gruppo di coefficienti dell'accuratezza si basa sulla misclassificazione dei dati. Per questo motivo, i coefficienti sono concettualmente simili a quelli precedentemente introdotti.

Siano date le seguenti grandezze:

n numero di dati dell'insieme classificato: $\sum_{i,j=1}^{C} n_{ij}$

n_{i+} Numero di dati presenti nell'i-esima classe della classificazione da confrontare: $\sum_{j=1}^{C} n_{ij}$

n_{+j} Numero di dati presenti nell'j-esima classe della classificazione di riferimento: $\sum_{i=1}^{C} n_{ij}$

si ricavano i seguenti coefficienti:

Overall Accuracy $\frac{1}{n}\sum_{i=1}^{C} n_{ii}$

User's Accuracy $\frac{n_{ii}}{n_{i+}}$

Producer's Accuracy $\frac{n_{ii}}{n_{+i}}$

Average accuracy (user's) $\frac{1}{C}\sum_{i=1}^{c} \frac{n_{ii}}{n_{i+}}$

Average accuracy (producer's) $\frac{1}{C}\sum_{i=1}^{C}\frac{n_{ii}}{n_{+i}}$

La *Overall accuracy* è la generalizzazione della Eq. 1.10 a molte classi.

Producer e *User's accuracy* indicano l'accuratezza delle singole classi e pertanto ve ne sono tante quante le classi.

Considerando ad esempio una matrice di confusione di due classificazioni booleane, i due coefficienti dell'user's accuracy non sono altro che i veri positivi Eq. 1.8 e negativi Eq. 1.7, mentre uno dei producer's corrisponde alla precision Eq. 1.9 (ed in particolare il coefficiente corrispondente alla classe True).

Le due formule di *average accuracy* sono la media di tutte le *user's accuracy*, in un caso, e delle *producer's accuracy* nell'altro[37].

Questi coefficienti non prendono in considerazione un possibile accordo casuale fra le classificazioni, tendendo a sovrastimare l'accuratezza [38].

Per questo motivo vengono introdotti altri coefficienti. In particolare illustriamo qui il *kappa coefficient* [39] e il *conditional kappa coefficient* [42].

Il kappa coefficient K fa riferimento all'intera classificazione e viene definito come segue:

$$K = \frac{P_o - P_e}{1 - P_e} \tag{1.18}$$

dove $P_o = \frac{\sum_{i=1}^{C} n_{ii}}{n}$; mentre:

$$P_e = \sum_{i=1}^{C} P_{i+}P_{+i} \tag{1.19}$$

con: $P_{i+} = \sum_{j=1}^{C} n_{ij}/n = n_{i+}/n$ e $P_{+i} = \sum_{i=1}^{C} n_{ij}/n = n_{+i}/n$

I valori P_{i+} e P_{+i} sono le probabilità di trovare un dato nella i-esima classe della prima e della seconda classificazione. Conseguentemente si ha che l'Eq. 1.19 esprime la probabilità attesa che un elemento si trovi nell'intersezione dei due cluster quando le classificazioni sono completamente scorrelate.

Il conditional kappa coefficient invece è relativo a una singola classe; ne esiste quindi uno per ogni i-esimo cluster. Il conditional kappa coefficient viene espresso

come segue:

$$\hat{K}_i = \frac{\frac{n_{ii}}{n} - P_{i+}P_{+i}}{1 - P_{i+}P_{+i}} \tag{1.20}$$

da notare che dagli i conditional kappa coefficient è possibile comunque ricavare un coefficiente globale della classificazione, sommando e mediando per il numero di classi.

$$\hat{K} = \frac{1}{C}\sum_{i=1}^{C}\hat{K}_i \tag{1.21}$$

In generale abbiamo che $\hat{K} \neq K$. Il rapporto fra $\hat{K}$ e K è simile al rapporto relativo all'Average Accuracy e l'Overall Accuracy cioé in entrambi i casi usano un tipo di coefficiente tra loro simile solo che il primo lo calcola su tutte le classi (Overall) mentre il secondo per ogni classe ed infine determina la media.

La famiglia dei Kappa coefficients hanno valori tra $[-1,1]$ e nello specifico valgono:

- 1 se le classificazioni coincidono (esempio tutti i casi positivi e negativi correttamente classificati)
- 0 se non esiste correlazione tra le due classificazioni (esempio la percentuale di casi positivi e negativi correttamente classificati è esattamente quella attesa se si ipotizza che almeno una delle due classificazioni classifichi casualmente)
- -1 le classificazione sono anticorrelate (esempio tutti i dati positivi vengono classificati come negativi e il viceversa)

I kappa coefficient sono sempre più adottati per confrontare classificazioni[54].

1.4.2 Confronto di classificazioni ottenute in modo unsupervised

Introduzione

Esistono tre famiglie di coefficienti per il confronto di classificazioni non supervisionate.

Il primo approccio che tratteremo in 1.4.3 è un adattamento dei coefficienti trattati in 1.4.1, il secondo trattato in 1.4.4 introduce un concetto nuovo che verra ripreso e formalizzato in §5 ed è quello che viene maggiormente usato ed utilizzeremo in seguito, infine un altro gruppo di coefficienti interessanti si basa sulla mutua informazioni tra le classificazioni [40], [41], coefficienti che non tratteremo solo perché non verrà mai utilizzata dal PDM e discosta dalla discussione che si farà in 5

1.4.3 Coefficienti che si basano sulla misclassificazione dei dati

Alcuni coefficienti che si basano sulla misclassificazione sono stati adattati al caso unsupervised.

Usando la stessa notazione precedende chiamati C_a e C_b il numero di cluster della classificazione A e B, riportiamo i seguenti indici ([49], [50],[51]):

$$L(A,B) = \frac{1}{C_a}\sum_{i=1}^{C_a}\frac{2max_j(n_{ij})}{n_{i+}+n_{+i}} \tag{1.22}$$

$$H(A,B) = \frac{1}{n}\sum_{i}^{C_a} match_j(n_{ij}) \tag{1.23}$$

$$D(A,B) = 2n - \sum_{i=1}^{C_a} max_j(n_{ij}) - \sum_{i=j}^{C_a} max_i(n_{ij}) \tag{1.24}$$

dove:

- $max_j(n_{ij})$ e $max_i(n_{ij})$ sono il massimo elemento n_{ij} presente rispettivamente nella i-esima riga e nella j-esima colonna;
- $match_j(n_{ij})$ è un operatore che seleziona l'elemento più grande della matrice ed elimina la riga e la colonna corrispondenti, l'operazione prosegue sulla matrice restante fino a esaurimento.

Questi coefficienti rispondono molto bene quando le due operazioni di data clustering individuono più o meno i stessi cluster, le operazioni $max_i(n_{ij})$, $max_j(n_{ij})$ e $match_j(n_{ij})$ servono a risolvere il problema dell'ordine dei cluster. Nell'ipotesi che i due classificatori determinano le stesse classi queste vengono prima ricavate con le operazioni di match o max e poi si può calcolare la misclassificazione.

I tre coefficienti misurano la misclassificazione in maniera diversa.

L(A,B) sceglie le classi attraverso $max_i(n_{ij})$ e calcola la misclassificazione per ogni classe similmente all'*improvement accuracy*. H(A,B) usa l'operatore match e misura la misclassificazione come l'*accuracy*. Infine D(A,B) fa una media tra l'operatore $max_i(n_{ij})$ e $max_j(n_{ij})$ in modo da rendere l'operazione commutativa rispetto ad A e B ed ottenere un criterio di dissimilarità che dà 0 nel caso di perfetta coincidenza tra i cluster.

In generale però l'ipotesi, nel caso di algoritmi non supervisionati, che i classificatori determinano le stesse classi è ottimistica.

1.4.4 Coefficienti che si basano sull'accordo nel raggruppare i dati: Pair misure

L'idea di base della *Pair misure* venne introdotta da Rand nel 1971 in [44] e ripresa in seguito anche in maniera apparentemente indipendente ([43]). Chiamati:

N_{11} Il numero di coppie di dati incluse contemporaneamente in un cluster della classificazione A e classificazione B

N_{00} Il numero di coppie che non sono incluse in un solo cluster sia della classificazione A che classificazione B

N_{10} Il numero di coppie che sono incluse in un cluster della classificazione A ma non in B

N_{01} Il numero di coppie che non sono incluse in un cluster della classificazione A ma che sono in B

Il Rand index viene definito nella seguente maniera:

$$R(A,B) = \frac{N_{11} + N_{00}}{n * (n-1)/2} \tag{1.25}$$

dove n è al solito il numero di dati del data base classificato.

Il Rand index viene ancora usato molto nell'ambito del confronto dei clustering, questo viene anche espresso su tutte le n^2 possibili coppie ([57]) indice che chiameremo:$Rand^I$ per distinguerlo da quello sulle coppie distinte. Il Rand index è alla base di altri indici introdotti in seguito:

W indexs [45]:

$$W_I(A,B) = \frac{N_{11}}{\sum_{i=1}^{C_a} n_{i+} * (n_{i+} - 1)} \tag{1.26}$$

$$W_{II}(A,B) = \frac{N_{11}}{\sum_{j=1}^{C_b} n_{+j} * (n_{+j} - 1)} \tag{1.27}$$

Dove al solito $n_{i+} = \sum_{j=1}^{C_b} n_{ij}$ e $n_{+j} = \sum_{i=1}^{C_a} n_{ij}$ rispettivamente le dimensioni dell i-esimo cluster della classificazione A e j-esimo cluster della classificazione B.

Il Fwl index [46]

$$Fwl(A,B) = \sqrt{W_I(A,B) * W_{II}(A,B)} \tag{1.28}$$

Il Jacard index ([47],[52]):

$$J(A,B) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}} \tag{1.29}$$

La Mirkin metrica [48]:

$$Mrk(A,B) = \sum_{i=1}^{C_a} (n_{i+})^2 + \sum_{j=1}^{C_b} (n_{+j})^2 - 2 \sum_{i,j=1}^{C_a,C_b} n_{ij}^2 \tag{1.30}$$

L' adjusted rand index [62] [53]:

$$RandAdj(A,B) = \frac{\sum_{i,j=1}^{C_a,C_b} \binom{n_{ij}}{2} - 1/\binom{n}{2} \sum_{i=1}^{C_a} \binom{n_{i+}}{2} \sum_{j=1}^{C_b} \binom{n_{+j}}{2}}{1/2[\sum_{i=1}^{C_a} \binom{n_{i+}}{2} + \sum_{j=1}^{C_b} \binom{n_{+j}}{2}] - [1/\binom{n}{2}] \sum_{i=1}^{C_a} (\binom{n_{i+}}{2}) \sum_{j=1}^{C_b} \binom{n_{+j}}{2}} \tag{1.31}$$

Per confronti di questi indici si può far riferimento ai lavori di Saporta [55] [56], in §5 invece proponiamo un nuovo approccio al problema che permette di ricavarli automaticamente e contemporaneamente tutti.

Questo approccio semplificherà la capacità di interpretarli ed inoltre apre la strada all'introduzione di nuovi coefficienti con interessanti caratteristiche.

1.5 PDM

L'idea alla base del PDM ([1]) è spostare la ricerca dallo spazio dei dati ad uno spazio parallelo di predatori di dati. I predatori sono strutture dati che, alla stregua degli individui nei paradigmi evolutivi, sono dotati di operatori genetici e sopratutto di una funzione chiamata *hunter-fitness* che permette di decidere per ogni dato il predatore migliore.

L'insieme dei dati è l'ambiente dove i predatori competono e si riproducono, saranno premiati dalla selezione della riproduzione quelli che riusciranno a conquistare più risorse, alcune delle popolazioni di predatori verranno analizzate per estrarre i predatori più significativi con cui classificare i dati.

La *hunter-fitness* è un criterio di dissimilarità tra strutture dati differenti, nel caso in cui si utilizza come predatore una struttura identica a quella dei dati si può usare una *hunter-fitness* ricavata da un criterio di similarità o dissimilarità classico, in questo caso i predatori determinati sono dei centroidi con cui è possibile classificare i dati similmente ad un algoritmo di data clustering, altrimenti i dati vengono raggruppati con metodi di aggregazione alternativi.

La scelta del predatore e della *hunter-fitness* associata, condiziona la metodologia con cui vengono raggruppati i dati, questo permette di raggrupparli con criteri che vanno al di là di una metrica o di un criterio di similarità, le possibilità sono enormi ed in questa tesi sono state appena introdotte a titolo di esempio. Questa scelta inoltre è condizione sufficiente per applicare il PDM e potendo scegliere il predatore anche in funzione del tipo di dati da analizzare il PDM non ha nessun limite in questo senso, tra l'altro l'*hunter-fitness* deve essere definita per ogni dato appartenente all'insieme analizzato ma non deve necessariamente essere la stessa per ogni dato, questo permette di lavorare su insiemi che sono collezione di tipo di dati differenti. In definitiva l'approccio originale al problema da al PDM parecchi vantaggi:

- la possibilità di utilizzare metodi di aggregazione complessi,
- la possibilità di lavorare su qualsiasi tipo di dato anche su insieme di dati non omogeneo,

- la presenza di un informazione aggiuntiva alla classificazione che dà un indicazione della qualità di questa.

Lo svantaggio principale è il costo computazionale dell'algoritmo, questo è proporzionale alla complessità della *hunter-fitness* limitando molto le possibili *hunter-fitness* utilizzabili.

Capitolo 2

PDM

2.1 IL PDM

Per scoprire zone marine con alta concentrazione di Plancton non è necessario misurare effettivamente la concentrazione di Plancton ma basterebbe notare la presenza e concentrazione di Balene e di altri predatori di Plancton. Esiste la possibilità di dedurre conoscenza su uno spazio di risorse analizzando i predatori di queste risorse.
L'idea del PDM è utilizzare quest'approccio per estrarre informazioni da insieme di dati, per, cioé, problemi di Data Mining Unsupervised.
Il PDM tratta l'insieme dei dati come un insieme di risorse e tenta di estrarre informazioni su di esso, prima costruendo e poi analizzando insiemi di predatori di dati.
Il PDM effettua le seguente operazioni:

- Attraverso un algoritmo chiamato CED (*Competitive Evolution on Data*) simula un processo evolutivo creando diverse popolazioni di predatori.
- Estrae, analizzando alcune di queste popolazioni, i predatori che ritiene più "significativi"
- Classifica i dati dell'insieme rispetto a questi predatori.

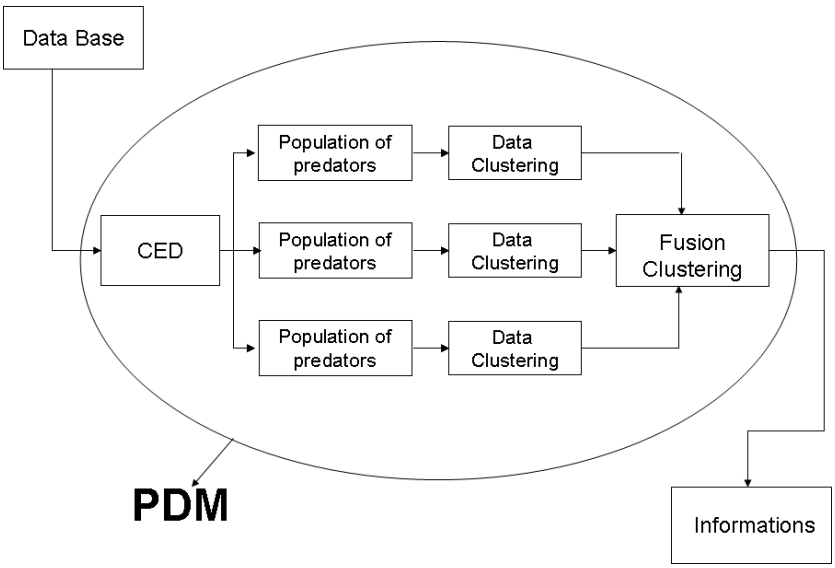


Figura 2.1: PDM Scheme

Le informazioni interessanti che si possono estrarre in questo modo dall'insieme dei dati sono di due tipi: le classi o cluster in cui vengono raggruppati i dati ed i predatori piú significativi che possono essere visti come dei classificatori.

Un predatore , per il PDM, non è un oggetto definito. La sua sola caratteristica è che deve confrontarsi con i dati in modo da stabilire la capacità di depredare ogni singolo dato. Il predatore può avere la stessa struttura dei dati ed essere per esempio un possibile centroide di un cluster, ma può anche avere strutture diverse (rette , ellissi o anche predicati [2]) permettendo di estrarre informazioni maggiori o diverse rispetto ad una classica operazione di *Data Clustering*.
In altri termini lo spazio dei predatori è uno spazio differente rispetto allo spazio in cui sono definiti i dati, ovviamente per applicare il PDM bisogna definire a priori il predatore, tale definizione necessita a parte che di una codifica matematica del predatore (genotipo) e di alcuni operatori genetici, di una funzione chiamata *hunter-fitness* (in breve *hf*) che quantificà la capacità del generico predatore di appropriarsi di ogni generica risorsa.

In questa tesi ci si concentra su una particolare tipologia di dati, il data base che si chiamerà D è composto da N punti in $\mathbb{R}^n (N \neq n)$. Un generico punto è chiamato x^i mente la sua generica componente verra determinata con x^i_j.

Dato questo insieme di oggetti il primo passo è definire il predatore, ad esempio un vettore in R^m ed una funzione *hunter-fitness* che collega il predatore al singolo dato. La quantità n non è necessariamente uguale ad m.
La procedura è indipendente dalla scelta del predatore e per questo può essere applicata senza problemi su insieme di oggetti con attributi di tipo differenti (per esempio nominali, interi e reali) o anche con "missing value", questi aspetti, infatti, influiscono solo sulla scelta della struttura del predatore e della *hf*. I risultati, invece, dipendono da questa scelta, questo permette, dato uno stesso insieme di dati, di effettuare un diverso tipo di ricerca.

2.1.1 I Predatori

Scegliamo come possibile predatore p^i un vettore di reali che possono rappresentare punti, linee o ellisse (alcuni degli esempi che verranno presentati).

Chiamato P l'insieme di tutti i possibili predatori p^i questo è separato dall'insieme D, un pò come lo spazio dei parametri nella trasformata di Hough è ([69]) separato dallo spazio delle immagini, in entrambi i casi infatti si usa una funzione che lega i due spazi.

Un predatore è un oggetto su cui sono definiti le seguenti operazioni:

1. Un operatore di *inizializzazione* che crea, in maniera casuale un predatore.

2. Un operatore di *mutazione* che dato un predatore $p \in P$ ne crea un altro stocasticamente vicino p'.

3. Una funzione di *Hunter-fitness*

$$hf : P \times D \rightarrow R^+ \tag{2.1}$$

   che quantifica la capacità del predatore p nel conquistare la risorsa x.

La funzione *hf* collega l'insieme dei predatori con l'insieme dei dati. Questa generalizza il concetto di vicinanza tra due punti al caso in cui gli oggetti appartengono a differenti spazi. In altre parole questa funzione, per ogni x^i, dà una relazione d'ordine nell'insieme dei predatori.

$$p^1 <_{x^i} p^2 \Leftrightarrow hf(p^1, x^i) < hf(p^2, x^i) \tag{2.2}$$

dove $<_{x^i}$ indica l'ordinamento per l'insieme P dato dal particolare punto x^i e dalla funzione *hf*.

Anche se il predatore ha diverse analogie con gli individui usati nei classici paradigmi evolutivi, vi sono delle sostanziali differenze. La prima è che sono rappresentanti di due cose diverse. Gli individui sono generiche soluzioni di un particolare problema di ottimizzazione, mentre i predatori sono dei possibili "centri di aggregazione" dell'insieme D, un unica componente di una generica soluzione di un *unsupervised data mining* problema.
Un altra differenza tra predatori ed individui è la metodologia usata per selezionare gli elementi più adatti alla riproduzione. Generalmente per gli individui si utilizza una funzione di *fitness* che indica quanto la soluzione rappresentata dall'individuo è buona a risolvere il problema di ottimizzazione. Metaforicamente la funzione di *fitness* è associata alla capacità dell'individuo di adattarsi all'ambiente. Nel caso del CED invece vi è l'associazione metaforica direttamente all'ambiente, i predatori più adatti saranno quelli che riescono a conquistare più risorse, per ogni risorsa si organizzano delle vere e proprie battaglie che simulano la concorrenza naturale tra esseri viventi, la risorsa verrà conquistata dal predatore più adatto ad essa, questo lo si fà attraverso la funzione di *hunter-fitness*.

2.1.2 Risultati del PDM

Come risultato finale il PDM estrae da un insieme di possibili predatori P un sottoinsieme finito $S \equiv \{p^1, p^2, ..., p^k\}$ che risultano essere una buona soluzione del seguente problema:

- La determinazione di $S \subset P$ con k elementi che minimizza la seguente funzione:

$$\sum_{x^i \in D} min_{p \in S}(hf(p,x)) \tag{2.3}$$

- IL valore più vantaggioso per k.

Il secondo punto è ambiguo, ma è equivalente al problema di determinare il numero di *cluster* in un problema di *data clustering*. In effetti la soluzione di un problema di *data clustering* può essere vista come il caso particolare della soluzione soprascritta:

- Determinare un insieme finito C di possibili centroidi con K elementi che minimizzano al seguente funzione:

$$\sum_{x \in D} min_{c \in C}(d(c,x)) \tag{2.4}$$

  dove d è la metrica utilizzata.

- Determinare il più vantaggioso valore di K.

In questo caso la funzione di *hunter-fitness* è il criterio di similitudine o metrica nello spazio D con cui si vuole effettuare l'operazione di *clustering* mentre K il numero di *Cluster*[1].

Ogni punto è assegnato a un predatore, quello, appartenente a S con il valore minore della funzione *hf*. Questo permette una classificazione dell'insieme di dati. Nel caso in cui si utilizzano come predatori centroidi e come funzione *hf* una metrica definita in D la soluzione trovata risulta essere una buona soluzione di un problema di *data clustring*.

La scelta del predatore determina la specie di informazioni che si vogliono estrarre. Predatori di tipi differenti permettono di estrarre informazioni differenti e di raggruppare i dati con criteri molto più generici ed ampi rispetto a semplici criteri di somiglianza.

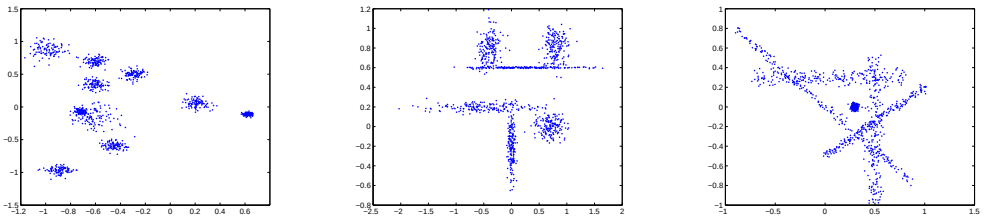
Nel prossimo paragrafo verranno illustrati i risultati del PDM su alcuni semplici insiemi di dati con differenti tipi di predatori.

[1]Nella maggior parte degli algoritmi di *data clustering* si fissa un valore di ingresso, per esempio il numero di cluster, proprio per eliminare l'ambiguità del secondo punto

2.1.3 Utilizzo del PDM: Esempi

Per semplificare il problema e per permettere una facile visualizzazione dei dati e dei predatori, saranno usati tre insieme di dati in $\mathbb{R}^2$. Gli insieme di dati sono generati ad hoc per enfatizzare, nei vari casi, i vantaggi ed i svantaggi dell'utilizzo delle diverse classi di predatori. È importante evidenziare che le operazioni del PDM sono identiche in tutti gli esempi.

Verranno utilizzati tre differenti insiemi di dati: $D_i \subset \mathbb{R}^2$ con $i = 1,2,3$ mostrati rispettivamente nelle figure 2.2(a), 2.2(b), 2.2(c)



(a) Primo data set (b) Secondo data set (c) Terzo data set

Figura 2.2: Insieme di dati composti da punti di un piano

il generico dato $d \in D$ è una coppia (d_x, d_y), si applicherà ad essi il PDM utilizzando tre classe di predatori differenti:

- P_1 centroidi
- P_2 ellissi con area costante
- P_3 rette

Primo esempio di predatori: Punti (Centroidi)

Il primo esempio è il più semplice da immaginare, i predatori sono omogenei ai dati, cioé sia i predatori che le loro risorse (i dati) sono dei punti di $\mathbb{R}^n$ in questo caso $n = 2$.
Più specificatamente abbiamo:

- Un generico predatore p è rappresentato da una coppia di numeri: (p_x, p_y)

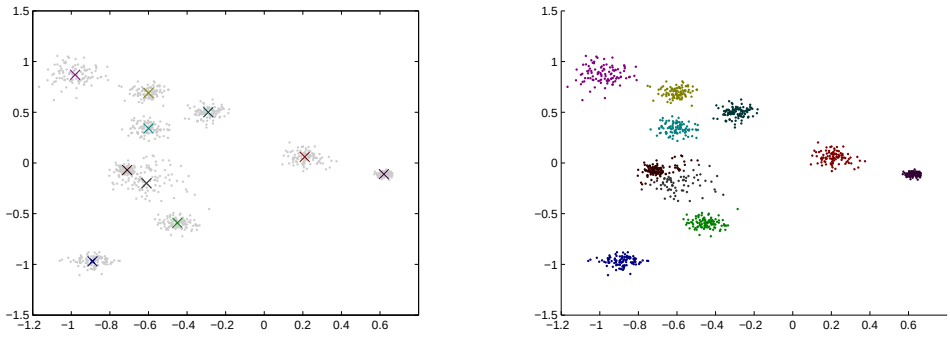
- L'operatore di inizializzazione estra casualmente una coppia di numeri appartenenti all'intervallo $[-1,1]$.
- L'operatore di mutazione genera un secondo predatore

$$p' = (p_x + \delta_x, p_y + \delta_y) \tag{2.5}$$

  dove δ_i è 0 con probabilità $1 - P_{Mut}$ ed è , con probabilità P_{Mut}, una quantità aleatoria con distribuzione gaussiana centrata in 0 e con deviazione standard σ_{Mut} . Questi due paremetri danno la probabità che un punto "muta", in questo caso che si "sposta" nel piano.
- La *hunter-fitness* è la distanza euclidea

$$hf(p,d) = \sqrt{(p_x - d_x)^2 + (p_y - d_y)^2} \tag{2.6}$$

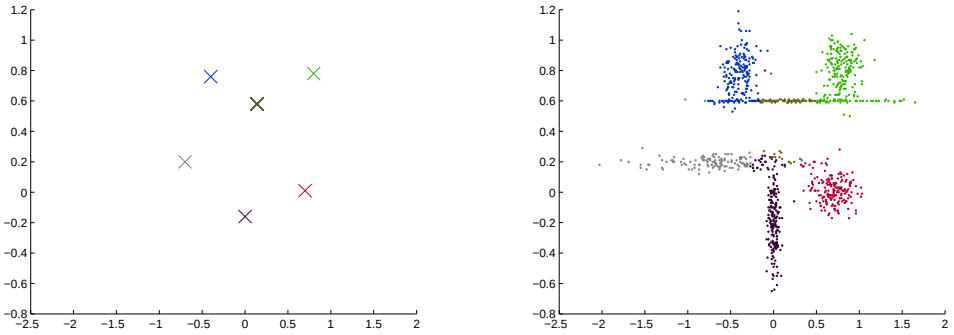
In questo caso i migliori predatori estratti dal PDM sono una buona soluzione di un problema di *data clustering* . Si possono visualizzare i predatori scoperti dal PDM nelle figure 2.3(a), 2.4(a), 2.5(a) rispettivamente per i tre data sets mentre le classificazioni nelle figure 2.3(b), 2.4(b), 2.5(b).



(a) Punti predatori

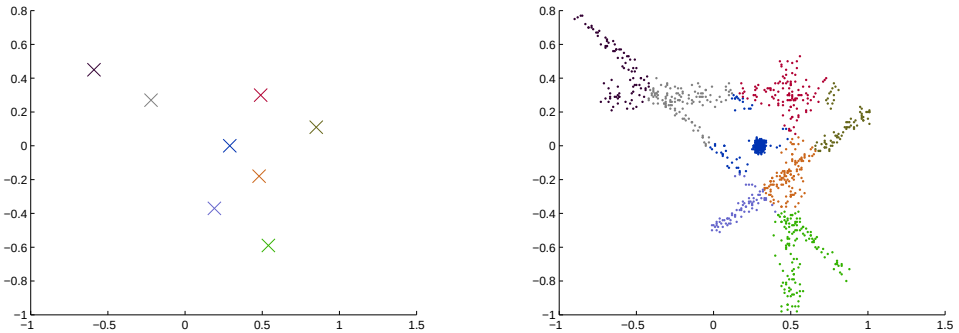
(b) Classificazione

Figura 2.3: Punti predatori selezionati dal PDM e classificazione risultante su primo data set



(a) Punti predatori (b) Classificazione

Figura 2.4: Punti predatori selezionati dal PDM e classificazione risultante su secondo data set



(a) Punti predatori (b) Classificazione

Figura 2.5: Punti predatori selezionati dal PDM e classificazione risultante su terzo data set

Secondo esempio di predatore: ellissi con area costante

Nel secondo esempio i predatori non sono singoli punti, ma delle ellissi con area costante. L'ellisse ha gli assi orientati lungo l'asse x e y dell'insieme dei dati.

- Un generico predatore p è rappresentato da 4 parametri (generalmente $2n$) $\{(c_1,c_2),(a_1,a_2)\}$ che rappresentano il centro e la lunghezza delle assi dell'ellisse, tali parametri sono vincolati dalla relazione $a_1a_2 = 1$ in modo da rappresentare ellissi con area costante. Il predatore ha quindi tre parametri liberi (in generale 2n-1). Il valore dell'area non influenza il risultato.

- L'operatore di inizializzazione estrae la coppia di numeri (c_1,c_2) nell'intervallo $[-1,1]$ e la coppia (a_1,a_2) mell'intervallo $(0,1]$ successivamente la seconda coppia viene normalizzata $a_i = a_i/\sqrt[n]{\prod_{i=1}^{n} a_i}$.

- L'operatore di mutazione genera un secondo predatore

$$P' = \{(c_1+\delta_1, c_2+\delta_2),(a_1+\delta_3, a_2+\delta_4)\} \tag{2.7}$$

  dove i δ_i sono 0 con probabilità $1-P_{Mut}$ e con probabilità P_{Mut} sono una quantità aleatoria con distribuzione gaussiana centrata in 0 e con deviazione standard: σ_{Mut}. Se, come risultato di questa operazione, uno dei semiassi a_i diventa negativo o comunque minore di una piccola quantità ε allora lo si pone uguale ad ε. Infine per completare la mutazione gli a_i sono normalizzati come sopra.

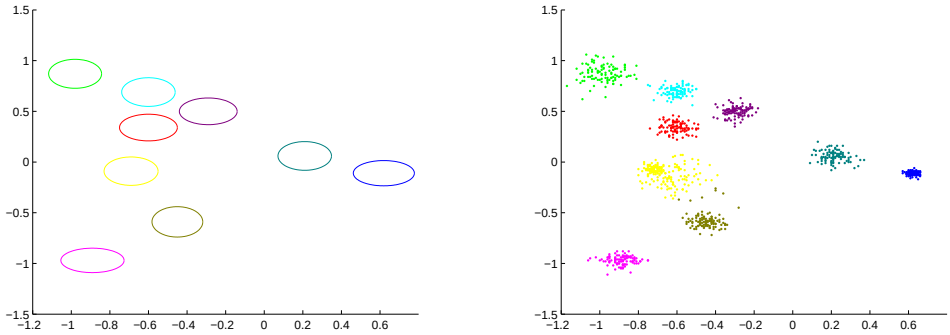
- La funzione di *hunter-fitness* è definita nel seguente modo

$$hf(p,d) = \sqrt{\sum_{i=1}^{n} \frac{(c_i-d_i)^2}{(a_i)^2}} \tag{2.8}$$

  Questa corrisponde geometricamente alla distanza tra il punto ed il centro dell'ellisse, dopo pero aver effettuato una dilatazione in una direzione e contrazione nell'altra che trasforma le ellissi (o iperelissoidi) in cerchi (o ipersfere)[2].

[2]L'utilizzo di una costante differente per l area dell'ellisse non influisce sulla relazione d'ordine che per ogni dato d la hf effettua nell'insieme P. Per questo i risultati risulteranno indipendenti dal valore di questa costante.

Per semplificare la visualizzazione dei predatori rispetto ai dati, fissiamo la costante dell'area delle ellissi a 0.01. Nelle figure 2.6(a), 2.7(a), 2.8(a) si vedono i predatori, differenziati tramite colori, e le corrispondenti classificazioni sono mostrate nelle figure 2.6(b), 2.7(b), 2.8(b)



(a) Ellissi predatori

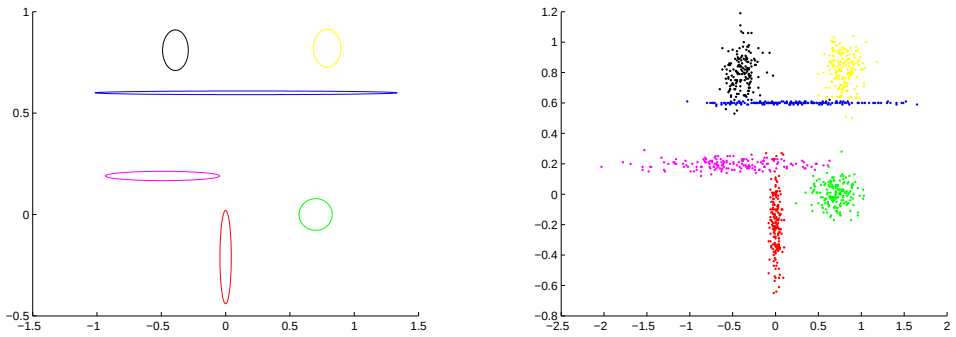
(b) Classificazione

Figura 2.6: Ellissi predatori selezionati dal PDM e classificazione risultante su primo data set

Terzo esempio di predatori: Rette

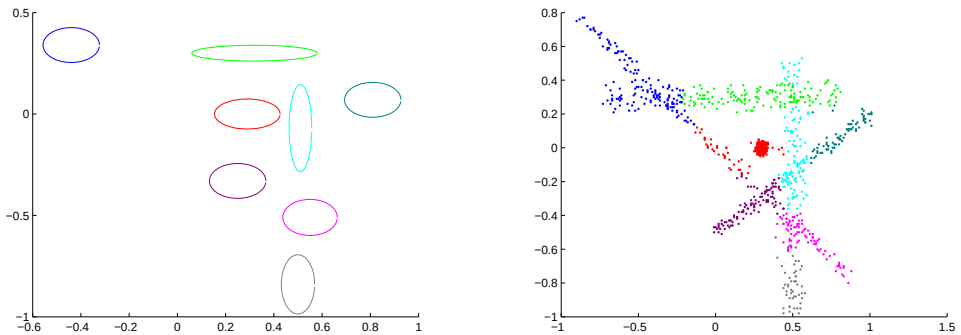
In questo caso i predatori sono rette, non sono quindi oggetti locali, sono caratterizzati nel seguente modo:

- Il generico predatore è rappresentato da quattro reali: $(\alpha_1, \beta_1, \alpha_2, \beta_2)$ (in generale $2n$ valori) rappresentanti una retta parametrizzata come $(x(t) = \alpha_1 + \beta_1 t, y(t) = \alpha_2 + \beta_2 t)$. (in generale $x_i(t) = \alpha_i + \beta_i t$)
- L'operatore di inizializzazione estrae casualmente i 4 parametri nell'intervallo $[-1, 1]$ generando con essi il predatore.
- L'operatore di mutazione agisce nel solito modo, addizionando con una certa probabilità un rumore gaussiano alle componenti.



(a) Ellissi predatori (b) Classificazione

Figura 2.7: Ellissi predatori selezionati dal PDM e classificazione risultante su secondo data set



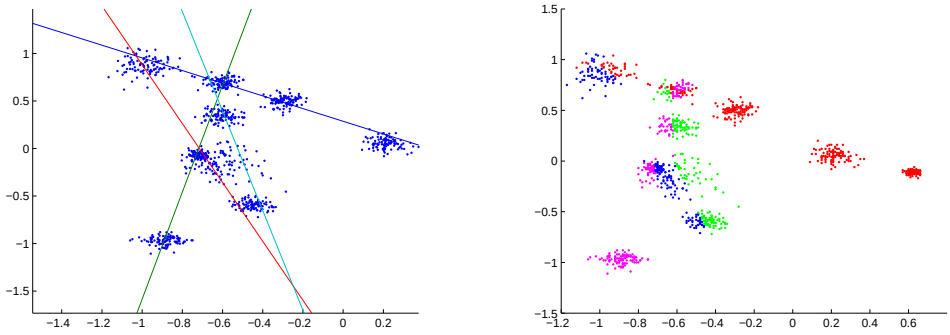
(a) Ellissi predatori (b) Classificazione

Figura 2.8: Ellissi predatori selezionati dal PDM e classificazione risultante su terzo data set

- La hunter-fitness è la distanza tra una retta (predatore) e punto (risorsa/dato) in due dimensioni possiamo definirla cosi:

$$d(p,d) = \frac{|\frac{1}{\beta_1}d_x + \frac{1}{\beta_2}d_y + (\frac{\alpha_2}{\beta_2} - \frac{\alpha_1}{\beta_1})|}{\sqrt{(\frac{1}{\beta_1})^2 + (\frac{1}{\beta_2})^2}} \tag{2.9}$$

Nelle figure 2.9(a), 2.10(a), 2.11(a) possiamo vedere le rette scoperte dal PDM e nelle figure 2.9(b), 2.10(b), 2.11(b) le rispettive classificazioni.



(a) Rette predatori

(b) Classificazione

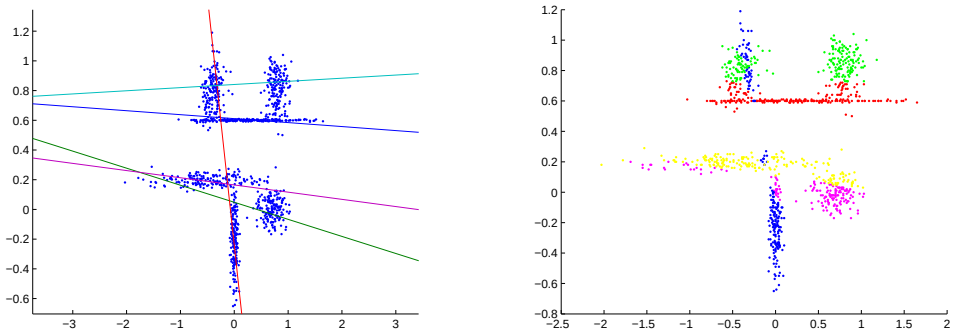
Figura 2.9: Rette predatori selezionati dal PDM e classificazione risultante su primo data set

2.2 L'algoritmo

L'algoritmo può essere diviso in sezioni, precisamente possiamo separare il PDM nei seguenti passi:

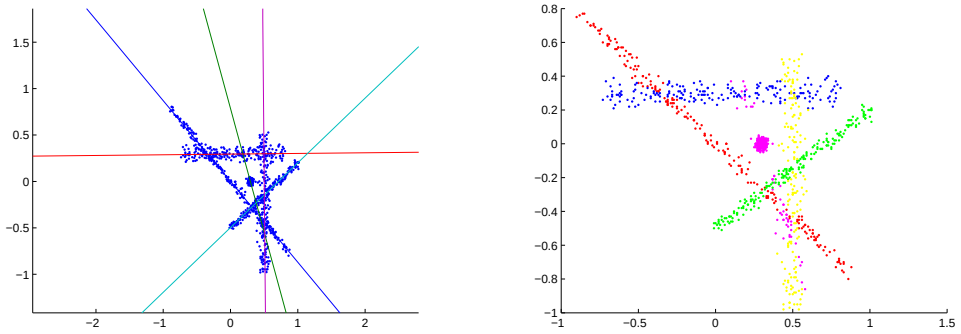
1. Il CED: un algoritmo evolutivo che crea popolazioni di predatori.(§2.2.1)

2. Un algoritmo di data clustering ad hoc che attraverso una metrica, intrinsicamente definita nel PDM ed indipendente dalla struttura di predatori e dati, estrae, da una popolazione di predatori, quelli che ritiene più significativi, classificando i dati rispetto a questi. (§2.2.2)



(a) Rette predatori (b) Classificazione

Figura 2.10: Rette predatori selezionati dal PDM e classificazione risultante su secondo data set



(a) Rette predatori (b) Classificazione

Figura 2.11: Rette predatori selezionati dal PDM e classificazione risultante su terzo data set

3. Un algoritmo di *fusion clustering* che da diverse classificazioni dello stesso insieme di dati (ottenute estraendo i migliori predatori da diverse popolazioni del CED) determina quella finale ed un informazione aggiuntiva che indica la stabilità di questa classificazione.(§2.2.3)

2.2.1 CED: *Competitive Evolution on Data*

La prima operazione fatta dal PDM è la creazione di un insieme di popolazioni di predatori, che saranno dopo analizzate. A questo scopo è stato sviluppato un algoritmo evolutivo chiamato *Competitive Evolution on Data* (CED).

Il CED come accennato in §2.1.1 si differenzia dai classici paradigmi evolutivi, (§1.3) sia per l'assenza di una funzione di fitness, ma sopratutto, per lo scopo. La funzione di *hunter-fitness*, infatti, relaziona il generico dato al predatore, cioé formalizza la bontà del predatore rispetto al singolo dato, ma non dice nulla di quanto è buono il predatore rispetto alle soluzioni cercate. In effetti la dinamica evolutiva organizzata nel CED somiglia più alle dinamiche usate nei sistemi immunitari artificiali [29] che nei classici paradigmi evolutivi.
Riguardo allo scopo il CED non cerca una soluzione ad un problema ma è usato solo per creare popolazioni di predatori, in altri termini lo scopo del CED è creare dei data set differenti da quello di partenza, sui quali verranno effettuate analisi con tecniche di *data clustering*.

Con questo tipo di dinamica evolutiva i predatori non convergeranno ad un ottimo globale ma si specializzeranno cercando nicchie ambientali particolarmente vantaggiose, tenteranno di essere molto "buoni" solo per un gruppo ristretto di risorse(dati).

Più in dettaglio, chiamata P_g la generica popolazione di predatori, il CED effettua le seguenti operazioni:

- Crea una popolazione iniziale di predatori: P_0, attraverso l'operatore iniziale della classe predatore.

- Da una generica popolazione P_g, alcuni predatori sono selezionati con una certa procedura. Questi generano, attraverso l'operatore di mutazione o una semplice copia, una nuova popolazione: P_{g+1}.
- Il processo continua fino al raggiungimento di una generazione massima fissata a priori.

Procedura di selezione

La selezione è fatta con un processo probabilistico proporzionale al numero di risorse che il predatore riesce a "conquistare". Questo numero viene stabilito dalla seguente procedura:

- Ad ogni generazione g, un sottoinsieme $R_g \subseteq D$ viene estratto casualmente dal data set. Il numero di elementi di R_g viene posto pari al numero di predatori nella popolazione
- Similarmente per ogni $d \in R_g$ si estrae casualmente un sottoinsieme $T_d \subset P_g$ della popolazione corrente di predatori. Il numero dei predatori di questo torneo è posto pari a $1 \setminus 6$ del numero di predatori della popolazione (e quindi dei dati in R_g)
- Per ogni dato $d \in R_g$ è assegnato un predatore in $T_d \subset P_g$. Il predatore assegnato è quello con il valore migliore (più piccolo) della *hunter-fitness*. Praticamente, per ogni dato si organizza un torneo, il vincitore del torneo sarà, tra i partecipanti, il predatore più adatto al dato d. I predatori non parteciperanno a tutti i tornei ma solo ad una percentuale di essi. Ovviamente ci saranno dei predatori che conquisteranno più risorse, mentre altri non ne conquisteranno nessuna.

I parametri che regolano la dinamica evolutiva sono i seguenti:

n_p Il numero di predatori nella popolazione.

n_r Il numero di dati che rappresentano le risorse a disposizione dei predatori nella corrente generazione, cioé il numero di dati di R_g.

n_T Il numero di predatori che partecipano ad ogni singolo torneo.

Negli esempi che verranno riportati i parametri sono stati fissati nella seguente maniera: $n_p = n_r = 6n_t$ in §3.2 si utilizzerà sempre $n_t = 50$, mentre invece in 3 e 4 è stato usato $n_t = 100$.

Il costo computazionale è alto, dipende dall'elevato numero di volte che viene richiamata l'*hunter-fitness*: numero di generazioni * n_r * n_t, ma non dipende da N. Considerando che il numero massimo di generazione utilizzato è di circa 8000 questa funzionalità è generalmente la più costosa.

2.2.2 Analisi della popolazione di predatori

Il CED crea le popolazioni di predatori simulando un processo evolutivo in cui predatori competono per conquistare le risorse dell'ambiente, le risorse come già detto sono i dati dell'insieme da analizzare. La probabilità che un singolo predatore ha di essere selezionato per generare eredi nella popolazione successiva dipende dal numero di risorse che riuscirà a conquistare, questo dipenderà anche dal numero di predatori che concorrono con lui sulle stesse risorse. Quando un predatore conquista molte risorse sarà probabilmente selezionato più volte per la copia e la mutazione, conseguentemente nella generazione successiva ci saranno molti predatori simili o identici che competeranno sulle stesse risorse. Il risultato di questo processo è che alla fine i predatori migliori non saranno quelli che conquisteranno più risorse ma quelli che avranno il numero maggiore di concorrenti, di predatori cioè che competono con loro similmente sulle stesse risorse.

Se, per predatori simili si intende quelli che competono similmente sulle stesse risorse, e si formalizza il concetto con un criterio matematico, è possibile distinguere i migliori predatori come i centroidi di cluster di predatori. Si trasforma quindi il problema de per questo è stata definita una nuova metrica nello spazio dei predatori. Partendo dall'insieme dei dati originali attraverso la *hunter-fitness* tra predatore e dato è stata costruita una metrica indipendente dalla particolare codifica sia dei predatori che dei dati.

Criterio di somiglianza tra predatori

Due predatori sono vicini se competono per le stesse risorse. Si potrebbe definire una metrica utilizzando la forma codificata dei predatori (genotipo), ma, non è sempre vero che predatori che hanno genotipi simili competono sulle stesse risorse ne tantomeno è vero che per competere sulle stesse risorse devono necessariamente avere genotipi simili (per esempio iene e avvoltoi). Inoltre il PDM vuole essere un algoritmo generale, si vuole definire una metrica indipendentemente dal tipo di struttura del predatore e per alcune codifiche complesse, si immagini ad esempio i classici alberi che rappresentano il genotipo nella programmazione genetica [27], sarebbe difficoltoso definire una distanza [65].
Questo suggerisce di definire la distanza tra predatori attraverso la *hunter-fitness* che regola la competizione sulle risorse. Si introduce una metrica tra predatori che è indipendente dalle codifiche e soddisfa la richiesta che due predatori saranno vicini se competono similmente sulle stesse risorse.
Ad ogni predatore p si associa un vettore $c_p \in \mathbb{R}^N$ dove la i^{th} coordinata è il valore della hunter-fitness tra il predatore e l'i^{th} dato dell'insieme:

$$c_{p_i} = hf(p, x^i) \tag{2.10}$$

Questo vettore codifica la bontà del predatore rispetto a tutti i dati, due predatori che hanno associato lo stesso vettore sono, per il PDM, equivalenti, indipendentemente se hanno o no genotipi identici. Dati due predatori p_1 e p_2 si definisce una distanza attraverso i loro vettori associati, che chiameremo $c1$ e $c2$, nella seguente maniera:

$$d(p_1, p_2) = 1 - \sigma_{c1,c2} \tag{2.11}$$

dove $\sigma_{c1,c2}$ è il coefficiente di relazione di Pearson centrato tra i vettori $c1$ e $c2$:

$$\sigma_{c1,c2} = \frac{\sum_{i=1}^{N}(c1_i - mean(c1))\sum_{i=1}^{N}(c2_i - mean(c2))}{\sqrt{\sum_{i=1}^{N}(c1_i - mean(c1))^2 \sum_{i=1}^{N}(c2_i - mean(c2))^2}} \tag{2.12}$$

La scelta di questa metrica è arbitraria, si potrebbe scegliere anche, per esempio, la metrica Euclidea.

Il costo computazionale di questa metrica dipende linearmente da N, inoltre, per preparare i vettori associati ai predatori bisogna ripetere per ogni N volte la *hf*, quest'aspetto con insieme di grosse dimensioni di dati appesantirebbe molto ed anche abbastanza inutilmente il costo computazionale. Per questo motivo per $N > 1000$ si estrae in maniera casuale un sottoinsieme di D con 1000 elementi e su questo si ricavano i vettori associati ai predatori.

Determinazione dei migliori predatori da una popolazione

Scopo del PDM non è raggruppare i predatori appartenenti alle popolazioni ma bensì determinare i migliori rappresentanti di questi gruppi.

La tecnica utilizzata è basata su considerazione di densità spaziale. Si sceglie un reale positivo α ed un intero m_c e si procede come segue:

- Da un sottoinsieme $P_* \subseteq P$ della popolazione, inizialmente coincidente con P viene scelto il "miglior predatore" $p_i \in P_*$, i.e. il predatore col maggior numero di $p_j \in P_*$ tali che $d(p_i, p_j) < \alpha$ $(i \neq j)$. Chiamato questo numero n_α si ha che:

- se $n_\alpha \geq m_c$

  - p_i è uno dei *predatori significativi* .
  - $P_* = P_* - \{p_j : d(p_i, p_j) < \alpha\}$
  - Si continua la ricerca di un altro predatore significativo.

- quando $n_\alpha < m$ la ricerca termina.

Alla base di quest'algoritmo, come usualmente succede per gli algotritmi basati sulla densità (§1.2.3) vi sono due parametri, α e m_c che corrispondono a ε e $MinPts$ introdotti in 1.2.3. In generale i risultati sono stabili a piccole variazioni di m_c che viene scelto tra il 2 e il 3% circa della popolazione. Se, comunque, si utilizza un valore di m_c troppo alto, si corre il rischio di perdere traccia di predatori associati a classi poco numerose, mentre invece se è troppo piccolo, si rischia di considerare significativi predatori che non lo sono. Per scoprire predatori associate

a classi poco numerose bisogna comunque diminuire m_c una soluzione, per mantenere la significatività dei predatori estratti è aumentare il numero di individui della popolazione, anche se in questo caso si aumentano i costi computazionali dell'algoritmo, per questo motivo nei capitoli 3 e 4 vengono raddoppiati il numero di predatori appartenenti ad una popolazione.

La scelta del parametro α è più delicata. Il suo valore può influire sul numero di predatori selezionati e quindi sulle dimensione ed il numero dei cluster. Un buon valore di α è quello che mette assieme i predatori che competono sullo stesso gruppo di risorse, ovviamente questo valore dipende dalla particolare applicazione oltre che dal gruppo in questione. Non è possibile individuare il miglior α rispetto al gruppo di risorse ma si può provare a valutare un buon valore di $alpha$ rispetto al problema analizzato. La metodologia adottata è determinare questo valore attraverso un analisi visuale della distribuzione delle distanze tra i predatori della popolazione.

L'idea alla base di quest'analisi è che, l'eventuale presenza di classi ben discriminabili, è metaforicamente traducibile con l'esistenza di nicchie ambientali all'interno delle quali un numero più o meno grande di predatori sopravvive. La distanza tra i predatori appartenenti alla stessa nicchia è più piccola rispetto alla distanza tra predatori appartenenti a nicchie diverse e in questa ipotesi determinabile orientativamente visualizzando l'istogramma delle distanze tra i predatori.

Nelle figure 2.12 e 2.13 si possono vedere due esempi della distribuzione delle distanze tra i predatori in due differenti popolazione. La prima riferita al caso della figura 2.3(a), la seconda a 2.9(a).

In entrambi i casi si notano nettamente due minimi vicino allo 0 che vengono orientativamente utilizzati per fissare il parametro α. La presenza di questo minimo non è necessariamente causata dall'esistenza di nicchie ambientali ben separate ma dipende anche dalla dinamica evolutiva adottata. Ad ogni generazione si mantengono inalterati molti dei predatori migliori della generazione precedente e questo agevola la presenza in una popolazione di molte coppie di predatori identici.

Una volta determinati i migliori predatori si può classificare i dati D rispetto

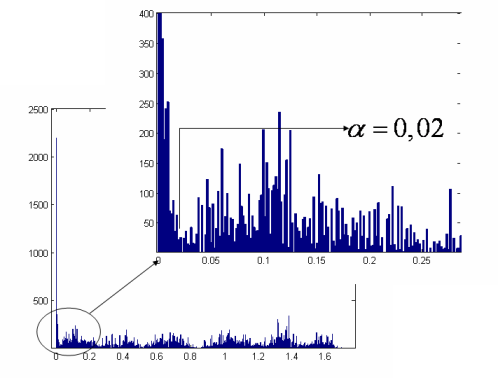


Figura 2.12: Distanze tra predatori dell'esempio raffigurato in 2.3(a)

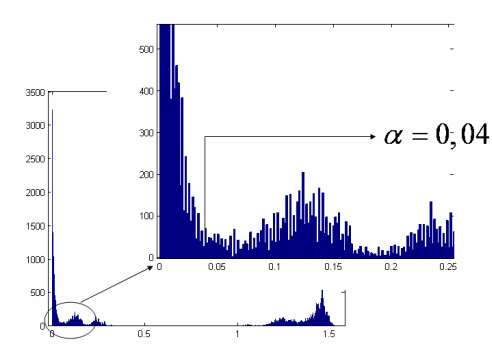


Figura 2.13: Distanze tra predatori dell'esempio raffigurato in 2.5(a)

a questi, ovviamente questa classificazione dipenderà dalla popolazione P da cui sono stati estratti i predatori per questo la chiameremo C_P.

Computazionalmente il numero di volte che si ripete la distanza tra i predatori è proporzionale ad n_p^2. Rispetto al CED, anche quando la distanza può risultare computazionalmente più dispendiosa della *hf*, risulta meno costosa, perché il numero di popolazioni analizzate è di un paio d'ordini di grandezza inferiore al numero di generazioni.

2.2.3 Metodologia di scelta tra classificazioni e stima dell'errore commesso.

Essendo le popolazioni ottenute dal CED diverse, le soluzioni che si estraggono da queste saranno diverse e quindi la classificazione finale dipenderà dalla popolazione scelta per determinare i centroidi, però più i dati di base sono discriminabili in cluster meno i risultati dipenderanno da questa scelta.
Si è osservato che la popolazione di predatori si adatta abbastanza velocemente all'ambiente in cui vive ma la dinamica evolutiva ha un intrinseca componente rumorosa che bisogna prendere in considerazione.
La pressione della selezione porta i singoli predatori a cercare di migliorarsi rispetto alla popolazione corrente ma questo, dato che il cambiamento avviene in par-

allelo fra tutti i predatori, non implica che la popolazione successiva sia migliore di quella corrente. Cioé, anche se la pressione della selezione in generale è attratta dai migliori "centri aggregativi", la mutua interazione tra i predatori aggiunge una pressione alla selezione di tipo rumoroso. La componente rumorosa della pressione alla selezione è qualcosa di intrinseco alla dinamica e non dipende dall'insieme di dati analizzato, mentre, l'altra componente è più o meno forte a secondo di quanto buoni e individuabili sono i migliori centri di aggregazione. Le oscillazioni fra i risultati determinati tra le popolazioni diverse dipendono dal rapporto tra le due pressioni alla selezione, essendo queste, una dipendente e l'altra indipendente dai dati, le oscillazioni danno un informazione aggiuntiva sull'insieme dei dati: più queste sono piccole più significa che, rispetto al predatore utilizzato, l'insieme dei dati è discriminabile.

L'avere classificazione diverse è in questo senso un vantaggio. Del resto, negli ultimi anni si sono sviluppati diversi algoritmi che partendo da classificazioni diverse dello stesso insieme ottengono partizioni del data base più robuste e meno dipendenti dalle inizializzazioni [63]. Il Tight Clustering [58], ad esempio, è un algoritmo che lavora su diverse classificazioni dello stesso insieme di dati. Generalmente, ma non necessariamente, le classificazioni sono ottenute attraverso il k-means, in [53] le diverse classificazioni analizzate sono ottenute applicando lo stesso algoritmo con diversi parametri iniziali, lo scopo è determinare i migliori parametri iniziali ed in particolare il numero di cluster corretti.

Il PDM da una certa generazione: gen_{start}, che si ritiene sufficientemente avanzata, prende un certo campione di popolazioni: $\{P_i\}_{i=1..n_c}$ con $n_c <= max_{gen} - gen_{start}$, su queste estrae i migliori predatori, come spiegato in 2.2.2, ed opera una classificazione dei dati di partenza, ottenendo cosi un insieme di classificazioni: $\{C_i\}_{i=1..n_c}$ dello stesso insieme di dati. Come anche fatto in [53] si sceglie la classificazione che meglio coincide con le altre.

Per confrontare la coincidenza tra le classificazioni si usa lo stesso coefficiente utilizzato in [53] , un evoluzione del rand index coefficient [44] che chiameremo adjusted rand index sviluppato in [62] calcolato tramite l'equazione 3.1 riportata in 1.4.4.

In pratica per ogni classificazione C_i si calcola l'adjusted rand index medio, chiamato Rnd_{mean}, rispetto alle altre

$$Rnd_{mean}(C_i) = \frac{1}{n_c - 1} \sum_{j=1..n_c \wedge i \neq j} RandAdj(C_i, C_j) \tag{2.13}$$

e si sceglie come risultato finale la classificazione, che chiameremo C_{PDM}, con tale valore più vicino a 1.

$$C_{PDM} = C_k : k = arg(max_{i=1..n_c}(Rnd_{mean}(C_i))) \tag{2.14}$$

il valore $Rnd_{mean}(C_k)$ è una misura della stabilità del risultato che chiameremo: Ac_{PDM}, dato che in qualche modo può essere vista come una misura di accuracy della classificazione ottenuta dal PDM

$$Ac_{PDM} = max_{i=1..n_c}(Rnd_{mean}(C_i)) \tag{2.15}$$

.

Computazionalmente sia la classificazione dei dati sia il confronto fra classificazioni dipendono da N, ma linearmente.

2.3 Cenni sull'implementazione del PDM

Il PDM è stato implementato attraverso la cosidetta "programmazione generica". Questa consente di applicare lo stesso codice a tipi diversi, cioé di definire template (modelli) di classi e funzioni parametrizzando i tipi utilizzati: nelle classi, si possono parametrizzare i tipi dei dati-membro; nelle funzioni (e nelle funzioni-membro delle classi) si possono parametrizzare i tipi degli argomenti e del valore di ritorno. In questo modo si raggiunge il massimo di indipendenza degli algoritmi dai dati a cui si applicano: per esempio, un algoritmo di ordinamento può essere scritto una sola volta, qualunque sia il tipo dei dati da ordinare.

I template sono risolti staticamente (cioé a livello di compilazione) e pertanto non comportano alcun costo aggiuntivo in fase di esecuzione; sono invece di

enorme utilità perché permettono di scrivere del codice "generico", senza doversi preoccupare di differenziarlo in ragione della varietà dei tipi a cui tale codice va applicato. Ciò è particolarmente vantaggioso quando si possono creare classi strutturate identicamente, ma differenti solo per i tipi dei membri e/o per i tipi degli argomenti delle funzioni-membro.

Nel nostro caso i template rispondono perfettamente alla natura generica del PDM, i tipi parametrizzati sono la struttura e le funzioni associate al predatore ed alla risorsa. Il PDM è stato implementato in C++ in ambiente windows (Dev C++), predatore e risorsa sono classi scritte in C++ e sono degli ingressi del PDM. Ovviamente, queste classi, devono avere alcuni metodi particolari.

2.3.1 Classi predatore e risorsa

La classe predatore deve avere i seguenti metodi:

- 3 costruttori, uno senza argomento che crea un predatore casualmente: utilizzato per generare la popolazione iniziale, uno che ha come argomento una stringa all'interno della quale deve essere scritto il genotipo del predatore ed infine un altro che prende come ingresso un oggetto predatore e lo costruisce copiandolo.
- un metodo chiamato mutazione che "muta" il genotipo.
- un metodo chiamato scrivi che ha come ingresso una stringa su cui scrive il genotipo del predatore, stringa che eventualmente servirà come ingresso al costruttore di un altro predatore
- un metodo chiamato hunter-fitness che prende come ingresso un oggetto risorsa e ritorna un reale positivo.

La classe risorsa invece deve avere un costruttore senza argomento e due metodi uno per la lettura e l'altro per la scrittura su FILE.

Al PDM non sono necessarie altre informazioni, in altri termini la struttura dei dati, del predatore e dell'hunter-fitness sono dei parametri.

2.3.2 Il PDM

Il primo passo del PDM è, attraverso uno dei suoi metodi: *LeggiRisorse*, caricare in memoria a tabella (un file di testo) che rappresenta il nostro insieme di dati. L'operazione viene fatta attraverso il metodo leggi dell'oggetto risorsa, in pratica il PDM carica un vettore di N risorse, N al solito è il numero di pattern del data base, in questo caso della tabella. Per N molto grandi (10^5) non è possibile caricare in memoria tutto il data base, questa operazione però non è necessaria e solo più comoda.

Le funzionalità principali del PDM sono le seguenti:

CED Il metodo che implementa la dinamica evolutiva, questo memorizzera una serie di popolazioni di predatori in file di testo ed in un altro le distanze tra questi predatori.

AnalisiPopolazioni Il metodo che estrae dalle popolazioni memorizzate dal CED i predatori migliori e le rispettive classificazioni, sia gli uni che gli altri vengono memorizzati su file di testo.

FusionClustering Il metodo che permette di selezionare una sola classificazione fra quelle prodotta da AnalisiPopolazione e di determinare l'Ac_{PDM}.

In figura 2.14 possiamo vedere lo schema generale. I moduli del PDM si scambiano le informazioni attraverso file esterni questo permette di poterli utilizzare indipendentemente l'uno dall'altro.

2.3.3 Complessità computazionale

La complessità computazionale del PDM dipende da diversi parametri il numero di pattenr dell'insieme di dati , il numero di predatori delle popolazioni, il numero di risorse che vengono assegnate ad ogni generazione, il numero di partecipanti ai tornei, il numero di generazioni ma sopratutto è cruciale la complessità dell'hunter-fitness. Supponendo come suggerito in §2.2.1 che alcuni dei parametri di ingresso sono correlati $n_p = n_r = 6n_t$ possiamo identificare 4 parametri principali che regolano i tempi di computazione.

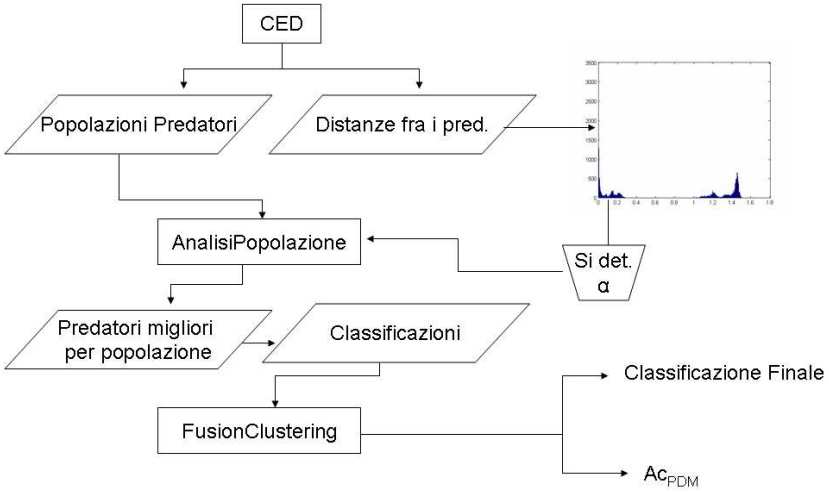


Figura 2.14: Schema moduli PDM

- Maxgen il numero massimo di generazioni del CED.
- n_p il numero di predatori della popolazione
- N il numero di dati del data base
- la complessità della *hunter-fitness* che chiameremo $O(hf)$.

La funzionalità che ha il maggior costo computazionale è generalmente il CED, ma non dipende da N. Rispetto agli altri algoritmi i costi computazionali, specialmente su insieme di dati di piccole dimensioni, sono generalmente molto più alti, ma comunque i tempi sono più che accettabili a meno di non usare delle funzioni di *hunter-fitness* computazionalmente complesse.

Capitolo 3

Risultati su dati simulati

3.1 Introduzione

Il CED può essere applicato ad una ampia categoria di problemi di data mining unsupervised, una completa investigazione delle possibili applicazioni richiederebbe troppo spazio ed in questa tesi l'intento è verificare più la bontà del modello che le sue possibili applicazioni. Per prima cosa in 3.2 si andrà nel dettaglio dei risultati visualizzati in 2.1.3, poi ci concentra su un applicazione reale dove è possibile confrontare i risultati del CED con altri algoritmi. L'applicazione scelta è l'analisi delle espressione genetiche dei Microarray .

3.1.1 Tecnica di validazione degli algoritmi

Le tecniche di Supervised Data Mining necessitano di un insieme preclassificato e sono validate su questo; pubblici data sets, conosciuti in ambito scientifico, sono convenzionalmente usati allo scopo di confrontare le performance di un algoritmo rispetto ad altri. La tecnica di validazione in questo caso utilizza parte dell'insieme preclassificato come insegnante (*training set*) e la restante parte per validare i risultati ottenuti (*validation set*).

Questo metodo e questi insieme non sono applicabili a validare tecniche di unsupervised data mining, in particolare, tecniche di data clustering: infatti lo scopo del data clustering è raggruppare i dati rispetto ad un criterio di somiglianza.

Questa operazione è indipendente dalla presenza di un insieme preclassificato e conseguentemente è inutile a validare su questi.

Due differenti approcci sono usati per validare algoritmi di Data Mining Unsupervised: il primo usa alcuni indici che misurano quanto risulta buona l'operazione di aggregazione [61], mentre il secondo confronta i risultati estratti su insieme di dati generati con un ipotetica ed ideale classificazione di riferimento. La seconda tecnica ([64]) è adatta a valutare l'ablità a "scoprire" una certa specie di informazione dai dati, ed è quella più adatta ai nostri scopi.

L'utilizzo di insiemi di dati simulati permette di conoscere a priori l'ideale classificazione con la quale il risultato dell'algoritmo può essere comparato.

3.1.2 Confronto fra classificazioni

Il confronto tra differenti classificazioni dello stesso insieme di dati avviene attraverso l'adjusted rand index (§1.4.4) che riportiamo quì per comodità.

$$RandAdj(A,B) = \frac{\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2} - 1/\binom{n}{2}\sum_{i=1}^{C_a}\binom{n_{i+}}{2}\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}{1/2[\sum_{i=1}^{C_a}\binom{n_{i+}}{2} + \sum_{j=1}^{C_b}\binom{n_{+j}}{2}] - [1/\binom{n}{2}]\sum_{i=1}^{C_a}(\binom{n_{i+}}{2})\sum_{j=1}^{C_b}\binom{n_{+j}}{2}} \tag{3.1}$$

dove n_{ij} è il generico elemento della matrice di confusione 1.4.1, C_a a C_b sono rispettivamente il numero di cluster della classificazione A e B , n_{i+} e n_{+j} sono rispettivamente la somma della $i-th$ riga e $j-th$ colonna della matrice e corrisponde al numero di dati appartenenti rispettivamente al generico cluster delle due classificazioni. Questo coefficiente misura l'accordo nel raggruppare i dati, il valore massimo è 1 ed indica che le due classificazioni sono equivalenti, per differenti classificazioni il valore decresce, il valore 0 si ottiene quando l'accordo tra le due classificazioni è perfettamente casuale.

3.2 Risultati su esempi didattici

3.2.1 Dati simulati e classificazione di riferimento

I tre insieme dati visualizzati nelle figure 2.2(a), 2.2(b), 2.2(c) sono stati creati nella seguente maniera:

- Nel primo insieme (fig. 2.2(a)) si sono presi 10 punti: $\{(y_1^i, y_2^i)\}_{i=1..10}$ nel piano e per ognuno di questi, attraverso una funzione stocastica che aggiungeva una variabile aleatoria alle componenti del punto, si sono generati i dati dell'insieme. La variabile aleatoria ha una distribuzione gaussiana centrata in 0 ed una deviazione standard variabile per ogni punto, in modo da poter creare delle nuvole di punti più o meno dense.

$$(x_1^j, x_2^j) = (y_1^i + \sigma(0, \rho^i), y_2^i + \sigma(0, \rho^i)) \tag{3.2}$$

  dove $i = jmod(10) + 1$ e ρ^i la deviazione standard della distribuzione gaussiana della variabile aleatoria.

- Nel secondo invece (fig. 2.2(b)) si sono presi sei punti: $\{(y_1^i, y_2^i)\}_{i=1..6}$. Il procedimento è simile al precedente ma stavolta la deviazione standard della distribuzione della variabile aleatoria dipende oltre che dal punto i anche dalla componente.

$$(x_1^j, x_2^j) = (y_1^i + \sigma(0, \rho_1^i), y_2^i + \sigma(0, \rho_2^i)) \tag{3.3}$$

  In questo modo è stato possibile creare nuvole molto appiattite sia in orizzontale ($\rho_1^i >> \rho_2^i$) che in verticale ($\rho_1^i << \rho_2^i$).

- Il terzo insieme di dati (fig. 2.2(c)) è stato invece creato non fissando dei punti y^i ma bensì degli intervalli $\{(I^i)_{i=1..5}\}$ di cui uno praticamente a lunghezza quasi nulla. L'equazione usata è comunque la 3.2 con la differenza che le y^i in questo caso non sono fisse ma sono variabile aleatorie appartenenti all'intervallo I^i.

Gli insiemi di dati creati nascondono le informazioni della procedura di creazione, nel primo insieme di dati le informazioni sono i 10 punti generatori, nel secondo invece oltre ai 6 punti generatori c'è anche il rapporto esistente tra l'errore sulle componenti, sul terzo insieme invece sono i 5 intervalli. Scopo di un algoritmo di data mining unsupervised è classificare i dati mettendo in evidenza queste informazioni nascoste, in questo caso la classificazione ideale che chiameremo C_{ref} è quella che associa i dati al punto o intervallo da cui sono stati generati.

3.2.2 Analisi classificazioni ottenute dal PDM

I risultati visualizzati in 2.1.3 vengono quì analizzati. Le Classificazioni ottenute dal DPM: C_{PDM} vengono confrontate con quelle di riferimento C_{ref} determinata dalla procedura di creazione dei tre insiemi di dati 3.2.1.
Ad ogni classe predatore è associabile il tipo di informazione che si vuole estrarre dall'insieme di dati, ad esempio nei tre casi presi in esame in quest'articolo possiamo dire che il predatore punto cerca i migliori punti, il predatore ellisse cerca oltre ai punti anche le unita di misura per ogni punto migliori ed infine il terzo predatore cerca le migliori rette. Come accennato i tre insieme di dati nascondono le informazioni della procedura di creazione e conseguentemente i predatori utilizzati possono essere più o meno adatti a scoprirle. La C_{ref} inoltre non è detto che sia determinabile a posteriori anche conoscendo a priori le informazioni nascoste, ad esempio nel primo insieme di dati Fig. 2.2(a) si vedono ad occhio nudo 9 cluster distinti quando invece la C_{ref} è composta da 10 cluster, il motivo è che due dei cluster di C_{ref} si intersecano, questo vuol dire che anche conoscendo i punti generatori non si riesce a posteriori a determinare la C_{ref} essendo alcuni dei dati generati da un punto generatore più vicini ad un altro punto.
Sul primo insieme di dati (Tabella 3.1) la migliore coincidenza con C_{ref} il PDM la ottiene usato i punti come predatori in questo caso si riescono a determinare in maniera corretta 10 cluster riuscendo a separare anche quelli che si intersecano. Ottimo risultato l'ottiene anche usando come predatori le ellissi, del resto questo predatore è una generalizzazione del primo. I cluster di C_{ref} in questo caso sono di tipo sferico, usando il punto come predatore si ottengono cluster di tipo sferici

mentre usando ellissi si ottengono cluster ellissoidali (e quindi anche sferici) per questo si ottiene con entrambi una buona coincidenza con C_{ref}.
Discorso a parte meritano i risultati usando rette, le informazioni cercate sono completamente diverse da quelle nascoste rappresentate in qualche modo da C_{ref} e giustamente non ci si poteva aspettare di determinare una buona coincidenza.

Tabella 3.1: Risultati primo insieme di dati

Predator	$Rand(C_{PDM}, C_{ref})$	Number cluster	Ac_{PDM}
Point	0.9305	10	0.945
Ellipse	0.8859	9	0.86982
Line	0.2484	8	0.4944

In tabella 3.2 si possono vedere i risultati sul secondo insieme di dati, si capisce come l'informazione aggiuntiva presente nel predatore ellisse rispetto al punto diventa determinante per scoprire, in questo caso, le informazioni nascoste.

Tabella 3.2: Risultati su secondo data set

Predator	$Rand(C_{PDM}, C_{ref})$	Number cluster	Ac_{PDM}
Point	0.7552	6	0.84778
Ellipse	0.9367	6	0.83619
Line	0.4836	5	0.67722

Infine sul terzo insieme di dati (Tabella 3.3) i migliori risultati li ottiene il PDM usando come predatore delle rette in questo caso le informazioni cercate: migliori rette, sono quelle che più si avvicinano a quelle nascoste (intervalli generatori).

3.2.3 L'Ac_{PDM}

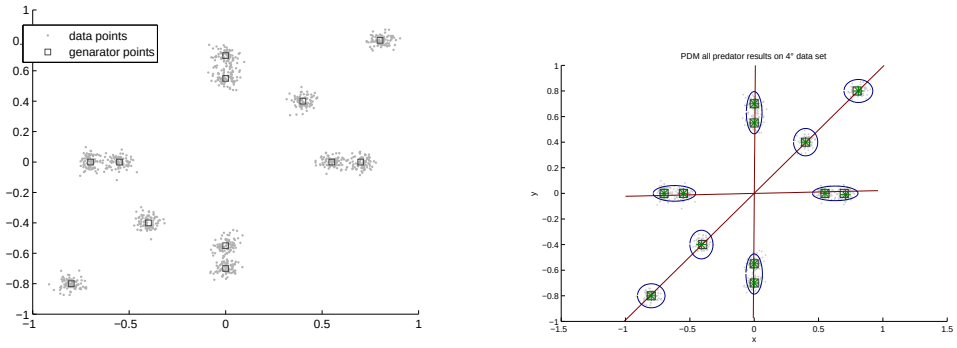
L'Ac_{PDM} indica l'oscillazione intrinseca al PDM nel determinare la classificazione, la quantità $1 - Ac_{PDM}$ può essere visto come raggio di confidenza di

Tabella 3.3: Risultati su terzo data set

Predator	$Rand(C_{PDM}, C_{ref})$	Number cluster	Ac_{PDM}
Point	0.3777	8	0.77906
Ellipse	0.4442	7	0.65859
Line	0.7209	6	0.55522

C_{PDM} questa dipende principalmente dalla possibilita di discriminare i dati rispetto all'informazione cercata col predatore e quindi un informazione aggiuntiva sull'insieme dei dati.
In generale più l'informazione discrimina i dati più l'Ac_{PDM} è vicino a 1 e più è alta la probabilità che questa informazione sia significativa, che poi possa coincidere o meno nel caso di dati simulati con l'informazione nascosta è un altro discorso.

Per chiarire questo punto si costruisce un esempio ad hoc, con la stessa procedura usata per creare il primo insieme di dati ma stavolta usando 12 punti generatori scelti opportunamente costruiamo l'insieme visualizzato in figura 3.1(a).



(a) Quarto data set

(b) Punti generatori e predatori

Figura 3.1: Insieme di dati, punti generatori (quadrati), punti (asterisco) rette ed ellissi predatori estratti dal PDM utilizzando le tre distinte classi

A questo insieme applichiamo il PDM con tutti e tre i predatori presi in esempio. In tabella 3.4 vediamo sia il confronto con C_{ref} che la Ac_{PDM}, l'impressione

vedendo questi risultati è che più l'informazione discrimina i dati (Ac_{PDM} vicino a 1) più il risultato sembra scadente ($Rand(C_{PDM}, C_{ref})$ lontano da 1. È una conclusione errata e quest'esempio ci permette di chiarire alcuni aspetti fondamentali.
La Ac_{PDM} indica solo la disciminabilità dei dati rispetto al predatore utilizzato, significa che nonostante i dati siano stati generati dai punti sia le rette che le ellissi li discriminano meglio. Analizzando meglio i dati ed i punti generatori (fig 3.1(a)) si nota che 8 dei 12 cluster sono accoppiati a formare 4 ellissi mentre a

Tabella 3.4: Result on third data

Predator	$Rand(C_{PDM}, C_{ref})$	Number cluster	σ_{PDM}
Point	0.9595	12	0.87672
Ellipse	0.7178	8	0.92815
Line	0.3053	3	1

gruppi di 4 i 12 cluster sono allineati. In tabella 3.5 ed in figura 3.1(b) possiamo visualizzare oltre ai punti generatori anche i predatori determinati dal PDM con le tre distinte classi utilizzate .

Si capisce ora il motivo dei valori del Ac_{PDM} difatti i 3 gruppi di 4 cluster allineati individuano 3 rette molto separate, non c'è ambiguità sul numero di rette individuabili ed anche spostando di poco le stesse la classificazione non cambia. Con i punti invece ad un piccola variazione della soluzione la classificazione cambia.

3.3 Data Clustering su Microarray simulati

3.3.1 Introduzione

I microarrays sono una delle più recenti tecniche che sfruttano le caratteristiche peculiari della doppia elica del DNA, ovvero la natura complementare delle due catene e la specificità dell'accoppiamento delle basi.
Infatti, da circa 30 anni, le tecniche standard di laboratorio per il rilevamento di specifiche sequenze nucleotidiche utilizzano una sonda (probe) di DNA, costituita

Tabella 3.5: Predator Results on fourth data

Generator Points		Predator points		Predator Ellipses				Predator lines			
x	y	p_1	p_2	c_1	c_2	a_1	a_2	α_1	β_1	α_2	β_2
0.7	0	0.71	-0.01	0.63	0	1.77	0.57	4.03	4.46	0.09	0.10
0.55	0	0.55	0								
-0.55	0	-0.55	0	-0.62	0	1.65	0.61				
-0.7	0	-0.7	0								
0	0.7	0	0.7	0	0.63	0.61	1.65	-0.01	-0.06	-1.35	-7.31
0	0.55	0	0.55								
0	-0.55	0	-0.55	0	-0.63	0.64	1.57				
0	-0.7	0	-0.7								
0.8	0.8	0.81	0.8	0.81	0.8	1.12	0.9	0.03	-1.96	0.03	-1.94
0.4	0.4	0.4	0.4	0.4	0.4	0.94	1.07				
-0.4	-0.4	-0.41	-0.4	-0.41	-0.4	0.9	1.12				
-0.8	-0.8	-0.8	-0.8	-0.8	-0.8	1.07	0.94				

da un piccolo frammento di acido nucleico marcato con un isotopo radioattivo o una sostanza fluorescente. La sonda, rappresentante la sequenza complementare a quella del gene da individuare, viene posta in contatto con un supporto solido (ad esempio, un gel od un filtro poroso) sulla cui superficie sono ancorati acidi nucleici provenienti da un dato genoma. Grazie alla peculiarità degli acidi nucleici di riconoscere le sequenze ad essi complementari, la sonda può legarsi in maniera selettiva al frammento ancorato ad essa complementare così che, semplicemente misurando la presenza e la quantità di marcatore legato al supporto solido, è possibile quantificare se e quanto è stato espresso un determinato gene .

Questa tecnica applicata per la prima volta da Ed Southern nel 1975 [71], ha aperto di fatto la strada alla possibilità di analizzare i profili di espressione genica di un intero organismo. Tuttavia, l'applicazione su larga scala di questa metodologia si è avuta solo di recente grazie all'utilizzo di supporti solidi non porosi, come il vetro, e alla messa a punto di tecniche foto-litografiche per la sintesi di frammenti oligonucleotidici ad alta densità spaziale. In particolare, i protocolli sviluppati

dal gruppo di Pat Brown a Stanford, hanno permesso di ancorare automaticamente migliaia di catene di cDNA su vetrini da microscopio e, grazie alla loro ibridazione con campioni di mRNA marcati selettivamente con molecole fluorescenti, di studiare il profilo di espressione di colture cellulari in stati fisiologici diversi [72]. Parallelamente, sono state messe a punto tecniche di mascheramento fotolitografico, normalmente utilizzate nell'industria dei semiconduttori, per la produzione di microarray capaci di 400.000 sonde oligonucleotidiche su una superficie di un pollice quadrato [73].

La tecnologia dei Microarray permette quindì di osservare simultaneamente le espressioni geniche di migliaia di geni in differente condizioni [72].
I Geni con espressioni "simili" (co-espressi) possono essere implicati negli stessi processi biologici. L'identificazione di questi gruppi di geni è usualmente fatta attraverso tecniche di data clustering. Queste dividono generalmente l'insieme totale dei geni in K sottoinsiemi (cluster) raggruppando i geni "vicini" tra loro rispetto ad una distanza o un criterio di dissimilarità (generalmente basato o sulla distanza euclidea o sulla correlazione tra le espressione geniche).

Diversi algoritmi di Data Clustering sono stati proposti ed applicati in quest'ambito. Algoritmi di tipo gerarchico (§1.2.1), K-means, PAM, K-medoids (§1.2.2) self-organizing maps (§1.2.5) sono tradizionalmente gli algoritmi più utilizzati nelle analisi dei Microarray.

In [64] sono stati messi a confronto valutandoli su alcuni insieme di dati simulati.

3.3.2 Metodologie confronto algoritmi

In [64] si sono inizialmente generati 100 insieme di dati campioni che simulano l'espressioni geniche dei microarrays. Ognuno di questi insiemi è stato creato partendo da 15 array che simulano espressione geniche casuali, in pratica per ognuno di questi array generatori, attraverso un numero casuale di copie ripetute, si sono generati dei gruppi di dati che uniti assieme formano l'insieme dei dati di partenza. Questi gruppi sono i cluster di riferimento, rappresentano dei geni

perfettamente coespressi.

Partendo da questi insiemi, Thalamuthu e il suo gruppo [64] ne hanno creati altri con le seguenti procedure:

Tipo I Aggiungendo all'insieme dei dati casuali che simulano geni sparsi (*scattered genes*)

Tipo II Aggiungendo ai dati un errore casuale che simula un rumore bianco.

Tipo III Eseguendo contemporaneamente le procedure precedenti

Il PDM non prevede l'identificazione dei geni sparsi, anche se è uno dei prossimi sviluppi, per questo motivo si è interessati ai risultati degli altri algoritmi sugli insieme di dati a cui viene aggiunto un rumore bianco. Questi ne sono 600, infatti, per ognuno dei 100 insiemi di partenza ne sono stati generati altri 6 aggiungendo via via un rumore crescente.

Le classificazioni risultanti dagli algoritmi testati vengono confrontate rispetto alle classificazioni di riferimento attraverso l'*adjusted rand index* 1.31

3.3.3 Algoritmi di Clustering utilizzati

I dati simulati vengono preventivamente preprocessati in modo tale da avere media 0 e varianza 1. Su questi dati in [64] vengono testati diversi algoritmi appartenenti a varie famiglie:

Gerarchico: è stato usato un metodo aggregativo e come criterio di terminazione si è fissato il numero di cluster (conoscenza in questo caso nota), mentre sono state usate sia il collegamento singolo il medio ed il completo, i risultati migliori che sono quelli che si sono ottenuti con il collegamento completo, la metrica di base è la euclidea.

K-means: implementazione classica dell'algoritmo spiegato in §1.2.2, il numero di cluster viene fissato a 15 (conoscenza in questo caso nota a priori) mentre vengono ripetute le applicazioni con diverse condizioni iniziali scelte

casualmente e tra queste alla fine viene scelta quella classificazione con la minima distanza intracluster.

PAM: classico algoritmo di k-medoids , è utile quando è difficile determinare matematicamente il centro dei cluster ma in questo caso è praticamente equivalente al k-means.

SOM: Self-organizing-maps, sono state applicate in parecchie analisi dei microarray. Sostanzialmente le SOM possono essere viste come un criterio k-means ristretto su una griglia bidimensionale. Rispetto al k-means ottimizzano in uno spazio ridotto è questo può ridurre l'efficienza sopratutto quando la differenza tra le dimensioni tra lo spazio di origine e la griglia dei nodi è alta. Similmente poi al K-means le SOM sono molto sensibili alla scelta del numero dei nodi.

M-clust: questo algoritmo si basa su un modello probabilistico preciso, cerca la migliore mistura di gaussiane che minimizza la probabilità della distribuzione dei dati nell'insieme.

Dato $\theta_j = (\mu_j, \Sigma_j)$ essere il parametro associato ad una distribuzione di probabilità $f_j(x;\theta_j)$, μ_j il centro e Σ_j la covarianza del cluster j. Chiamata p_j la probabilità che un osservazione appartenga al j_esimo cluster ($p_j \geq 0; \sum_{j=1}^{k} p_j = 1$), la probabilità di classificazione di n osservazioni indipendenti $x_1, x_2, ..., x_n$ è data da:

$$L(\theta, p|x) = \prod_{i=1}^{n} [\sum_{j=1}^{k} p_j f_j(x_i;\theta_j)] \tag{3.4}$$

dove $X = (x_1, x_2, ...x_n)$ è l'insieme dei dati e $(\theta_1, \theta_2, ..., \theta_k)$ sono i parametri delle 15 gaussiane a cui si associano i 15 cluster.

Anche in questo caso l'algoritmo è molto sensibile al parametro del numero di cluster.

Tight-clustering: è una metodologia per ottenere, dato un algoritmo di data clustering, risultati più robusti e meno sensibili ai parametri iniziali. General-

mente viene utilizzato con il k-means, sostanzialmente genera molte classificazioni sia con differenti parametri iniziali che determinando i migliori centroidi utilizzando solo una parte casuale consistente dei dati di partenza. La metodologia permette di "fondere" una serie di classificazioni in una sola, in questo modo si riesce a valutare anche, ad esempio, il numero di cluster. Tra tutti gli algoritmi è quello meno sensibile quindi al numero di cluster.

3.3.4 Risultati ottenuti

In figura 3.2(a) Thalamuthu et al. graficano, per ognuno dei 6 algoritmi sopra descritti, i risultati medi ottenuti sui 100 insiemi di dati generati al variare del rumore aggiunto. In pratica chiamando con:

D_i con $i = 1..100$ il generico insieme di dati generato,

C_i la classificazione ideale di D_i,

D_i^σ il data base generato aggiungendo un rumore bianco quantificato da σ a D_i,

$C_i^\sigma(Algoritmo)$ la classificazione risultante dal generico algoritmo su D_i^σ

$AdgRandIndexMean^\sigma(Algoritmo) = \frac{\sum_{i=1}^{100} AdgRandIndex(C_i^\sigma(Algoritmo), C_i)}{100}$

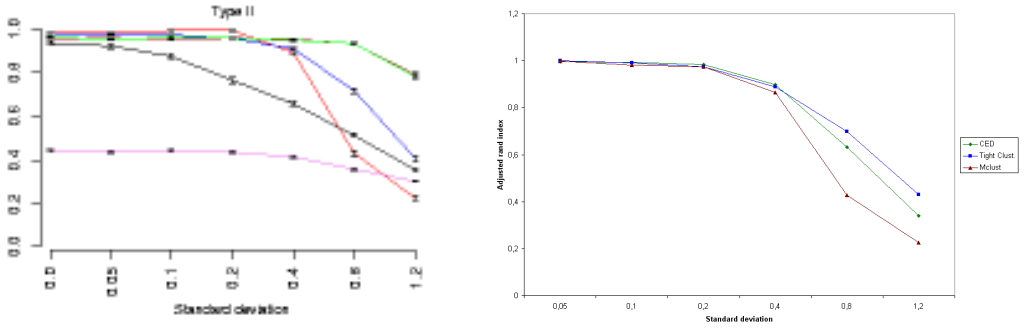
in figura 3.2(a) si visualizza i valori dell'$AdgRandIndexMean^\sigma(Algoritmo)$ al variare di σ (ascissa) e algoritmo (colore).

Dal grafico si nota come Il Pam ed il k-means traggono molti benefici dalla conoscenza del numero di cluster, ottimo risultato anche del tight-clustering, e buono l'm-clust ed il gerarchico, le som non hanno buone performance del resto vengono più spesso utilizzate come algoritmo di data visualizzazione.

3.4 PDM applicato ad i dati simulati

3.4.1 Classe predatore adottata

La classe predatore usata ha le seguenti caratteristiche:



(a) Confronto senza PDM (b) Confronto con PDM

Figura 3.2: adjusted rand index (asse y) per SOM (viola), gerarchico (nero), K-means(marrone), Pam (verde), Mclust (rosso), Thight-clustering(blu) e PDM (verde), risultati su dati simulati

- Il generico predatore è codificato da un punto.
- L'operatore iniziale è un punto casuale.
- L'operatore di mutazione a partire da un predatore ne genera un altro addizionando un rumore gaussiano ad alcune componenti scelte casualmente.
- La funzione di hunter-fitness tra predatore e dato usata è proporzionata al coefficiente di correlazione di Pearson centrato tra le coordinate del predatore e quelle del dato:

$$hf(p,x) = 1 - \sigma_{p,x} \hookrightarrow \sigma_{p,x} = \frac{\sum_{i=1}^{N}(p_i - mean(x))\sum_{i=1}^{N}(x_i - mean(x))}{\sqrt{\sum_{i=1}^{N}(p_i - mean(p))^2 \sum_{i=1}^{N}(x_i - mean(x))^2}} \tag{3.5}$$

dove $mean(p) = \frac{\sum_{i=1}^{N} p_i}{N}$ e $mean(x) = \frac{\sum_{i=1}^{N} x_i}{N}$

3.4.2 Risultati PDM

Il PDM impiega su un semplice personal computer circa un ora con i parametri adottati per restituire i risultati, questo tempo anche se accettabile crea qualche problema quando si vogliono investigare i risultati su un ampia campionatura, per

questo motivo si è preferito ridurre la campionatura con i dati generati a soli 10 dei 100 disponibili, la metodologia della scelta della campionatura è casuale ed indipendente dai possibili risultati ottenuti.
La media è stata fatta quindi su un campione di 10 insieme di dati invece che 100. Per valutare i risultati si è utilizzata la stessa procedura, in figura 3.2(b) possiamo vedere i risultati ottenuti confrontandoli con quelli ottenuti in [64] con i due algoritmi che hanno andamenti più similari.

I risultati del PDM si avvicinano molto a quelli ottenuti dal tight-clustering , c'è da dire che il PDM cosi come il tight-clustering non sono sensibili al numero di cluster, quest'aspetto, che è generalmente un vantaggio, nel particolare contesto sperimentale è uno svantaggio. La scoperta del numero di cluster è un operazione non esente da errori, soprattutto su complesse strutture di insiemi di dati. Il tight-clustering ed il PDM usano principi che si basano sul fusion clustering per determinare il numero di Cluster corretti, producendo un certo numero di classificazioni. I risultati analizzati sono stati ottenuti dal tight-clustering utilizzando proprio il k-means per produrre le varie classificazioni, la differenza quindì con i risultati ottenuti dal k-means è da attribuire alla componente di errore dovuta alla variabile incognita aggiuntiva del numero di cluster, presente nel secondo e non nel primo.

3.4.3 Verifica relazione tra l'errore commesso e Ac_{PDM}.

Il PDM oltre alla classificazione da un informazione aggiuntiva, la misura media di quanto la classificazione è stabile. L'idea, basata su considerazioni intuitive, che questa informazione possa misurare l'errore commesso nel classificare trova una conferma sperimentale significativa.

In figura 3.3 vengono graficati al variare del rumore i risultati medi ottenuti (linea blu) i risultati medi "attesi" (linea rossa) e soprattutto lo scarto quadratico medio tra i due risultati (linea verde). In altri termini per ogni D_i^σ il PDM ha due risultati: 1) la classificazione che chiameremo C_i^σ, 2) l'oscillazione media di questa classificazione che chiameremo Ac_i^σ. La linea rossa indica i valori del $AdgRandIndexMean^\sigma(PDM)$ coefficiente che stiamo ipotizzando di poter utiliz-

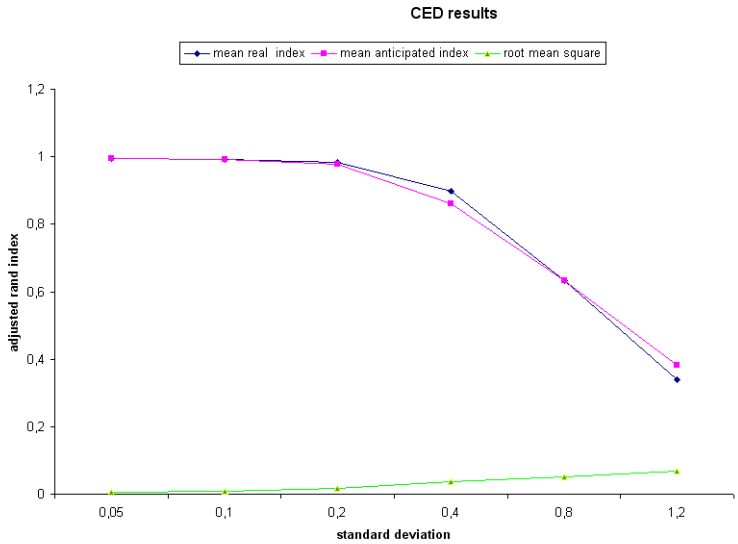


Figura 3.3: Errore commesso e previsto del PDM

zare come previsore di errore, la linea blu indica l'errore effettivamente commesso $\frac{\sum_{i=1}^{10} AdgRandIndex(C_i^{\sigma}, C_i)}{10}$ ed infine la linea verde indica $\sqrt{\frac{\sum_{i=1}^{10} (AdgRandIndex(C_i^{\sigma}, C_i) - Ac_i^{\sigma})^2}{10}}$ lo scarto quadratico medio tra l'"errore" atteso e quello ottenuto.

Si nota una significativa correlazione tra il risultato reale e la stabilità della classificazione misurata dal PDM, la differenza tra i due risultati è tanto più bassa quanto più il risultato atteso è buono (vicino a 1). La possibilità di avere una stima della bontà del risultato ottenuto è nei casi reali un informazione estremamente utile.

Capitolo 4

Applicazione sui dati reali

4.1 Introduzione

Il PDM è stato testato sul dataset di microarray del ciclo cellulare del lievito Saccharomyces cerevisiae reso pubblico e disponibile da Spellman et al. [74]. Ogni singolo dato di questo insieme è ricavato da 4 esperimenti sincronizzati, tralasciando i "missing value" comunque consistenti, si hanno, per i 4 esperimenti, 18, 24, 17 e 14 osservazioni che possono essere visti come 4 vettori o serie temporali di reali che messi assieme formano un vettore di 77 componenti.
I dati se non sono affetti da missing value sono punti di R^{77} altrimenti sono rette piani o iperpiani di questo iperspazio a secondo se manca uno, due o più osservazioni.
Per una descrizione dettagliata di questo insieme di dati si può fare riferimento al lavoro originale di Spellman [74].

In 4.2 si descriverà un metodo alternativo di analisi [4] con i relativi risultati sugli stessi dati, in 4.3 si illustrerà l'impostazione ed i risultati ottenuti dal PDM ed infine si confronteranno i risulti evidenziando quelli che sono i pregi ed i difetti del PDM.

4.1.1 Tecnica di validazione

In questo capitolo si computa il P-value biologico dei cluster ottenuti utilizzando il GOTerm Finder, una web application messa a disposizione dal database S.cerevisiae ([75]).

Questo valore indica la probabilità di mettere assieme i geni appartenenti ad una certa categoria in maniera casuale, le categorie in questione sono determinate da alcune conoscenze biologiche che, sopratutto nel gene del lievito, sono ampie.

4.2 Classificazione dei dati combinando diverse tecniche di analisi.

In [4] si propone un approccio, basato su un solido formalismo matematico, atto a visualizzare e classificare espressione geniche rumorosi con presenza di *missing value*. Il metodo può essere diviso in due parti:

***Preprocessing*:** l'insieme dei dati viene preventivamente processato, tale operazione, come spesso succede, è stata effettuata sfruttando alcune caratteristiche del particolare insieme di dati.

***Clustering* e *visualizzazione*:** una più generale operazione di clustering che permette tra le altre cose un utile visualizzazione dei risultati.

4.2.1 Preprocessing

I microarray sono dati molto rumorosi, e conseguentemente il *preprocessing* ha un ruolo importante nella buona riuscita dell'analisi. In questo caso il preprocessing è utile per scartare i dati più rumorosi e eliminare i *missing value*. Sono state effettuate due operazioni distinte, la prima consiste nell'eliminare quei geni che vengono considerati troppo rumorosi, nella seconda si determinano le componenti principali dai dati che rappresenteranno un nuovo insieme di dati che verrà

analizzato.

Scarto dei geni rumorosi

La struttura del rumore a cui sono affetti i dati delle espressione geniche è stata studiata in una serie di articoli ([78]; [79]) dove è stato provato essere abbastanza complicata. Per il caso del dataset analizzato è impossibile effettuare un accurata analisi preventiva del rumore. Tuttavia in questo insieme ci sono solo sequenze temporali per ogni gene. Il numero di valori in ogni sequenza è tra 14 e 24. Questo permette le applicazioni di usuali metodi statistici per stimare il rumore ([80]; [82]), ad esempio il modello ANOVA e il metodo *bootstrap* ([83]; [79]), metodi bayesiani per stimare il livello di trascrizione ([84] ; [81]), metodi basati su ripetizione multiple del segnale ([86]) o avanzati metodi di normalizzazione ([85]). In questo caso si usa un semplice metodo per stimare il rumore che si basa sulla periodicità dell'espressione genica,

L'insieme di dati sono osservazioni prese ad intervalli temporali della relativa abbondanza di MRna di ogni gene nel test, comparata con la sua abbondanza in un campione di riferimento (precisamente il $\log_2$ di questo rapporto che per semplicità chiameremo espressione genica). Questi segnali sono periodici (il periodo viene determinato dal tempo necessario tra una divisione del gene del lievito e la successiva divisione), il periodo dipende da molti parametri la maggior parte dei quali collegati all'ambiente sperimentale. Secondo Spellman et al. [74] in condizioni normali, anche se ogni esperimento può avere un suo periodo specifico compreso tra i 79 ed i 101 minuti, il principale è 90 minuti. La differenza tra le espressione geniche in due periodi può essere considerata come una stima del rumore.

Il primo passo è stimare il periodo per ogni esperimento. Per ogni esperimento, fissato un periodo T si misura il coefficiente di correlazione tra i valori compresi tra 0 e $T/2$, con quelli compresi tra T e $3/2T$ e si sommano, si sceglie quel T che massimizza questa somma. Una volta determinato il periodo è possibile avere una stima del rumore misurando le differenza $f(t+T)-f(t)$ e sopratutto determinare

quelli che sono i geni più rumorosi. In questa maniera in [4] eliminano circa la meta dei geni passando da 6125 a 2679

4.2.2 Normalizzazione e determinazione delle componenti principali

Passo successivo è quello di eliminare i missing value, questa operazione viene effettuata attraverso un estrazione delle componenti principali dei microarray tramite una PCA non-lineare.

Come primo passo le espressione geniche dei dati restanti vengono normalizzate. La media e la deviazione standard delle righe dei segnali di microarray vengono calcolate, il segnale viene prima sottratto dalla media e poi diviso per la deviazione standard, in modo da ottenere processi a media nulla ed a varianza unitaria. In questo modo si hanno, per ogni processo 4 segnali normalizzati (uno per ogni esperimento) con media nulla e deviazione standard unitaria, che verranno raggruppati in un unico vettore.

Per geni che non hanno "missing value", il vettore risultante ha 73 dimensioni (18, 24, 17 e 14 osservazioni rispettivamente per gli esperimenti alpha, CDC15, CDC28 e elutriation).

Uno dei principali problemi dei microarray è la presenza dei "missing value". Il primo passo per risolvere questo problema è stato eliminare quei geni che ne presentono troppi per essere significativamente analizzati, da questo ulteriore filtraggio si passa da 2679 a 2445 geni.

Il secondo passo è l'estrazione delle componenti principali (autovettori) della matrice di autocorrelazione dei geni restanti, le componenti principali sintetizzeranno le informazioni presente nelle espressione geniche ed allo stesso tempo non saranno soggette ai *missing value*.

Questa processo è basato sul metodo delle PCA non lineari che possono stimare gli autovettori da dati campionati non uniformemente. Quest' approccio si basa su STIMA un algoritmo descritto in [76] e [77].

Le PCA sono reti neurali atte alla estrazioni delle componenti principali è una tecnica largamente usata in data analisi ([91], [76]; [77]; [87], [88]). queste possono essere realizzate in vario modo ([76]; [77]; [89], [90]; [87]). In questo caso si è utilizzata una rete neurale PCA non lineare.
Dopo aver applicato questa metodologia si ottengono per ognuno dei 2445 geni un vettore con 32 componenti (8 componenti per ogni esperimento) che costituiranno l'insieme dei dati analizzato con il modello delle PPS.

4.2.3 PPS Descrizione dell'algoritmo

Le PPS ([92]; [93]) sono un estensione non lineare delle componenti principali, nel quale ogni nodo sulle PPS è la media di tutti i punti (elementi dell'insieme dei dati) che sono proiettari vicino ad esso. Dal punto di vista teorico, le PPS sono una generalizzazione del Generativi Topografiche Mappe (GTM) ([24]), che possono essere viste come una parametrica alternativa alle SOM(§1.2.5). Tra i vantaggi delle PPS vi è la formulazione parametrica e flessibile per ogni geometria/topologia in ogni dimensione, la convergenza garantita (In verità l'addestramento delle PPS è compiuto attraverso l'algoritmo Expectation-Maximization). Le PPS sono caratterizzate dalla topologia latente, grazie alla intrinseca flessibilità si possono creare una varietà di topologie differenti per le PPS, tra queste anche la sfere tridimensionale.

Una sfera è finita, simmetrica e senza bordi per questo motivo le PPS sferiche sono in grado di modellare i dati di input di dimensione molto elevata, in quanto catturano il comportamento dei dati che si localizzano nelle aree di confine dello spazio multi-dimensionale, una situazione questa conseguenza della *curse of dimensionality* (maledizione delle alte dimensioni) che si manifesta in spazi di dimensione elevata. Inoltre la topologia della sfera può essere facilmente compresa dall'uomo e per questo può essere usata per permettere la visualizzazione di dati ad alta dimensionalità.

Le PPS definiscono una mappa non lineare $y(x;W)$ da uno spazio latente Q-dimensionale $(x \in R^Q)$ allo spazio dei dati D-dimensionale $(t \in R^D)$, dove generalmente $Q < D$. La funzione (continua e derivabile) $y(x;W)$ associa ognuno

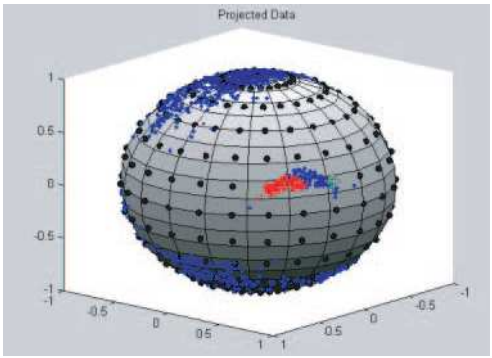


Figura 4.1: Sfera PPS: i punti neri rappresentano le variabili latenti mentre i punti colorati sono le proiezione dei dati sulla sfera

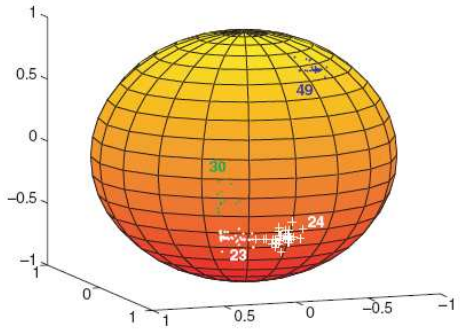


Figura 4.2: Proiezione dei geni appartenenti ad alcuni rilevanti cluster scoperti in [4]

degli M punti (variabili latenti) nello spazio latente (dove M è il numero delle variabili latenti che sono fissati a priori) al corrispondente punto nello spazio dei dati. Al cambiare dei parametri W si ottengono modi differenti di proiettare le variabili latenti nello spazio dei dati, le PPS considerano questi punti come centri di distribuzione gaussiane e cercano i parametri w che minimizzano la probabilità dei dati di appartenere ad esse. I dati poi possono essere classificati e proiettati sulla sfera rispetto alla variabile latente che nello spazio D-dimensionale risulta più vicina.

L'aspetto interessante delle PPS che permettono di visualizzare i dati sulla sfera tridimensionale. Di contro l'algoritmo di clustering è estremamente sensibile al numero di variabili latenti.

4.2.4 (NEC) Negentropy Clustering

Le PPS sono molto sensibile al numero di variabili latenti che diventano anche il numero di cluster, per ovviare a questo problema le PPS vegono accoppiate con un algoritmo gerarchico. L'idea è utilizzare un numero di variabili latenti molto alto ed i corrispondenti cluster metterli assieme attraverso un algoritmo gerarchico di tipo agglomerativo.

In particolare l'algoritmo utilizzato in questo lavoro si basa sull'informazione della Negentropia (Negentropy o NEC).

La Negentropy ([91]) è un importante misura di Nongaussianità ed è la misura utilizzata per fondere i cluster, i cluster più simili cioé quelli che fondendosi risultano più "gaussiani" vengono raggruppati finchè non si raggiunge un valore di soglia.

La sola informazione necessaria è la definizione del valore di soglia. In questo lavoro si clusterizza utilizzando diversi valori di soglia all'interno di un certo intervallo, la scelta viene fatta osservando la distribuzione dei cluster ottenuti sulla sfera.

4.2.5 Risultati

Le PCA non lineari sono state usate ad estrarre il primo autovettore della matrice di autocorrelazione dei 2445 geni che sono passati dalla procedura di filtro, in seguito sono state applicate le PPS con 98 variabili latenti ed il NEC con un valore di soglia di 0.6. Tutti questi parametri sono stati fissati empiricamente. Il NEC inizia con 98 cluster (il numero delle variabili latenti utilizzato dalle PPS) e ottiene come uscita 56 clusters.

In Figure 4.2 si mostrano le proiezioni sulla sfera dei geni appartenenti ad alcuni dei cluster scoperti. Per valutare il significato biologico dei cluster ottenuto si è computato il P-value usando lo strumento: GOTerm Finder dal database S.cerevisiae (§4.1.1). La affidabilità biologica della metodologia adottata è confermata dal fatto che diversi cluster (22 su 56) mostrano un P-value significativo $< 10^{-2}$. In particolare viene messo in evidenza la scoperta di un cluster con un P-value molto basso dell'ordine di 10^{-23}.

4.3 Approccio PDM

Il PDM, come accennato, è stato applicato ai microarray del gene del lievito. Uno degli aspetti caratteristici del PDM è che indipendentemente dal preda-

tore utilizzato si lavora localmente rispetto al dato, quest'aspetto può in maniera estremamente naturale e semplice risolvere il problema dei missing value 4.3.1.

Il primo passo è stato testare il metodo su alcuni dei geni classificati da Spelmann e filtrati dai geni rumosori usando la precedura spiegata in 4.2.1.

In seguito poi si applicherà il PDM su l'intero insieme di dati, lo si farà utilizzando come predatori sia i punti ma anche le rette e le ellissi, verranno poi valutati i P-value dei cluster ottenuti.

4.3.1 il PDM applicato sui missing value

Le espressione geniche dei microarray contengono spesso dei missing value, la maggior parte degli algoritmi di data clustering (tra cui tutti quelli confrontati col CED) necessitano di una matrice completa di valori, sono necessari quindi algoritmi per stimare i dati mancanti ([94], [95]) o estrarre stimando dai dati lo stesso numero di componenti principali (PCA, Non-Linear PCA, STIMA [76] e [77])
Al di là della bontà o meno della metodologia usata i missing value restano un problema serio e aperto nell'analisi dei microarray.
L'approccio usato nel CED permette di scavalcare questo problema lavorando direttamente su una matrice non completa di valori. Alla base del funzionamento del CED non c'è una metrica ma una funzione che permette, per ogni dato, di determinare il centro di aggregazione (predatore) a lei più vicino, la funzione può essere quindi definita localmente al dato. Prendiamo come esempio la classe di predatori utilizzati negli esperimenti sui dati simulati l'adattamento naturale ad un problema con i missing value consiste semplicemente nel sostituire il valore $\sigma_{p,x}$ nell'equazione 3.5 con il seguente valore:

$$\sigma_{p,x}^{mod} = \sigma_{p,x} = \frac{\sum_{i=1}^{N}(p_i - mean(x))\sum_{i=1}^{N} a_i}{\sqrt{\sum_{i=1}^{N}(p_i - mean(p))^2 \sum_{i=1}^{N} a_i^2}} \tag{4.1}$$

dove $a_i = (x_i - mean(x))$ se esiste x_i altrimenti $a_i = 0$ mentre $mean(x)$ è la media calcolata sulle componenti esistenti di x.
Analogamente una simile modifica può essere apportata anche volendo usare la

metrica euclidea.

Prendiamo ad esempio due punti con missing value nello spazio : $(1,?,3)$ e $(?,4,3.2)$ i centroidi più vicini al primo dato sono tutti quelli appartenenti alla retta: $r_1 \equiv \{x = 1, z = 3\}$ mentre quelli più vicini al secondo dato sono quelli appartenenti alla retta: $r_2 \equiv \{y = 4, z = 3.2\}$ il centroidi che meglio minimizza la distanza euclidea per entrambi i punti è: $(1,4,3.1)$.
Supponiamo invece che attraverso una semplice regressione lineare si stimino i due missing value si ottengono i seguenti valori: $(1,2,3)$ e $(4.8,4,3.2)$, conseguentemente come centroidi che minimizza la distanza euclidea ottengo: $(2.9,3,3.1)$.
I due risultati sono molto diversi, l'ultimo è influenzato fortemente dall'ipotesi che i dati regrediscono linearmente, se l'ipotesi è corretta l'ultimo risultato è preferibile ma se non c'è conoscenza a priori sui dati è preferibile il primo risultato che non fa nessuna ipotesi a priori .

4.3.2 Il PDM sul sottoinsieme classificato da Spellman

Si applica il PDM alla intersezione del sottoinsieme dei geni classificati da Spellman[1]([3]) e filtrati in 4.2.1. La classificazione di Spelmann viene usata come riferimento. Il PDM viene applicato sia ai dati originali sia alle componenti principali estratte come spiegato in 4.2.2, su quest'ultimo insieme di dati è stato applicato anche il k-means[2], chiameremo le tre classificazioni risultanti rispettivamente C_{PDM}, C^{Pr}_{PDM},C^{Pr}_{Kmeans} mentre quella di Spellman $C_{Spellman}$. Il predatore utilizzato è rappresentato da un generico punto in R^{77}, concettualmente rispetto al predatore introdotto in 3.4.1 differisce solo per la hunter-fitness modificata come spiegato nel paragrafo precedente per adattarsi ai *missing value*.

[1]Bisogna annotare che si sono usati i geni appartenenti solo ad 8 delle 9 classi determinate da Spellman dato che non si è riuscito ad avere a disposizione informazioni sui geni appartenenti al nono cluster.

[2]È stato utilizzata la funzione presente in Matlab 7 con procedura standard di inizializzazione, il numero di cluster fissato a 8, metrica euclidea e 1000 iterazioni

I risultati ottenuti sono interessanti. Si può notare dalla matrice di confusione 4.2

$$MDC(C_{PDM}, C_{Spellman}) \begin{pmatrix} 29 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 39 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 27 \\ 0 & 0 & 0 & 0 & 0 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 8 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.2}$$

come i risultati del PDM sui dati originali coincidano molto bene con la classificazione di Spellman. Sui dati preprocessati i risultati sono peggiori ma comunque simili a quelli ottenuti dal kmeans

$$MDC(C^{Pr}_{PDM}, C_{Spellman}) \equiv \begin{pmatrix} 0 & 40 & 0 & 0 & 2 & 0 & 0 & 4 \\ 31 & 0 & 0 & 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 11 & 0 \\ 0 & 0 & 0 & 4 & 10 & 0 & 3 & 0 \\ 0 & 0 & 8 & 0 & 3 & 14 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 7 & 1 \\ 0 & 0 & 0 & 3 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 22 \end{pmatrix} \tag{4.3}$$

$$MDC(C^{Pr}_{Kmeans}, C_{Spellman}) \equiv \begin{pmatrix} 31 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 14 & 0 & 2 & 0 \\ 0 & 5 & 0 & 0 & 1 & 0 & 0 & 27 \\ 0 & 0 & 0 & 2 & 14 & 1 & 0 & 0 \\ 0 & 35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 14 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \end{pmatrix} \tag{4.4}$$

I Cluster ottenuti dal PDM sull'insieme di dati originali coincidono molto bene con la classificazione di Spellman (solo 5 geni su 186 sono stati misclassificati) , l'adjusted rand index è uguale a 0.936 molto vicino a 1.

Un interessante aspetto è il confronto fra i risultati prima e dopo la procedura di pulitura dei missing value, cioé la differenza tra i risultati ottenuti sui dati originali e sulle componenti principali. In tabella 4.3.2 possiamo visualizzare l'adjusted rand index per le tre classificazioni ottenute.

Dati	K-means	PDM
Preprocessati	0.743	0.663
Originali		0.936

Sulle componenti principali il k-means ha migliori performance del PDM $0.743 > 0.663$ a conferma di quello osservato anche in 3.4.2, ma non avendo il PDM necessità di eliminare i missing value può essere applicato direttamente sui dati originali senza perdere quelle informazioni che, comunque, nell'operazione di estrazione delle componenti principali vengono perse, in quest'ottica il confronto è nettamente a vantaggio del PDM $0.743 < 0.936$.

In definitiva la possibilità di lavorare direttamente sui dati originali sembra dare un vantaggio consistente.

4.3.3 Risultati PDM sulle espressioni geniche dei geni del lievito

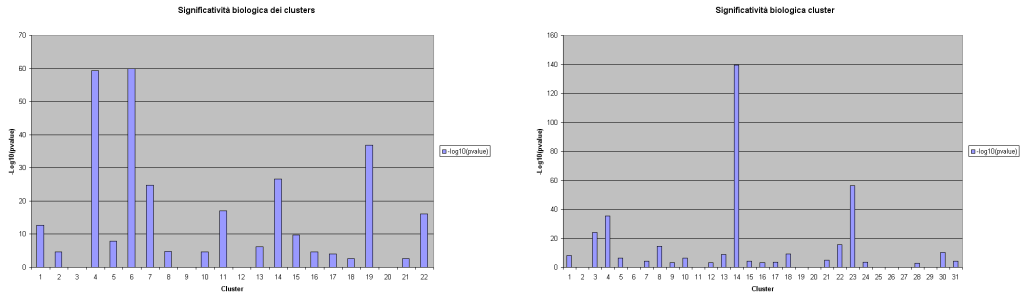
In questa sezione si applica il PDM all'insieme di dati completo non preprocessato. Si sono usati le tre differenti classi di predatore: punti, rette ed ellissoidi con aria costante. Si è valutato in prima analisi l' Ac_{PDM} ottenuto con le tre predatori e poi il p-value dei cluster ottenuti.

Utilizzando come predatori i punti il PDM determina automaticamente 22 clusters ed un Ac_{PDM} pari a 0.41034, utilizzando invece le rette si determinano 32 clusters con Ac_{PDM} pari a 0.33295 infine utilizzando come predatori ellissoidi si sono ottenuti 11 cluster ma un Ac_{PDM} praticamente pari a 0. Questi valori

danno delle informazioni importanti: per il PDM, tra i tre predatori, il miglior discriminante è il punto seguito dalle rette e poi dagli ellissoidi.

Gli ellissoidi che sono una generalizzazione del predatore punto, hanno risultati peggiori perché hanno probabilmente il numero di parametri liberi quasi doppio rispetto alle rette ed ai punti. Un Ac_{PDM} pari a 0 significa che le classificazioni ottenuta in questo caso è casuale. Nel caso dei punti invece l'Ac_{PDM}, anche se basso, è consistente con i valori ottenuti sui data set che simulano un rumore molto alto, questo può essere vista come un ulteriore prova dell'altà rumorosità dei dati analizzati.

La stima dell'Ac_{PDM} viene confermata anche dalla significatività biologica dei cluster, i cluster ottenuti con le elissoidi hanno un p-value maggiore di 0.01 che è per convenzione la soglia massima per dire se un cluster è biologicamente significativo, di contro molti dei cluster ottenuti sia con le rette che con i punti sono significativi (Fig 4.3).



(a) Significatività clusters PDM con punti come predatori

(b) Significatività clusters PDM con punti come predatori

Figura 4.3: Nelle figure si indica il $-\log_{10}$ dei pvalue dei cluster ottenuti utilizzando come predatori i punti 4.3(a) e le rette 4.3(b)

Utilizzando come predatori punti il PDM ha determinato 22 cluster di cui 18 risultano avere una significatività biologica, due di questi hanno un p-value dell'ordine di 10^{-60} ben 5 su 23 cluster risultano avere un p-value $< 10^{-23}$ in figura 4.3(a) si possono visualizzare e quantificare le significatività biologiche dei cluster ottenuti.

Utilizzando come predatori rette la significatività dei cluster è ancora più interessante, (figura 4.3(b)) in percentuale il numero di cluster significativi determinati sono minori, 22 su 32, ma tra questi ce ne è uno che ha un p-value dell'ordine di 10^{-140} un altro dell'ordine dell'ordine di 10^{-57}, in un certo senso le espressione geniche corrispondenti a geni accomunati da alcuni processi biologici sembrano più essere correlati linearmente che vicini.

4.4 Vantaggi e svantaggi delle due differenti metodologie

In [4] si sono usati sia conoscenza specifiche sul particolare data base e si sono combinate avanzate tecniche di preprocessing e classificazione, rispetto al PDM ha il vantaggio di poter visualizzare i dati sulla sfera, ma i risultati e la signifiatività biologica dei cluster ottenuti è estremamente inferiore a quella ottenuta col PDM sia utilizzando rette sia i punti.

Un confronto oggettivo tra le due metodologie non è stato fatto, il PDM mostra di avvantaggiarsi notevolmente dalla possibilità di lavorare direttamente sui dati senza necessità di stimare i *missing value* o utilizzare complicate componenti principali estratte dai dati. La possibilità inoltre di poter usare differenti predatori apre la possibilità di investigare nuove metodologie di aggregazione che come dimostrato per le rette potrebbero avere interessanti implicazioni teoriche. I cluster comunque andrebbero valutati da un biologo come fatto in [4].

Capitolo 5

Nuovo approccio per confrontare classificazioni

5.1 Introduzione

In questa capitolo introdurremo un nuovo approccio al problema del confronto fra classificazioni non supervisionate, questo permette le seguenti cose:

1. Ricavare i coefficienti più usati per il confronto fra classificazioni non supervisionate partendo da quelli più usati per il confronto fra classificazioni supervisionate.

2. Semplificare l'interpretazione dei coefficienti per il confronto tra le classificazioni non supervisionate.

3. Determinare nuovi coefficienti particolarmente utili quando almeno una delle due classificazioni è vincolata da un parametro che viene dato in ingresso al classificatore non supervisionato.

In §5.2 si discuterà il nuovo approccio al problema basato sull'introduzione di particolari classificatori booleani associati ai generici classificatori non supervisionati e pardendo da essi si introdurranno nuovi coefficienti, in §5.3 si dimostrerà come molti dei coefficienti ricavati sono equivalenti a quelli usati in letteratura

per il confronto fra classificazioni non supervisionate, in §5.4 si interpreteranno i vari coefficienti ricavati con particolare attenzione ad uno che sembra nuovo, in §5.5.1 infatti si analizzeranno i possibili utilizzi di quest'ultimo che risulta essere estremamente adatto a confrontare classificazioni vincolate da un parametro fissato inizialmente, situazione che è comune a molte famiglie di algoritmi di data mining unsupervised.

5.2 Classificatore booleano di accoppiamento

La domanda giusta da porsi nelle classificazioni unsupervised è se il classificatore è capace di raggruppare i dati correttamente, piuttosto che di classificarli correttamente, dato che le classi non hanno nessun significato a priori.

Data una classificazione di riferimento, le decisioni corrette sono quelle in cui il classificatore raggruppa o meno due dati se questi sono presenti congiuntamente in uno stesso cluster della classificazione di riferimento.

La questione fondamentale riguarda la decisione di raggruppare assieme oppure separatamente i dati;

Per formalizzare l'aspetto decisionale di accoppiare o meno i dati si definisce un evento associato ad ogni classificazione A di un generico insieme D nella seguente maniera :
L'evento α^I associato alla classificazione A è quello che preso in maniera casuale con ripetizione 2 elementi di D questi appartengono ad uno stesso cluster della classificazione A. o alternativamente:
L'evento α^{II} associato alla classificazione A è quello che preso in maniera casuale una coppia distinta inclusa in D questa é inclusa in uno stesso cluster della classificazione A.

Gli eventi α^I ed α^{II} sono sottoinsiemi dello spazio $D \times D$, dove D è l'insieme dei dati appartenenti al data base.

Tale sottoinsieme corrisponde al dominio per il quale è vera la funzione booleana

definita nella seguente maniera:

$$f_{\alpha^I}(x,y) = \begin{array}{ll} vero & \exists A_i : (x,y) \in A_i \\ falso & altrimenti \end{array} \tag{5.1}$$

dove x e y sono dei generici dati appartenenti a D e A_i è un cluster della classificazione A.

Date due classificazioni $A = \{A_i\}_{i=1..C_a}$ e $B = \{B_j\}_{j=1..C_b}$, e due eventi α^I e β^I si ha che: $f_{\alpha^I} = f_{\beta^I} \Leftrightarrow A \equiv B$ a meno di una permutazione dei nomi dei cluster.

Si definisce classificatore di Accoppiamento con ripetizione associato alla classificazione A e si chiama A^I il classificatore booleano che classifica *true* tutti i dati nell'insieme $D \times D$ che appartengono ad uno stesso cluster della classificazione A e *false* quelli che appartengono a due cluster diversi.

Analogamente si definisce classificatore di Accoppiamento delle coppie distinte associato alla classificazione A e si chiama A^{II} il classificatore booleano che classifica *true* tutte le distinte coppie nell'insieme D incluse in uno stesso cluster della classificazione A e *false* le altre (quelle incluse nell'unione di due cluster).

Abbiamo che $A^I(x,y) = true \Leftrightarrow \exists A_i \in A : x \in A_i \wedge y \in A_i$ ed $A^{II}(x,y) = true \Leftrightarrow \exists A_i \in A : (x,y) \subseteq A_i$

Il vantaggio dei classificatori di accoppiamento sta nel fatto che possono essere facilmente confrontati usando i classici coefficienti per il confronto tra classificazioni supervisionate, dimostreremo nei prossimi paragrafi che in questo modo si ottengono molti dei coefficienti introdotti in §1.4.4.

5.2.1 Matrice di confusione tra i classificatore di accoppiamento

Date due classificazioni A e B con $A = \{A_i\}_{i=1..C_a}$ e $B = \{B_j\}_{j=1..C_b}$ dello stesso data base associando ad esse gli eventi α^I e β^I come spiegato in 5.2 abbiamo che:

$P(\alpha^I)$**:** Probabilità che due dati appartengano a un cluster della classificazione A.

$P(\beta^I)$**:** Probabilità che due dati appartengano a un cluster della classificazione B.

$P(\alpha^I, \beta^I)$**:** Probabilità che due dati appartengano contemporaneamente al medesimo cluster della classificazione A e B.

$P(\alpha^I, \neg\beta^I)$**:** Probabilità che due dati appartengano a uno stesso cluster della classificazione A e a due cluster diversi della classificazione B.

$P(\neg\alpha^I, \beta^I)$**:** Probabilità che due dati appartengano a due cluster diversi della classificazione A ed allo stesso cluster della classificazione B

$P(\neg\alpha^I, \neg\beta^I)$**:** Probabilità che due dati non appartengano ad uno stesso cluster sia della classificazione A che della classificazione B.

Queste probabilità sono semplici da calcolare partendo dalla matrice di confusione:

$$\begin{pmatrix} n_{11} & \dots & n_{1C_b} \\ \dots & \dots & \dots \\ n_{C_a 1} & \dots & n_{C_a C_b} \end{pmatrix} \tag{5.2}$$

Si ha infatti che:

$P(\alpha)^I = \frac{\sum_{i=1}^{C_a} n_{i+}^2}{n^2}$

$P(\beta^I) = \frac{\sum_{i=1}^{C_b} n_{i+}^2}{n^2}$

$P(\alpha^I, \beta^I) = \frac{\sum_{i,j=1}^{C_a,C_b} n_{ij}^2}{n^2}$

$P(\alpha^I, \neg\beta^I) = \frac{\sum_{i,j=1}^{C_a,C_b} n_{ij} * (n_{i+} - n_{ij})}{n^2}$

$P(\neg\alpha^I, \beta^I) = \frac{\sum_{i,j=1}^{C_a,C_b} n_{ij} * (n_{+j} - n_{ij})}{n^2}$

$P(\neg\alpha^I, \neg\beta^I) = 1 - P(\alpha^I) - P(\beta^I) + P(\alpha^I, \beta^I)$

dove:

$n = \sum_{i,j=1}^{C_a,C_b} n_{ij}$

$n_{i+} = \sum_{j=1}^{C_b} n_{ij}$

$n_{+j} = \sum_{i=1}^{C_a} n_{ij}$

Analogamente alle stesse classificazioni si possono associare gli eventi α^{II} e β^{II}, le formule ricavate in questo caso cambiano leggermente, essendo il numero di coppie di un insieme di dimensioni N pari a $N(N-1)/2$. Abbiamo quindi che:

$P(\alpha^{II}) = \frac{\sum_{i=1}^{C_a} n_{i+}/(n_{i+}-1)}{n(n-1)}$

$P(\beta^{II}) = \frac{\sum_{j=1}^{C_b} n_{+j}(n_{+j}-1)}{n(n-1)}$

$P(\alpha^{II},\beta^{II}) = \frac{\sum_{i,j=1}^{C_a,C_b} n_{ij}(n_{ij}-1)}{n(n-1)}$

$P(\neg\alpha^{II},\neg\beta^{II}) = 1 - P(\alpha^{II}) - P(\beta^{II}) + P(\alpha^{II},\beta^{II})$

Con queste quantità è possibile ricavare le matrice di confusione normalizzate a 1 tra i classificatori A^I,B^I e A^{II},B^{II}.

$$Mdc(A^I,B^I) \equiv \begin{pmatrix} P(\neg\alpha^I,\neg\beta^I) & P(\neg\alpha^I,\beta^I) \\ P(\alpha^I,\neg\beta^I) & P(\alpha^I,\beta^I) \end{pmatrix} Mdc(A^{II},B^{II}) \equiv \begin{pmatrix} P(\neg\alpha^{II},\neg\beta^{II}) & P(\neg\alpha^{II},\beta^{II}) \\ P(\alpha^{II},\neg\beta^{II}) & P(\alpha^{II},\beta^{II}) \end{pmatrix}$$

.

5.2.2 Confronto tra classificatori di accoppiamento

I classificatori di accoppiamento sono dei semplici classificatori booleani ed è possibile usare per loro i coefficienti che vengono classicamente usati per questo caso 1.4.1. Partendo dalle matrici di confusione introdotte nel paragrafo precedente applicando i coefficienti introdotti in 1.4.1 si ha che:

$\mathbf{TN}(A^I,B^I) = \frac{P(\neg\alpha^I,\neg\beta^I)}{P(\neg\alpha^I,\neg\beta^I)+P(\neg\alpha^I,\beta^I)} = P(\neg\alpha^I|\neg\beta^I)$

$\mathbf{TP}(A^I,B^I) = \frac{P(\alpha^I,\beta^I)}{P(\alpha^I,\neg\beta^I)+P(\alpha^I,\beta^I)} = P(\alpha^I|\beta^I)$

$\mathbf{FP}(A^I,B^I) = \frac{P(\neg\alpha^I,\beta^I)}{P(\neg\alpha^I,\neg\beta^I)+P(\neg\alpha^I,\beta^I)} = P(\neg\alpha^I|\beta^I)$

$\mathbf{FN}(A^I,B^I) = \frac{P(\alpha^I,\neg\beta^I)}{P(\alpha^I,\neg\beta^I)+P(\alpha^I,\beta^I)} = P(\alpha^I|\neg\beta^I)$

$\mathbf{P}(A^I,B^I) = \frac{P(\alpha^I,\beta^I)}{P(\neg\alpha^I,\beta^I)+P(\alpha^I,\beta^I)} = P(\beta^I|\alpha^I)$

Overall Accuracy$(A^I,B^I) = P(\alpha^I,\beta^I)+P(\neg\alpha^I,\neg\beta^I)$

Average Accuracy$(A^I,B^I) = \frac{1}{2}(P(\alpha^I|\beta^I)+P(\neg\alpha^I|\neg\beta^I))$

Sostituendo in $G-mean_1$ $G-mean_2$ e F i valori TP,TN e P calcolati tramite la matrice di confusione delle probabilità di raggruppamento, abbiamo che:

- $G-mean_1(A^I,B^I) = \sqrt{P(\alpha^I|\beta^I)*P(\beta^I|\alpha^I)}$
- $G-mean_2(A^I,B^I) = \sqrt{P(\alpha^I|\beta^I)*P(\neg\alpha^I|\neg\beta^I)}$
- $F(A^I,B^I) = \frac{(\beta^2+1)*P(\beta^I|\alpha^I)*P(\alpha^I|\beta^I)}{\beta^2*P(\beta^I|\alpha^I)+P(\alpha^I|\beta^I)}$

Infine sostituendo i K coefficienti abbiamo che:

$\mathbf{K}(A^I,B^I) = \frac{P(\alpha^I,\beta^I)-P(\alpha^I)P(\beta^I)}{(P(\alpha^I)+P(\beta^I))/2-P(\alpha^I)P(\beta^I)}$

$\hat{K}(A^I,B^I) = \frac{1}{2}(\frac{P(\alpha,\beta)-P(\alpha)*P(\beta)}{P(\beta)-P(\alpha)P(\beta)}+\frac{P(\alpha,\beta)-P(\alpha)*P(\beta)}{P(\alpha)-P(\alpha)P(\beta)})$

$P(\alpha^I|\beta^I)$ è la probabilità che presi due dati appartenenti ad un solo cluster della classificazione B questi appartengono ad un solo cluster della classificazione A. $P(\beta^I|\alpha^I)$ invece è la probabilita che due elementi appartenenti ad uno stesso cluster della classificazione A appartengono ad uno stesso cluster della classificazione B.

Dimostrazioni:

$$\begin{aligned} K(A^I,B^I) &= \frac{P_o-P_e}{1-P_e} = \frac{[P(\alpha^I,\beta^I)+P(\neg\alpha^I,\neg\beta^I)]-[P(\alpha^I)P(\beta^I)+P(\neg\alpha^I)P(\neg\beta^I)]}{1-[P(\alpha^I)P(\beta^I)+P(\neg\alpha^I)P(\neg\beta^I)]} \\ &= \frac{2P(\alpha^I,\beta^I)-2P(\alpha^I)P(\beta^I)}{(P(\alpha^I)+P(\beta^I)-2P(\alpha^I)P(\beta^I)} = \\ &= \frac{P(\alpha^I,\beta^I)-P(\alpha^I)P(\beta^I)}{(P(\alpha^I)+P(\beta^I))/2-P(\alpha^I)P(\beta^I)} \end{aligned}$$

$$
\begin{aligned}
\hat{K}(A^I,B^I) &= \frac{1}{2}[\hat{K}_1+\hat{K}_2] = \\
\hat{K}_1 &= \left(\frac{P(\neg\alpha^I,\neg\beta^I)-P(\neg\alpha^I)P(\neg\beta^I)}{[P(\neg\alpha^I,\neg\beta^I)+P(\neg\alpha^I,\beta^I))-P(\neg\alpha^I)P(\neg\beta^I)]}\right) = \\
&= \frac{P(\neg\alpha^I,\neg\beta^I)-P(\neg\alpha^I)P(\neg\beta^I)}{P(\neg\alpha)-P(\neg\alpha^I)P(\neg\beta^I)]}) = \\
&= \frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{P(\beta^I)-P(\alpha^I)P(\beta^I)} \\
\hat{K}_2 &= \frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{[(P(\alpha^I,\neg\beta)+P(\alpha^I,\beta^I))-P(\alpha^I)P(\beta^I)]} = \\
&= \frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{P(\alpha^I)-P(\alpha^I)*P(\beta^I)} \\
\hat{K}(A^I,B^I) &= \frac{1}{2}\left(\frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{P(\beta^I)-P(\alpha^I)P(\beta^I)}+\frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{P(\alpha^I)-P(\alpha^I)P(\beta^I)}\right)
\end{aligned}
$$

Questi coefficienti fanno riferimento al confronto dei classificatori di accoppiamento con ripetizione ma è possibile da questi ricavare quelli del confronto fra classificatori di accoppiamento sulle coppie distinte semplicemente sostituendo gli eventi α^I con α^{II} e β^I con β^{II}.

5.3 Il collegamento tra due famiglie di coefficienti

Prese due generiche classificazioni A e B è possibile associare due coppie di classificatori di accoppiamento A^I,B^I e A^{II},B^{II}. Applicando a queste le formule usate classicamente per confrontare le classificazioni booleane si ricavano una serie di coefficienti che confrontano le classificazioni A e B.

La bontà di questo approccio è provata dal fatto che la maggior parte dei coefficienti ricavati in questo modo sono già usati da tempo. In questo paragrafo dimostreremo, infatti, come i coefficienti ricavati in §1.4.4 sono riconducibili con l'uso dei classificatori di accoppiamento a quelli ricavati in §5.2.2.

Partendo dal rand index (Eq. 1.25) riscriviamo i coefficenti N_{ij}. Per esempio, il coefficiente N_{11} corrisponde al numero di coppie che contemporaneamente sono presenti in un cluster di due classificazioni A e B. Infatti $P(\alpha^{II},\beta^{II})$ è la probabilità che una coppia stia in un cluster in entrambi le classificazioni, si ha

che $N_{11} = P(\alpha^{II}, \beta^{II}) * (n * (n-1)/2)$; generalizzando il ragionamento agli altri indici abbiamo che:

- $N_{00} = P(\neg\alpha^{II}, \neg\beta^{II}) * (n * (n-1)/2)$
- $N_{01} = P(\alpha^{II}, \neg\beta^{II}) * (n * (n-1)/2)$
- $N_{10} = P(\neg\alpha^{II}, \beta^{II}) * (n * (n-1)/2)$

sostituendo i valori ottenuti in (Eq. 1.25), abbiamo che:

$$R(A,B) = \frac{N_{11} + N_{00}}{n * (n-1)/2} = P(\neg\alpha^{II}, \neg\beta^{II}) + P(\alpha^{II}, \beta^{II}) = Acc(A^I, B^I) \quad (5.3)$$

con ragionamento analogo otteniamo che il $Rand^I(A,B) = Acc(A^I, B^I)$.
I W index (vedi Eqq. 1.26 e 1.27)

$$W_I(A,B) = \frac{N_{11}}{\sum_{i=1}^{C_a} n_{i+} * (n_{i+} - 1)}) = \frac{P(\alpha^{II}, \beta^{II})}{P(\alpha^{II})} = TP(A^{II}, B^{II}) \quad (5.4)$$

$$W_{II}(A,B) = \frac{N_{11}}{\sum_{j=1}^{C_b} n_{+j} * (n_{+j} - 1)} = \frac{P(\alpha^{II}, \beta^{II})}{P(\beta^{II})} = P(A^{II}, B^{II}) \quad (5.5)$$

Fwl index Eq. 1.28

$$\begin{aligned} Fwl(A,B) &= \sqrt{W_I(A,B) * W_{II}(A,B)} = \sqrt{TP(A^{II}, B^{II}) P(A^{II}, B^{II})} = \\ &= G - mean_1(A^{II}, B^{II}) \end{aligned}$$

Mirkin metric (vedi Eq.1.30):

$$\begin{aligned} Mrk(A,B) &= \sum_{i=1}^{C_a} (n_{i+})^2 + \sum_{j=1}^{C_b} (n_{+j})^2 - 2 \sum_{i,j=1}^{C_a, C_b} n_{ij}^2 = \\ &= n^2 (P(\alpha^I) + P(\beta^I) - 2P(\alpha^I, \beta^I)) = \\ &= n^2 (1 - Acc(A^I, B^I)) = d_{AC}(A^I, B^I) \end{aligned}$$

Il Jacard index dell'Eq. 1.29 :

$$\begin{aligned} \frac{N_{11}}{N_{11} + N_{10} + N_{01}} &= \frac{P(\alpha^{II}, \beta^{II})}{P(\alpha^{II}, \beta^{II}) + P(\alpha^{II}, \neg\beta^{II}) + P(\neg\alpha^{II}, \beta^{II})} = \\ &= \frac{P(\alpha^{II}, \beta^{II})}{P(\alpha^I) + P(\neg\alpha^{II}, \beta^{II})} = \\ &= \frac{P(\alpha^{II}, \beta^{II})}{P(\beta^{II}) + P(\alpha^{II}, \neg\beta^{II})} = IAC(A^{II}, B^{II}) \end{aligned}$$

Coeff	Formule	Forumua espliceita
Litt$\longrightarrow$ $AC(A^{II},B^{II})$	Rand(A,B) $P(\alpha^{II},\beta^{II})+P(\neg\alpha^{II},\neg\beta^{II})$	$1+\frac{2\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2}-\sum_{i=1}^{C_a}\binom{n_{i+}}{2}-\sum_{j=1}^{C_b}n_{j+}^2}{\binom{n}{2}}$
Litt$\longrightarrow$ $AC(A^{I},B^{I})$	$Rand^I(A,B)$ $P(\alpha^{I},\beta^{I})+P(\neg\alpha^{I},\neg\beta^{I})$	$1+\frac{2\sum_{i,j=1}^{C_a,C_b}n_{ij}^2-\sum_{i=1}^{C_a}n_{i+}^2-\sum_{j=1}^{C_b}n_{j+}^2}{n^2}$
Litt$\longrightarrow$ $TP(A^{II},B^{II})$	$W_I(A,B)$ $P(\alpha^{II}\|\beta^{II})$	$\frac{\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2}}{\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}$
Litt$\longrightarrow$ $Pr(A^{II},B^{II})$	$W_{II}(A,B)$ $P(\beta^{II}\|\alpha^{II})$	$\frac{\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2}}{\sum_{j=1}^{C_a}\binom{n_{i+}}{2}}$
Litt$\longrightarrow$ $d_{AC}(A^{I},B^{I})$	$K(A,B)$ $(1-AC(A^{I},B^{I}))*n^2$	$\sum_{i=1}^{C_a}n_{i+}^2+\sum_{j=1}^{C_b}n_{j+}^2-2\sum_{i,j=1}^{C_a,C_b}n_{ij}^2$
Litt$\longrightarrow$ $IAC(A^{II},B^{II})$	$J(A,B)$ $\frac{P(\alpha^{II},\beta^{II})}{P(\beta^{II})+P(\alpha^{II},\neg\beta^{II})}$	$\sum_{i,j=1}^{C_a,C_b}\frac{\binom{n_{ij}}{2}}{\binom{n_{+j}}{2}+n_{ij}*(n_{+j}-n_{ij})}$
Litt$\longrightarrow$ $K(A^{II},B^{II})$	$Rand^I(A,B)$ $\frac{P(\alpha^{II},\beta^{II})-P(\alpha^{II})*P(\beta^{II})}{\frac{P(\alpha^{II})+P(\beta^{II})}{2}-P(\alpha^{II})*P(\beta^{II})}$	$\frac{\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2}-1/\binom{n}{2}\sum_{i=1}^{C_a}\binom{n_{i+}}{2}\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}{1/2[\sum_{i=1}^{C_a}\binom{n_{i+}}{2}+\sum_{j=1}^{C_b}\binom{n_{+j}}{2}]-[1/\binom{n}{2}]\sum_{i=1}^{C_a}(\binom{n_{i+}}{2})\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}$

Tabella 5.1: Formule derivate già presenti in letteratura ([44, 45, 48, 47, 52, 53])

L'adjusted rand index (Eq. 1.31):

$$\begin{aligned} Rand^I(A,B) &= \frac{\sum_{i,j=1}^{C_a,C_b}\binom{n_{ij}}{2}-1/\binom{n}{2}\sum_{i=1}^{C_a}\binom{n_{i+}}{2}\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}{1/2[\sum_{i=1}^{C_a}\binom{n_{i+}}{2}+\sum_{j=1}^{C_b}\binom{n_{+j}}{2}]-[1/\binom{n}{2}]\sum_{i=1}^{C_a}(\binom{n_{i+}}{2})\sum_{j=1}^{C_b}\binom{n_{+j}}{2}}= \\ &= \frac{P(\alpha^{II},\beta^{II})-P(\alpha^{II})*P(\beta^{II})}{(P(\alpha^{II})+P(\beta^{II}))/2-P(\alpha^{II})*P(\beta^{II})}= \\ &= K(A^{II},B^{II}) \end{aligned}$$

5.4 Interpretazioni dei coefficienti ricavati

I coefficienti ricavati in 5.2.2 sono sempre in funzione delle probabilità degli eventi α e β e questo permette una facile interpretazione.

5.4.1 Il significato di $P(\alpha,\beta)$ e $P(\neg\alpha,\neg\beta)$ le Accuracy

$P(\alpha^I,\beta^I)$ è la probabilita prendendo due generici dati che questi risultano appartenere contemporaneamente ad un solo cluster sia delle prima che della seconda classificazione mentre $P(\neg\alpha^I,\neg\beta^I)$ è la probabilità dati due dati che questi non appartengono ad uno stesso cluster sia nella prima che nella seconda classificazione. La somma di queste due probablità congiunte da la probabilità che una decisione sia corretta, in effetti il clustering viene visto come un algoritmo di raggruppamento quindi per decisioni si intende se raggruppare o meno due dati, dato ora un clustering di riferimento chiamato B, A effettua una decisione corretta se è la stessa di quelle fatta da B. Dato che le decisioni sono 2 raggruppare assieme o non raggruppare la probabilità che una decisione sia corretta è la somma delle due probabilità $P(\alpha^I,\beta^I)$ e $P(\neg\alpha^I,\neg\beta^I)$.
Per gli eventi α^{II} e β^{II} il ragionamento è analogo.
L'$AC(A^I,B^I)$ è la probabilità che ha la classificazione A di effettuare la stessa decisione della classificazione B .
Questa misura però ha alcuni problemi derivanti dal fatto che generalmente $P(\neg\alpha^I,\neg\beta^I) >> P(\alpha^I,\beta^I)$ sopratutto quando si confrontano classificazioni molto raffinate, la conseguenza è che nella misura dell AC predomina il secondo fattore.

5.4.2 Concetto di inclusioni tra classificazioni

Date due classificazioni non supervisionate $A = \{A_i\}_{i=1..C_a}$ e $B = \{B_i\}_{i=1..C_b}$ diciamo che A è un raffinamento di B e scriveremo $A \subseteq_R B$ se e solo se $\forall i \exists j : A_i \subseteq B_j$
Se $A \subseteq_R B$ e $B \subseteq_R A$ diremo che $A \equiv_R B$ abbiamo che: $\forall i \exists j : A_i \equiv B_j$ consegue che i due insiemi dei cluster coincidono.
Se $A \subseteq_R B$ si ha che $\forall i \forall (x,y) \in A_i \times A_i \; \exists j : (x,y) \in B_j \times B_j$.

$P(\alpha^I|\beta^I)$ è la probabilita che presi due dati appartenenti ad un solo cluster della classificazione B questi appartengono ad un solo cluster della classificazione A. $P(\beta^I|\alpha^I)$ invece è la probabilita che due elementi appartenenti ad uno stesso cluster della classificazione A appartengono ad uno stesso cluster della classificazione B.
Si ha che $P(\beta^I|\alpha^I) = 1 \Leftrightarrow A \subseteq_R B$ e $P(\alpha^I|\beta^I) = 1 \Leftrightarrow B \subseteq_R A$
$P(\alpha^{II}|\beta^{II})$ e $P(\beta^{II}|\alpha^{II})$ si differenziano dai rispettivi $P(\alpha^I|\beta^I)$ e $P(\beta^I|\alpha^I)$ solo per il fatto che invece di scegliere due dati appartenenti ad un solo cluster si prendono le coppie distinte incluse nel cluster ne consegue che godono delle stesse proprietà.
Più le probabilità condizionate tra α e β si avvicinano a 1 e più le classificazioni A e B sono incluse l'una nell'altra al limite coincidono.

5.4.3 Accordo casuale tra classificazioni

Uno dei problemi di base nelle misure precedenti è che nessuna di queste prende in considerazione il possibile accordo casuale tra le classificazioni.

Questo problema esistente comunque in generale nel caso di qualsiasi confronto fra classificazioni supervisionate è ancora più accentuato nel nostro caso.
I classificatori booleani di accoppiamento hanno dei vincoli che i classificatori normali non hanno è questo pregiudica una serie di possibilità.
In primis, ad esempio, la matrice di confusione associata ai classificatori di accoppiamento ha i termini della diagonale maggiore strettamente positivi. Conseguentemente tutti i coefficienti Pr (eq. 1.9) TP (eq. 1.8) Acc (eq 1.10) sono sempre diversi da 0.

Applicando i kappa coefficient ai classificatori di accoppiamento otteniamo dei coefficienti che prendono in considerazione l'accordo casuale tra le classificazioni. Questi sono uguali a 0 nel caso sia verificata l'ipotesi nulla, cioé che le classificazioni sono tra loro distribuite in maniera casuale e sono uguali a 1 nel caso sia verificata l'ipotesi alternativa che le classificazioni coincidono completamente.

I valori dei kappa coefficient possono anche risultare negativi questo vuol dire che c'è addirittura anticorrelazione.
Nel caso del confronto tra i classificatori di accoppiamento l'interpretazione dell'anti correlazione fra essi non è cosi immediata ne così significativa come nel caso del confronto fra classificatori classici. In generale si è interessati più alla vicinanza rispetto all'ipotesi alternativa che alla distanza rispetto all'ipotesi nulla, analogamente a quello che si fá in un z-test ad una coda.
Date due classificazioni in linea teorica sarebbe possibile determinare una distribuzione statistica e associare ai k-coefficient calcolati un p-value, questo valore sarebbe legato oltre che ai k-coefficient (distanza dall'ipotesi nulla e alternativa) anche alle dimensioni dell'insieme classificato nonché alle distribuzioni delle classificazioni confrontate.
Questo tipo di analisi può risultare utile ed è uno dei possibili sviluppi di questo lavoro.

5.5 I coefficienti di accoppiamento e disaccopiamento

5.5.1 Utilizzo del conditional kappa coefficient

La caratteristica del conditional kappa coefficient è che viene calcolato rispetto alle singole classi, il conditional kappa coefficient è quindi un insieme di coefficienti uno per ogni classe da cui ad esempio mediando se ne può estrarre uno generale.
Nel caso dei classificatori di accoppiamento le classi sono due: la prima che chiameremo *acc* (accoppiamento) è quella che dice che una coppia o due dati generici (a secondo dell'evento considerato) appartengono ad uno stesso cluster della classificazione di partenza, la seconda che chiameremo *disacc* (disaccoppiamento) dice il contrario.
Riportando i risultati precedentemente ottenuti abbiamo che:

- $\hat{K}_{disacc}(A^I, B^i) = \frac{P(\alpha^I,\beta^I)-P(\alpha^I)*P(\beta^I)}{P(\beta^I)-P(\alpha^I)P(\beta^I)}$

- $\hat{K}_{acc}(A^I, B^I) = \frac{P(\alpha^I,\beta^I) - P(\alpha^I) * P(\beta^I)}{P(\alpha^I) - P(\alpha^I)P(\beta^I)}$
- $\hat{K}(A^I, B^I) = \frac{1}{2}(\hat{K}_{acc}(A^I, B^I) + \hat{K}_{disacc}(A^I, B^I))$

La $\hat{K}_{disacc}(A^I, B^i)$ è discontinua per $P(\alpha) = 1$ mentre $\hat{K}_{acc}(A^I, B^I)$ per $P(\beta) = 1$ questi valori corrispondono al caso in cui una delle due classificazioni raggruppa tutti i dati in un solo cluster. Per completezza e dato che queste due situazioni anche se banali sono accettabili poniamo in questi punti i valori di $\hat{K}_{acc}(A^I, B^I)$ e $\hat{K}_{disacc}(A^I, B^I))$ nella seguente maniera:

- $\hat{K}_{acc}(A^I, B^I) = 0$ se $P(\beta) = 1 \wedge P(\alpha) < 1)$
- $\hat{K}_{disacc}(A^I, B^I) = 0$ se $P(\alpha) = 1 \wedge P(\beta) < 1$
- $\hat{K}_{acc}(A^I, B^I) = \hat{K}_{disacc}(A^I, B^I) = 1$ se $P(\alpha) = 1 \wedge P(\beta) = 1$

I conditional kappa coefficient applicati ai classificatori di accoppiamento godono delle seguenti proprietà:

- Se $P(\alpha^I) = P(\beta^I)$ allora $\hat{K}_{disacc} = \hat{K}_{acc}$ La differenza tra il coefficiente di accoppiamento e quello di disaccoppiamento dipende solo dal diverso ordine di raffinatezza delle classificazioni confrontate.
- Se $P(\alpha^I) > P(\beta^I)$ allora $\hat{K}_{disacc} > \hat{K}_{acc}$
- Se $P(\alpha) < P(\beta)$ allora $\hat{K}_{disacc} < \hat{K}_{acc}$
- $\hat{K}_{disacc}(A^I, B^I) = \hat{K}_{acc}(B^I, A^I)$ ed il viceversa. Sono a specchio.
- $\hat{K}_{acc} = 0 \Longleftrightarrow \hat{K}_{disacc} = 0$

La coppia $(\hat{K}_{disacc}(A^I, B^I), \hat{K}_{acc}(A^I, B^I))$ sintetizza tutte le informazioni necessarie per confrontare le classificazioni A e B è può essere comodamente visualizzata in un piano.
Il punto (1,1) significa che le due classificazioni coincidono a meno di permutazioni, i punti (x,1), con $x \in [0, 1]$, rappresentano l'insieme dei valori delle classificazioni che includono una classificazione di riferimento B, i punti (1,x) rappresentano l'insieme delle classificazioni incluse in B ed infine il punto (0,0) è il caso in cui le classificazioni sono completamente scorrelate.

5.5.2 Possibili applicazioni

Per valutare la bontá di un algoritmo di classificazione unsupervised nel determinare classificazione giá note è necessario capire il meglio che l'algoritmo stesso possa fare e valutarlo in base a questo meglio. Come accennato questi algoritmi hanno in ingresso alcuni parametri (e.g. numero di cluster, densita o raggio massimo dei cluster) è il meglio vá definito in funzione di questi parametri.

Nel caso ad esempio del K-means è necessario fissare il numero di cluster a priori, gli algoritmi gerarchici producono un insieme di classificazioni incluse le une nelle altre e successivamente un criterio esterno o anche un osservatore ne sceglie una. Entrambi i criteri influiscono in maniera determinante sull'ordine di raffinatezza della classificazione risultante. Dato un insieme preclassificato si può fissare ad hoc questo parametro esterno ed ottenere dei risultati più o meno buoni, in questo caso però si dà all'algoritmo delle informazioni iniziali che sono note sull'insieme preclassificato ma non sull'insieme più ampio, la validazione sull'insieme preclassificato perde in affidabilità.

Nel caso in cui si usano degli algoritmi esterni che vengono accoppiati ad esempio al K-means per determinare il numero di cluster corretti, l'errore dipende da entrambi gli algoritmi ed è difficile capire quanto dell'errore dipende dall'algoritmo per determinare il numero di cluster e quanto invece dal K-means.

Il punto è che per valutare la bontà del K-means dovremmo escludere la componente dell'errore dipendente dalla scelta del numero di cluster iniziali, cioé valutare la bonta dei risultati condizionata al fatto che il numero di cluster iniziali è fissato.

Si consideri ora di voler classificare un insieme preclassificato con k cluster utilizzando un K-means, fissando inizialmente m cluster con $m > k$, qual'è la migliore classificazione che si può sperare? Le due classificazioni non possono coincidere, al meglio posso sperare che, essendo $m > k$, la classificazione ottenuta col K-means sia inclusa in quella di riferimento, analogamente se $m < k$ posso sperare che al meglio la include.

Questa informazione è contenuta solo nel fatto che almeno uno della coppia

di coefficienti ($\hat{K}_{acc}$, $\hat{K}_{disacc}$) sia pari al massimo valore raggiungibile: 1.

Chiamato ad esempio A_M il risultato di una classificazione di n dati ottenuta con il K-means fissando M cluster e chiamiamo B un classificatore di riferimento dello stesso insieme con K cluster mi aspetto al meglio che se $M > k$ allora $\hat{K}_{disac} = 1$ se $M < k$ allora $\hat{K}_{ac} = 1$ in generale per M qualunque e fissato posso dire che al meglio mi aspetto che $Max(\hat{K}_{disac}, \hat{K}_{ac}) = 1$, Il discorso si può estendere a qualunque classificazione ottenuta da un algoritmo una volta che si è fissato inizialmente un parametro λ che determina la raffinatezza della classificazione: *la migliore classificazione ottenibile rispetto ad una di riferimento condizionata alla scelta del parametro λ è una qualunque che gode della proprietà:* $Max(\hat{K}_{disac}(A^I_\lambda, R^I), \hat{K}_{ac}(A^I_\lambda, R^I)) = 1$ dove A_λ è la classificazione ottenuta una volta fissato λ,R la classificazione di riferimento.

Capitolo 6

Conclusioni e sviluppi

6.1 Conclusioni

6.1.1 PDM

Il PDM è un algoritmo di data mining unsupervised che estrae le informazioni non analizzando i dati ma i predatori dei dati. I predatori sono dei possibili centri di aggregazione, il PDM prima cerca i migliori centri di aggregazione e poi classifica i dati rispetto a questi. Utilizza inizialmente un algoritmo che simula una dinamica evolutiva è produce una serie di popolazioni di predatori. Queste popolazioni, dalla pressione alla selezione, tendono ad adattarsi all'ambiente e, per questo, hanno informazioni importanti sull'ambiente stesso. L'ambiente è il data base da analizzare. Alcune di queste popolazioni vengono analizzate e per ognuna di queste si estraggono un numero non fisso di predatori che vengono utilizzati come centri di aggregazione per classificate l'insieme dei dati. In questa maniera si producono un certo numero di classificazioni differenti, si misura il disaccordo fra queste classificazioni e si sceglie quella che minimizza il disaccordo medio con le altre. Il disaccordo misura la capacità dei predatori scelti e del PDM di discriminare i dati analizzati.

Lo scopo di questa tesi era introdurre e validare il modello, per questo motivo viene principalmente applicato per risolvere problemi di Data Clustering. Il

PDM, infatti, può, col predatore adatto, essere utilizzato come un classico algoritmo di data clustering. Lo si è testato su dati che simulano il comportamento dei microarray, dati utilizzati da Thalamuthu ed il suo gruppo ([64]) per confrontare i principali algoritmi di data clustering utilizzati in quest'ambito. I risultati ottenuti dal PDM si inseriscono perfettamente in questo confronto. Considerando che si determinano automaticamente il numero di cluster e che il PDM è poco sensibile ai parametri esterni, i risultati possono essere più che soddisfacenti e si può affermare che la metodologia è valida.

L'approccio originale al problema permette però di generalizzare l'operazione di data clustering aprendo nuove strade all'analisi dei dati. Nel preciso sono tre gli aspetti che caratterizzano il modello:

- La possibilità di progettare dei metodi di aggregazione che non dipendono da un criterio di similarità o dissimilarità fra i dati.
- La possibilità di lavorare su qualsiasi tipo di dato anche affetti da *missing value*.
- L'autovalutazione della qualità del risultato.

.

L'autovalutazione del risultato: l'Ac_{PDM} anche, ad esempio, nell'operazione di Data Clustering ha una notevole importanza. Se, come abbiamo visto il PDM sbaglia più o meno come gli altri algoritmi di data clustering a differenza di questi lo dice. Inoltre può risultare un potente strumento nella ricerca della classe di predatore più adatta al problema.

Un esempio delle possibilità del PDM di lavorare su qualsiasi tipo di dato viene mostrato in 4. Il PDM è stato applicato a microarray con *missing value* è si sono potuti notare dei notevoli vantaggi sia in termini di dati analizzabili che di qualità dei cluster ottenuti nel lavorare direttamente su questi dati invece che utilizzare complicati algoritmi di preprocessing.

Quest'applicazione mostra inoltre le notevoli possibilità pratiche del PDM. In questo caso non si è limitati a raggruppare i dati rispetto ad un criterio di somiglianza ma, utilizzando come predatori delle rette, si è usato un metodo di

aggregazione differente: i dati negli stessi cluster non sono vicini ma allineati. I cluster cosifatti hanno mostrato una buona per certi ottima significatività biologica. Ovviamente si potrebbe continuare a provare predatori differenti e l'Ac_{PDM} può dare un indicazione significativa della bontà delle classificazioni ottenute.

6.1.2 Coefficienti per confrontare classificazioni

In questa tesi è stato presentato un approccio al problema del confronto fra classificazioni tramite la definizione di particolari classificatori booleani associati ai clustering (§5.2). Quest'approccio si basa su un concetto già ampiamente usato e accettato: misurare la capacità di due classificatori di raggruppare similmente i dati (§1.4.4), ma la formalizzazione adottata permette di unificare buona parte dei coefficienti utilizzati per il confronto fra classificazioni non supervisionate con quella esistente per il confronto tra quelle supervisionate (§5.3). Inoltre in questa maniera è stato individuato un nuovo coefficiente (§5.5.1) che in certe situazioni sembra essere preferibile a quello generalmente usato in questi casi (Eq. 3.1). Questo nuovo coefficiente è stato ricavato usando il conditional kappa coefficient tra i classificatori booleani associati alle classificazioni da confrontare, esso mantiente tutte le caratteristiche dell'adjusted rand index ma ha delle informazioni in più estremamente utili. In sintesi fin quando si confrontano in assoluto due classificazioni si può usare equivalentemente l'uno o l'altro, ma se si vuole calcolare la bontà di un algoritmo in cui si fissa un parametro iniziale (tipo il numero di cluster) bisogna valutare la bontà condizionata alla scelta del parametro e il nuovo coefficiente introdotto è preferibile.

6.2 Sviluppi del PDM

La possibilità di definire a secondo del problema affrontato centri di aggregazione diversi e quindì di impostare dei metodi di aggregazione diversi dal semplice raggruppamento rispetto ad un criterio di somiglianza o metrica fra i dati apre una nuova strada all'analisi dei dati. Si può infatti cercare tra tutti gli infiniti possibili centri di aggregazione quello che è più adatto al tipo di informazione che si ha

interesse estrarre dal data base. Ad esempio se si è interessati a scoprire correlazioni lineari tra i dati è preferibile usare come centri di aggregazione delle rette piuttosto che dei punti. Inoltre l'algoritmo è indipendente dalla struttura sia dei dati che dei centri di aggregazione e può lavorare anche con missing value questo significa che in linea teorica può lavorare su qualsiasi tipo di data base relazionale. In altri termini l'applicazione al data clustering del PDM, che ha comunque alcuni aspetti che la possono far preferire in certe applicazioni ad altre, può essere vista solo come un lavoro introduttivo. Un algoritmo di base che ha nelle infinite possibili definizioni di predatori infiniti possibili sviluppi applicativi.

Il PDM ha inoltre come algoritmo altre possibilità, uno dei prossimi sviluppi è di scartare i dati troppo rumorosi. Questo lo si può fare in due modi indipendenti l'uno dall'altro: il primo valutando ed analizzando le "distanze" dai centri di aggregazioni finali, il secondo, osservando le classificazioni ottenute dal PDM analizzando differente popolazioni, valutando quei dati che risultano rispetto ad altri meno legati a gruppi.

L'implementazione dell'algoritmo non è ancora disponibile al pubblico essendo l'algoritmo stesso in una fase di test. I tempi di calcolo comunque consistenti fanno optare alla possibilità futura di un ottimizzazione e dare la possibilità di parallelizzare l'esecuzione su differenti macchine. Un interfaccia grafica *user friendly*, una capacità di gestire i dati da differenti data base ed una accurata guida per la scrittura di nuovi predatori sono i prossimi sviluppi informatici.

Bibliografia

[1] C. Del Mondo, F. Lizzi, *PDM Predator Data Mining* In lavorazione.

[2] C. Del Mondo *Un metodo evolutivo per la ricerca di predicati interessanti: un nuovo approccio per il Data Mining Unsupervised.* Tesi di Laurea in Fisica Federico II Napoli (2004)

[3] A. Ciaramella,C. Del Mondo, et al., *Novel Techniques for Microarray Data Analysis: Probabilistic Principal Surfaces and Competitive Evolution on Data* Journal of Computational and Theoretical Nanoscience, Volume 2, Number 4, December 2005 , pages 514-523(10)

[4] R.Amato,C. Del Mondo, A. Ciaramella et al., *A multi-step approach to time series analysis and gene expression clustering.* Bioinformatics, Vol. 22 no. 5 2006

[5] L. Kaufman and P.J. Rousseeuw *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, New York, 1990

[6] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications.* Data Mining and Knowledge Discovery Volume 11, Number 1 pages 5-33 2005

[7] W. Wang, J. Yang and R. Muntz. *STING: a statistical information grid approach to spatial data mining.* Conference on VLDB, pages 186–195, Athens, 1997

[8] G. Sheikholeslami, S. Chatterjee and A. Zhang. *Wavecluster: a multi-resolution clustering approach for very large spatial databases.* Conference on VLDB, pages 428–439, New York, 1998

[9] R. Sibson. *SLINK: an optimally efficient algorithm for the single link cluster method.* Computer Journal, Volume 16, pages 30–34, 1973

[10] G. Lance and W. Williams. *A general theory of classification sorting strategies.* Computer Journal, Volume 9 ,pages 373–386, 1967

[11] E.M. Voorhees. *Implementing agglomerative hierarchical clustering algorithms for use in document retrieval.* Information Processing and Management, Volume 22, pages 465–476, 1986

[12] D. Defays. *An efficient algorithm for a complete link method.* The Computer Journal, Volume 20, pages 364–366, 1977

[13] C. Olson. *Parallel algorithms for hierarchical clustering*. Parallel Computing, Volume 21, pages 1313–1325, 1995

[14] T. Zhang, R. Ramakrishnan, and M. Livny. *BIRCH: a new data clustering algorithm and its applications.* Journal of Data Mining and Knowledge Discovery, Volume 1, Number 2, pages 141–182, 1997

[15] S. Guha, R. Rastogi and K. Shim. *CURE: a clustering algorithm for large databases.* ACM SIGMOD International Conference on Management of Data, pages 73–84, 1998

[16] G. Karypis, E.-H. Han, and V. Kumar. *CHAMELEON: hierarchical clustering using dynamic modeling.* IEEE Computer, volume 32, number 8, pages:68–75, 1999

[17] R. Ng and J. Han. *Efficient and effective clustering methods for spatial data mining.* In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), pages 144–155, Santiago, Chile, 1994

[18] T.M. Mitchell. Machine Learning. McGraw-Hill, New York, 1997

[19] M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander. *Optics: ordering points to identify the clustering structure.* In Proceedings ACM SIGMOD International Conference on Management of Data, pages 49–60, Philadelphia, PA, USA, 1999

[20] A. Hinneburg and D. Keim. *An efficient approach to clustering large multimedia databases with noise.* In Proceedings of the 4th ACM SIGKDD, pages 58–65, New York, NY, USA, 1998

[21] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, 1995.

[22] T. Kohonen, *Self-Organized Formation of Topologically Correct feature Maps*, Biological Cybernetics, 1982.

[23] Vesanto J., 1997, Ph.D. Thesis, Helsinky University of Technology hinn 1998

[24] C. M. Bishop, M. Svensen, C.K.I. Williams, *GTM: The Generative Topographic Mapping*, Neural Computation, volume 10, number 1, 1998.

[25] D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[26] J.H. Holland. *Adaptation in Natural and Articial Systems.* University of Michigan Press, Ann Arbor, 1975.

[27] J. Koza, *Genetic Programming: A Paradigm for Genetically Breeding Population of Computer Programs to Solve Problems.* Stanford University, 1990

[28] I. De Falco, A. Della Cioppa, E. Tarantino. *Discovering interesting classi¯cation rules with genetic programming.* Applied Soft Computing, No. 1, pages 257-269, 2002.

[29] S. Forrest, B. Javornik, R. E. Smith, A. S. Perelson: *Using Genet- ic Algorithms to Explore Patter Recognition in the Immune System.* Evolutionary Computation, volume 1, Number 3,pages 191-211.

[30] L.O. Hall, B. Ozyurt and J.C. Bezdek: *Clustering with a genetically optimized approach.* IEEE Transactions on Evolutionary Computation, volume 3, number 2, pages 103–112, 1999

[31] B. Barmer, H. J. Hamilton. *Extracting Share Frequent Itemsets with Infrequent Subsets*. Data Mining and Knowledge Discovery, volume 7, pages 153-185. 2003

[32] M. Kubat, R. C. Holte and S. Matwin. *Machine learning for the detection of oil spills in satellite radar images*. Machine Learning, volume 30, number 1, pages 195-215.

[33] D.D Lewis and A. G. Gale. *A sequential algorithm for training text classifiers.* In Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Dublin, Ireland, pager 3-12, 1994.

[34] M. A. Shala, M. K. Arora and J. Elgy. *CAFCAM: Crisp And Fuzzy Classification Accuracy Measurement Software*, 7th International Conference on Geocomputation, Sept. 8-10, Southampton, UK 2003

[35] DG Rossiter. *Technical Note: Statistical methods for accuracy assesment of classified thematic maps.* Technical Report ITC, Enschede, NL. April 2004.

[36] J.A. Swets. *Measuring the accuracy of diagnostic systems.* Science, volume 240, number 4857, pages 1285-1293. 1988

[37] T. Fung and E. LeDrew. *The Determination of Optimal Threshold Levels for Change Detection using Various Accuracy Indices*. Photogrammetric Engineering & Remote Sensing, volume 54, pages 1449-1454. 1988

[38] Z. Ma and R.L. Redmond, T*au coefficients for accuracy assessment of classification of remote sensing data*. Photogrammetric Engineering & Remote Sensing, volume 61, pages 435-439. 1995

[39] R.G. Congalton, R. G. Oderwald and R. A. Mead, *Assessing Landsat Classification Accuracy using Discrete Multivariate Analysis Statistical Techniques*, Photogrammetric Engineering and Remote Sensing, Vol. 49, pages 1671-1678. 1983,

[40] M. Meilă, *Comparing clusterings by the variation of information.* Proceedings of the Sixteenth Annual Conference ofn Computational Learning The ory (COLT). Springer. (2003)

[41] M. Meilă, *Comparing clusterings—an information based distance*. Journal of Multivariate Analysis Volume 98, Issue 5, Pages 873-895 2007

[42] G. H. Rosenfield and Fitzpatrick-Lins, *A coefficient of agreement as a measure of thematic classification accuracy*. Photogrammetric Engineering & Remote Sensing, volume 52, pages 223-227, 1986.

[43] K. Wagstaff and C. Cardie, *Clustering with Instance-level Constraints*. Proceedings of the Seventeenth International Conference on Machine Learning, pages 1103-1110, 2000.

[44] W.M. Rand. *Objective criteria for the evaluation of clustering methods.* Journal of the American Statistical Association, volume 66, pages 846-850, 1971.

[45] D. L. Wallace. *Comment*. Journal of the American Statistical Association, volume 78, number 383, pages 569-576, 1983.

[46] E. B. Fowlkes and C. L. Mallows. *A method for comparing two hierarchical clustering.* Journal of the American Statistical Association, volume 78, number 383,pages 553-569, 1983.

[47] Asa Ben-hur, A. Elisseeff, and I. Guyon. *A stability based method for discovering structure in clustered data.* In Pacific Symposium on Biocomputing, pages 6-17, 2002.

[48] Boris Mirkin. *Mathematical classification and clustering.* Kluver Academic Press, 1996.

[49] B. Larsen and C. Aone. *Fast and effective text mining using linear time document clustering.* In Proceedingsof the conference Knowledge Discovery and Data Mining, pages 16-22, 1999.

[50] M. Meïla and d. Heckerman, *An experimental comparison of model-based clustering methods.* Machine Learning, volume 42, namber 1/2, pages 9-29, 2001.

[51] Stijn van Dongen. *Performance criteria for graph clustering and Markov cluster experiments.* Tecnical Report INS-R0012, Centrum voor Wiskundeen Informatica, 2000.

[52] AK Jain, RC Dubes. *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall; 1988.

[53] S. Dudoit and J. Fridlyand, *A prediction-based resampling method for estimating the number of clusters in a dataset*, Genome Biol, volume 3, pages research00, 2002.

[54] B. Di Eugenio and M. Glass. *The kappa statistic: A second look*. Computational Linguistics , volume 30 , Issue 1, pages 95-101, 2004.

[55] G. Saporta, G. Youness. *Comparing Two Partitions: some proposals and Experiments,Proceedings in Computational Statistics edited by Wolfgang Härdle*, Physica-Verlag, Berlin, 2002.

[56] G. Youness and G. Saporta. *Some Measures of Agreement Between Close Partitions* . Student, volume 5, number 1,pages 1-12, 2004.

[57] J.F. Marcotorchino and N. EL Ayoubi, *Paradigme logique des écritures relationnelles de quelques critéres fondamentaux d'association*, Revue de Statistique Appliquée, XXXIX , volume 2, pages 25-46, 1991

[58] G.C. Tseng and W.H. Wong, (2005) *Tight clustering: a resampling-based approach for identifying stable and tight patterns in data*. Biometrics, volume 61, pages 10-16.

[59] M. Ester,H. P. Kriegel,J. Sander and X. Xu. A *density-based algorithm for discovering clusters in large spatial databases with noise.* In Proceedings of the 2nd ACM SIGKDD, pages 226–231, Portland, OR, USA, 1996

[60] J. B. MacQueen.*Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, volume 1, pages 281-297, 1967

[61] M. Halkidi, Y. Batistakis,M. Vazirgiannis. *On clustering validation techniques*. Journal of Intelligent Information System, volume 17, number 2-3, pages 107-145,2002.

[62] L. HUBERT,P. ARABIE, *Comparing Partitions*. Journal of Classication, vol. 2, pages 193-218, 1985

[63] D. Frossyniotis and A. Stafylopatis, *A Multi-SVM Classification System*, Book: Multiple Classifier Systems, ISBN: 978-3-540-42284-6, pages 198-207, 2001.

[64] A. Thalamuthu, I. Mukhopadhyay, X. Zheng and G. C. Tseng. *Evaluation and comparison of gene clustering methods in microarray analysis.* Bioinformatics, volume 22, number 19, pages 2405-2412, 2006.

[65] S. Gustafson,L. Vanneschi, *Crossover-Based Tree Distance in Genetic Programming* Evolutionary Computation, IEEE Transactions on Volume 12, Issue 4, Pages 506-524, Aug. 2008.

[66] Berkhin Pavel , *Survey Of Clustering Data Mining Techniques*, book: Grouping Multidimensional Data, ISBN:978-3-540-28348-5, pages 25-71,2006.

[67] N. Grira, M. Crucianu and N. Boujemaa, *Unsupervised and semisupervised clustering: a brief survey*, in 'A Review of Machine Learning Techniques for Processing Multimedia Content', Report of the MUSCLE European Network of Excellence (FP6), 2004

[68] Book : Grouping Multidimensional Data: Recent Advances in Clustering (Hardcover) ISBN: 978-3-540-28348-5

[69] R. O. Duda, P. E. Hart, *Use of the Hough transformation to detect lines and curves in pictures* Commun. ACM, vol. 15, pages 11-15, 1972.

[70] B. Tjaden, J. Cohen *A Survey of Computational Methods Used in Microarray Data Interpretation* Book: Applied Mycology and Biotechnology,ISBN: 044451807X 9780444518071,pages 161-178, 2006.

[71] EM. Southern, *Detection of specific sequences among DNA fragments separated by gel electrophoresis.* J Mol Biol, volume 98, number 3, pages 503-517, 1975.

[72] P.O. Brown, D. Botstein. *Exploring the new world of the genome with DNA microarrays.* Nature Genet. volume 21, pages 33-37, 1999.

[73] R.J. Lipshutz, S.P.A. Fodor, T. R. Gingeras and D.J. Lockhart. *High density synthetic oligonucleotide arrays.* Nat. Genet. (Suppl.) volume 21, pages 20–24, 1999

[74] P.T. Spellman et al. (1998) *Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization.* Mol. Biol. Cell, vlume 9, pages 3273–3297, 1998

[75] K.R. Christie et al. *Saccharomyces Genome Database (SGD) provides tools to identify and analyze sequences from Saccharomyces cerevisiae and relat-*

ed sequences from other organisms. Nucleic Acids Res., 32, (Database issue), D311–D314, 2004.

[76] A. Ciaramella et al. *A Multifrequency Analysis of Radio Variability of Blazars.* J. Astron. Astrophys., volume 419, pages 485–500, 2004.

[77] R. Tagliaferri et al. *Soft computing methodologies for spectral analysis in cyclostratigraphy.* Comput. Geosci., volume 27, pages 535–548, 2001.

[78] J.P. Townsend(2004) *Resolution of large and small differences in gene expression using models for the Bayesian analysis of gene expression levels and spotted DNA microarrays.* BMC Bioinformatics, pages 5-54, 2004

[79] R. D. Wolfinger et al. *Assessing gene significance from cDNA microarray expression data via mixed models.* J. Comput. Biol., volume 8, pages 625–637, 2001.

[80] O. Ermolaeva et al. *Data management and analysis for gene expression arrays.* Nat. Genet., volume 20, pages 19–23, 1998.

[81] J. P. Townsend and D. L. Hartl, *Bayesian analysis of gene expression levels: statistical quantification of relative mRNA level across multiple treatments or samples.* Genome Biol., volume 3,pages research0071.1–0071.16, 2002

[82] Chen,Y. et al. *Ratio-based decision and the quantitative analysis of cDNA microarray images.* J. Biomed. Opt., pages 364–374, 1997.

[83] M. K.Kerr and G. A. Churchill *Statistical design and the analysis of gene expression microarray data.* Genet. Res., volume 77, pages 123–128, 2001.

[84] G. C. Tseng et al. *Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects.* Nucleic Acids Res., volume 29, pages 2549–2557, 2001.

[85] U. de Lichtenberg et al. *Comparison of computational methods for the identification of cell cycle-regulated genes.* Bioinformatics, volume 21, 1164–1171, 2005.

[86] T. R. Hughes et al. *Functional discovery via a compendium of expression profiles.* Cell, volume 102, pages 109–126, 2000.

[87] E. Oja, H. Ogawa and J. Wangviwattana *Learning in nonlinear constrained Hebbian network*. In Kohonen,T. et al. (eds.), Artificial Neural Networks. North-Holland, Amsterdam, p. 385, 1991.

[88] E. Oja, J. Karhunen, L. Wang and R. Vigario, *Principal and independent components in neural networks—recent developments.* World Scientific Publisher, Singapore, pages 16–35, 1996.

[89] Karhunen,J. and Joutsensalo,J. (1994) Representation and separation of signals using non-linear PCA type learing. Neural Netw., 7, 113–127.

[90] Karhunen,J. and Joutsensalo,J. (1995) Generalizations of principal component analysys, optimization problems and neural networks. Neural Netw., 8, 549–563.

[91] Hyv*ä*rinen,A., Karhunen,J. and Oja,E. (2001) Independent Component Analysis. John Wiley & Sons, New York.

[92] K. Chang *Nonlinear Dimensionality Reduction Using Probabilistic Principal Surfaces*, PhD Thesis, Department of Electrical and Computer Engineering, University of Texas at Austin, USA, 2000.

[93] K. Chang and J. Ghosh *A unified model for probabilistic principal surfaces.* IEEE Trans. Pattern Anal. Mach. Intell., 23, n1, 2001.

[94] O. Troyanskaya et al. *Missing value estimation methods for DNA microarrays.* Bioinformatics, volume 17, number 6, 2001.

[95] K. Hyunsoo, G. H. Golub and H. Park. *Missing value estimation for DNA microarray gene expression data: local least squares imputation* Bioinformatics, volume 21, number 2, 2005.